UNIVERSIDAD CENTROCCIDENTAL “LISANDRO ALVARADO”
DECANATO DE CIENCIAS Y TECNOLOGÍA
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

CONSTRUCCIÓN DE UN MOTOR DE BÚSQUEDA DE CONTENIDOS EN REPOSITORIOS CONFIABLES, BASADO EN CRAWLERS, ENMARCADO EN UNA ARQUITECTURA ORIENTADA A SERVICIO

Trabajo presentado para optar al grado de
Magister Scientiarum

Por: Edward J. Valera G.

Barquisimeto, 2012

CONSTRUCCIÓN DE UN MOTOR DE BÚSQUEDA DE CONTENIDOS EN REPOSITORIOS CONFIABLES, BASADO EN CRAWLERS, ENMARCADO EN UNA ARQUITECTURA ORIENTADA A SERVICIO

Por: Edward J. Valera G.

Trabajo de grado aprobado

_____ (Jurado 1)
Ramón A. Valera. A.

_____ (Jurado 2)

_____ (Jurado 3)

Barquisimeto, ___ de _____ de 2012

DEDICATORIA

A ti mi adorado Dios que me has dado la vida, salud, sabiduría, conocimiento para culminar este trabajo.

A mi bella esposa que es el amor de mi vida.

A mi hermoso hijo Adrián José Valera el cual me da el impulso para alcanzar nuevos objetivos.

A mis padres Omar Valera y Ofelia de Valera quienes me brindaron su apoyo incondicional, compensación y paciencia en todo este tiempo.

A mi querida hermana, Ana Karina, quien me prestó en muchas ocasiones su ayuda para culminar este trabajo de grado.

A mis queridos hermanos en Cristo Samuel Paris, Lisa Paris y los miembros de la célula de Villas de Yara que siempre están apoyándome en oración.

A mi iglesia bautista Cristo vive, quien ha comprendido y por las oraciones de cada uno de los miembros.

AGRADECIMIENTOS

Agradezco a mi Dios Amado por llegar a mi vida por brindarme su amor a través del sacrificio en la cruz que hizo Jesucristo para salvarnos de nuestros pecados. Por darme el fruto del espíritu que se resume en amor, gozo, paz, paciencia, benignidad, bondad y la fe en ti mi Dios, ya que esto me ayudo alcanzar este objetivo. Por estar conmigo en todos los momentos de mi vida. Por tu fidelidad, tu infinito amor y dedicación. Por darme el privilegio de ser tu hijo y servidor. Te agradezco señor por siempre mostrarme el camino que tu quieres que yo siga para contribuir en el crecimiento de tu obra.

Le Agradezco a mis padres, por brindarme su apoyo, ofrecerme sus cuidados, por darme fortaleza en los tiempos más difíciles.

Agradezco a mi guía espiritual Samuel Paris a Lisa su esposa, a mi Pastor porque se siempre estaba presente en sus oraciones.

Agradezco a mi tutor por orientarme y guiarme académicamente para culminar este objetivo propuesto.

ÍNDICE

ÍNDICE DE CUADROS

CUADRO

ÍNDICE DE ILUSTRACIONES

IMAGEN

UNIVERSIDAD CENTROCCIDENTAL “LISANDRO ALVARADO”
DECANATO DE CIENCIAS Y TECNOLOGÍA
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

CONSTRUCCIÓN DE UN MOTOR DE BÚSQUEDA DE CONTENIDOS EN REPOSITORIOS CONFIABLES, BASADO EN CRAWLERS, ENMARCADO EN UNA ARQUITECTURA ORIENTADA A SERVICIO

Autor (a): Edward J. Valera G.
Tutor (a): Ramón Valera

RESUMEN

El presente estudio tiene como objetivo desarrollar un motor de búsqueda en repositorios confiables, usando arquitectura orientada a servicios (SOA), acoplando un bus de integración (ESB) como middleware y tecnología NoSQL como almacén de datos; de esta forma dar respuesta a las necesidades de usuarios, en encontrar información de calidad en La Web y poder abarcar la información de La Web Invisible.

La investigación se circunscribe en una metodología de Proyecto Especial, apoyándose en una Investigación Documental para poder recabar información de la situación existente e identificar los requisitos. La misma se estructura en cuatro fases para cubrir cada uno de los objetivos específicos establecidos; las cuales están acopladas con técnicas y métodos de ingeniería de software, usando Programación Extrema (XP).

En la primera fase, a través de recopilación y análisis documental, se describe el problema de La Web Invisible y se realiza un recorrido del funcionamiento de los métodos de búsqueda basados en Crawlers, lo que permite describir el sistema y determinar los requisitos. En la segunda fase, se hace el diseño de la arquitectura y atributos de metadata; además de describir la tecnología a usar. En la tercera fase se realiza la programación de los componentes y servicios del motor de búsqueda; y una última fase de validación tecnológica, que ayudará determinar si la propuesta cumple con los requisitos planteados.

La propuesta está limitada a realizar un barrido de cierta cantidad de repositorios confiables, configurables llamados “semillas”, valiéndose de un método de búsqueda dirigido.

Palabras Claves: Motor de Búsqueda, Repositorio Confiable, SOA, ESB, NoSQL.

INTRODUCCIÓN

La sociedad contemporánea está marcada por los avances tecnológicos de los medios de comunicación; en este sentido, la disponibilidad de acceso a estas tecnologías, ha multiplicado las posibilidades de aprendizaje en una sociedad en constante cambio; esto debido a que el volumen de información crece de forma exponencial. Así, La Internet o La Web ha llegado a ser el medio de publicación más usado, desplazando incluso a las bibliotecas tradicionales; y es que según una entrevista realizada a Hamadoun Touré por Afp (publicada en El Mundo, 2011), secretario general de la Unión Internacional de Telecomunicaciones (UIT), en el 2000 habían solo 250 millones de usuario en internet; ya para el 2005 la cifra alcanzaba 1000 millones, pero a principios del 2011 la cantidad de usuarios llegó a 2000 millones; apenas en 5 años esa cantidad se duplicó.

El número de páginas web también se incrementa día tras día y los usuarios padecen de sobrecarga de información; pero el exceso de información no ayuda necesariamente en la optimización del proceso de aprendizaje y, consecuentemente, en la construcción de conocimiento; éste problema se hace más agudo cuando se desea acceder a información especializada y es cuando la búsqueda puede convertirse en algo frustrante para el investigador. Las ventajas de La Web son ampliamente conocidas; sin embargo, presenta a su vez nuevos riesgos y desafíos para la calidad de la información contenida en la misma.

Por ello este escenario exige que se desarrollen diferentes métodos para la búsqueda de información, requiriendo especial atención en las estrategias de búsqueda, para así limitar los efectos adversos de la información poco confiable y sin supervisión, ya que se ha generado una entropía de la misma; en tal sentido un grupo de organismos se han dado la tarea de investigar, establecer parámetros y delinear una serie de características para determinar la confiabilidad de un repositorio; ayudando a universidades, centros culturales, museos, entre otros a emplear sitios web con contenido confiable.

De este modo, tan importante como la calidad del contenido en La Web, es

disponer de un sistema de búsqueda que localice y permita el acceso a la información. Entre tantos métodos, el de uso más común, son los buscadores basados en Crawlers, los cuales constituyen una serie programas o “robots” que visitan páginas, las leen, las procesan y extraen su contenido.

Sin embargo, los buscadores actuales solo pueden accesar a una parte de La Web, aunque es una gran cantidad, pero solo constituye una pequeña parte de toda la información disponible, dejando a un lado a una gran porción, que muchos autores la denominan “Web Invisible”. Termino usado para referirse a aquellas páginas Web que no pueden ser indexadas o alcanzadas por los motores de búsqueda; bien sea porque no existen enlaces que hagan referencias a ellas o porque su contenido es privado.

La Web es tan grande que según Castillo (2004), es imposible abarcarla totalmente, debido a que la información de una página que se actualiza constantemente conlleva a otras páginas que también se actualizan constantemente, esto, aunado a la cantidad de información que se añade diariamente, el número de páginas en la Web puede llegar a ser potencialmente infinitas. Adicionalmente, Gruchawka (2005) asevera “Si conocemos los URLs de los sitios confiables dentro de la Web Profunda, sabiendo que tipo de información contiene, cada quien puede accesar a ellas pero puede convertirse en una tediosa tarea”.

Por tal motivo, en vez de tratar de abarcarla toda, ¿Cómo enfocarnos en rastrear solo repositorios cuyo contenido sea confiable?. Además, otro factor importante a considerar es la colaboración que proveen los buscadores actuales para compartir sus datos o conocimiento con otros sistemas externos; entonces, ¿Cómo hacer que estos pueda interactuar con otros sistemas, incluso desarrollados en distintas plataformas?, ¿Qué almacén de datos usar para mejorar los tiempos de repuesta? Y ¿Qué tipo de metadata emplear, para poder ser entendida por otros sistemas?.

Se han realizado algunos estudios y propuestas para conseguir el entendimiento o cruce de información entre los sistemas de forma automatizada; uno de los recursos usados para tal fin es el lenguaje XML para la descripción de los metadatos; a través del cual permite asociar información semántica a los datos. Sin embargo, esto no es suficiente para que varios actores independientes puedan colaborar para conseguir sus

objetivos. Para ello, es necesarios definir una ontología, es decir, describir los conceptos y las relaciones de un dominio común para que pueda existir una comunicación entre los distintos actores; o como lo define la W3C (2010) “Son conceptos y relaciones (también referidos como términos) usados para describir y representar un área específica”.

La importancia de esta temática radica en que el usuario necesita sentir seguridad de la información que consigue en La Web, como principal repositorio de datos que representa; por tal motivo se decidió acometer esta problemática en el presente estudio, que cada día se hace más evidente. Se analizan los métodos actuales de búsqueda y se determinan sus requisitos, componentes y arquitectura, para así, realizar un motor de búsqueda, enfocado solo en explorar repositorios que cumplan con requisitos de confiabilidad, tales como: accesibilidad del contenido, interoperabilidad, reusabilidad, extensibilidad, facilidad de localización, facilidad de gestión de contenidos y perdurabilidad (Sánchez-Alonso, Ovelar y Sicilia, 2010); además de enmarcar el sistema en una Arquitectura Orientada a Servicios (SOA) para poder darle acceso a los componentes del motor de búsqueda a sistemas externos; incluso se implementará fusionando con un bus de integración (ESB) para poder darle escalabilidad a la arquitectura y permitir que distintas tecnologías se puedan comunicar.

En sentido general W3C, definen los WebServices como un sistema de software diseñado para soportar la interoperabilidad maquina a máquina sobre distintas plataformas en una red. Así, estos permiten la integración de servicios desarrollados en plataformas heterogéneas. El concepto de WebServices es dado tanto para describir en sentido general el método de comunicación como para aquellos servicios basados en SOAP (Simple Object Access Protocol) y WS-*; en este trabajo el termino se usará de manera general y se hará distinción entre los servicios basados SOAP como en REST (Representational State Transfer), al momento de su especificación.

Ahora bien, es importante considerar que una SOA puede ser implementada utilizando WebServices; ya sea haciendo uso de los tradicionales, basados en SOAP y estándares WS-*, que son diseñados para ser integrados con otros, agregando

características como confiabilidad, transaccionalidad, seguridad, entre otros; o haciendo uso de Servicios REST, un estilo arquitectural que ha emergido en los últimos años como alternativa de uso de los WebServices y ha tenido un gran auge debido a su facilidad de uso y entendimiento según lo señala Hansen (2007).

Con estos beneficios y partiendo de la premisa de que el conocimiento humano le pertenece a la sociedad, se puede pensar en una alternativa, donde diversos motores de búsqueda enfocados en repositorios confiables, cada cual con su propia base de datos y abarcando un dominio especifico, todos interconectados entre sí, pueden ayudar en la optimización del proceso de aprendizaje y en la construcción de conocimientos; ergo, este enfoque se puede alcanzar usando como base esta investigación.

El estudio; enmarcado en la modalidad de Proyecto Especial, sustentada con investigación documental, ya que se revisan a fondo los conceptos y teorías necesarios para el desarrollo de la misma, se desarrollo según la siguiente estructura:

Capítulo I, en donde se desarrolla el planteamiento del problema para enfocar y enmarcar el estudio de manera clara y precisa la situación actual que se estudiará. Se exponen los objetivos que se deben cumplir para lograr la intención del trabajo, y se justifica la razón por la cual es importante solucionar el problema planteado.

Capítulo II, fundamentado en el marco teórico, conformado por los antecedentes y bases teóricas en donde se establecen las teorías y experiencias anteriores inherentes al tema. Este es el punto de partida para comprender la situación actual del problema, realizando un proceso de documentación que permita conseguir avales teóricos y los trabajos hechos con anterioridad para determinar el estado del arte.

Capítulo III; en este capítulo se exponen los métodos, técnicas y procedimientos que son necesarios aplicar para lograr resolver el problema planteado, es decir, describe las consideraciones metodológicas del proceso de investigación. Dadas las condiciones del estudio, se presentan cuatro fases, dando respuesta a los objetivos específicos, y en la cual se describe el método científico que se utilizará, el análisis de la información recopilada, la especificación de los requisitos, el diseño y

desarrollo de la propuesta y en última instancia la validación de la misma.

Capítulo IV, desarrollo de la propuesta del estudio; acoplada con las fases erigidas en la metodología; estas fases serán necesarias para producir el software de motor de búsqueda dirigido, denominado ARANEA; en este capítulo, además se presentan los principales artefactos UML que describen las diversas vistas del sistema en estudio.

Finalmente el Capitulo V, Conclusiones y Recomendaciones. Donde se podrá colegir los resultados del estudio y algunas recomendaciones enfocadas al software de motor de búsqueda ARANEA.

CAPITULO I

EL PROBLEMA

Planteamiento del Problema

Desde tiempos inmemoriales el ser humano ha tenido la necesidad de escribir lo que piensa, siente, sabe; en fin, todo lo que necesita hacer saber; por lo que ha inventado medios por el cual hacer sentir sus ideas. Igualmente ha tenido esa exigencia de buscar información, de poder sentirse satisfecho con las interrogantes que le conducen a dicha búsqueda; es por ello que tanto el que informa como el que busca, convergen en una mismo canal pero que por el crecimiento inminente y exponencial de la cantidad de información se convierte en muchas vías, pero sin dirección, que con el tiempo produce dificultad a la hora de encontrar aquello que en realidad necesita.

En 2003 el escritor y filósofo Umberto Eco (citado en Castillo, 2004), considera la escritura como una extensión de la memoria humana. Eco, en el marco de la inauguración de la nueva Librería de Alejandría establece tres tipos de memoria:

> Nosotros tenemos tres tipos de memoria. La primera orgánica, hecha de carne y sangre, administrada por nuestro cerebro. La segunda es mineral, y en este sentido la humanidad ha conocido dos tipos de memoria mineral: hace miles de años esta memoria fue representada por tablillas de arcillas y obeliscos, muy bien conocidas en nuestro país, en el que las personas grabaron sus textos. Sin embargo, el segundo tipo es la de hoy en día representada por la memoria electrónica de los ordenadores actuales, basados en silicio. Tenemos otro tipo de memoria bien conocida, la vegetal, representada por los primeros papiros, también bien conocidos en nuestro país, y luego por los libros, hechos de papel (pp. 13-14).

Tomando en cuenta lo dicho por Eco, entonces La WORLD WIDE WEB o Internet o también llamada La Web, es una memoria mineral inmensa, que se ha convertido en pocos años en un gran esfuerzo cultural de la humanidad; equivalente, según Eco, en importancia a la primera biblioteca de Alejandría; pero la principal diferencia entre la biblioteca de Alejandría y La Web no es que una sea de origen vegetal, hecha de pergaminos y tinta, y la otra sea mineral, hecha de cables y señales digitales.

La principal diferencia es que mientras los libros de la colección de Alejandría fueron copiados a mano y revisados; la mayor parte de la información en La Web se ha revisado una sola vez, por su autor, en el momento de la escritura, esto de acuerdo a Klaudinyi (2010) y Kaisler (2003); aunado a que la memoria mineral moderna permite la reproducción rápida de la obra, sin el esfuerzo humano. El costo de la difusión de contenidos es más bajo debido a las nuevas tecnologías, y ha disminuido sustancialmente de la tradición oral a la escritura, y después de la impresión y la prensa a las comunicaciones electrónicas.

Esto ha generado mucha más información que se puede manejar así como muchos más problemas en cuanto a la búsqueda de información confiable, problema al que autores como Castillo (2004) lo llama “el problema de la abundancia”; en pocas palabra, a pesar de tener gran cantidad de información, esta no puede ser verificada para su publicación; ya que hoy en día se vive la era de la explosión de la información, donde la misma en el 2007 estaba siendo medida en exabytes, de acuerdo a una investigación realizada por International Data Corporation (2008, citado por Short, Bohn y Baru, 2010) “Un estudio publicado en Marzo del 2008 por IDC, reporta que el total de la información del universo digital de la Web era aproximadamente de 281 exabyte” (p 14).

El Internet produce una sensación de estampida o arsenal de distintas ideas, de personas diferentes, de pensamientos y culturas distintas, donde pueden existir tantas y tantas probabilidades que resulta excesivo. Este crecimiento y la incorporación de cada vez más personas hace de lo que en un principio era una pequeña vía, se convierta en todo un planeta extremadamente conectado entre sí. Por ello, buscar

información en Internet se ha convertido en una actividad la cual está presente en cualquier proceso de enseñanza y cada vez es más común como recurso necesario para cubrir las necesidades de investigación.

Todo este gran cumulo de información, ha dado pie a crear métodos para encontrar la misma, es por eso que en los albores de la internet, la búsqueda de información se llevó a cabo principalmente mediante la exploración a través de listas de enlaces, recogidos y ordenados por los seres humanos de acuerdo con algunos criterios; pero luego cuando de miles de página la internet paso a tener millones y millones de ellas, surgieron los motores de búsqueda automatizados basados en Crawlers.

En 1994 la Dra. Ellsworth (citado por Ramírez, 2009) llamo “Internet Invisible”, para referirse a la información que los motores de búsqueda no pueden encontrar, aunque en realidad no es que sea Invisible sino que hay que saber llegar a ella; y es que en un estudio realizado por Varian y Lyman (2003), donde estiman cuanta información existen en medios digitales, entre ellos la encontrada en La Web, a través de una investigación de campo de tipo descriptiva y experimental, en resumen de sus resultados, encontraron que la información de La Web Invisible es hoy entre 400 y 550 veces mayor que la que abarcaba en el 2000; La Web Invisible contiene 91.850 terabytes de información, mientras La Web Superficial contiene sólo 167 terabytes. La primera contiene 126.500 millones de documentos mientras que la segunda sólo 33 mil millones. De los 60 sitios de La Web Invisible más grandes contienen 750 terabytes; se puede observar que esta cifra supera en 40 veces el tamaño de La Web Superficial y por último establecen que los sitios situados en La Web Invisible soportan un 50% más de tráfico que los sitios ubicados en La Web Superficial. Con estos resultados se puede palpar la importancia que tiene la búsqueda de los contenidos situados en La Web Invisible, que están siendo desperdiciados.

Ahora bien, Rodhenizer y Trudel (2007), Castillo, Marin, Rodríguez y Baesa-Yates (2004) y Pandia (2007) concuerdan que uno de los inconvenientes de los motores de búsqueda para encontrar información en La Web Invisible, es que tienen que tratar el gran tamaño y la tasa de cambio de La Web. Es así, como el número de

páginas va creciendo, será cada vez más importante centrarse en las páginas más "valiosas" o confiables. Sin embargo, el problema emerge cuando se desea acceder a esa información especializada confiable que se encuentra dentro de La Web Invisible.

Según una investigación realizada por NEC Research (2002, citado por Suárez y Plasencia, 2010) (véase la *Cuadro No. 1*), el buscador de mayor alcance, Google.com, dispone de 1,24 millones de páginas indexadas o catalogadas, apenas el 42% de todas las existentes en la Red. Aún a más distancia aparece el segundo buscador más potente, Fast, un buscador especializado en tecnología, cuyas 575 millones de páginas catalogadas le permiten acceder a sólo el 19% de la información disponible en la Red. Acá se puede observar el nivel de alcance que tienen los motores de búsqueda dentro de La Web; donde solo alcanzan una porción de ella; apenas indexan una cantidad de páginas de lo que pueda ofrecer La Internet.

Cuadro No. 1
Buscadores de Mayor Alcance

	Páginas catalogadas (millones)	**Alcance sobre el total de páginas (%)**
Google	1247	42
Fast	575	19
WebTop.com	500	17
InkTomi	500	17
Alta Vista	350	12
Nother Light	330	11
Excite	250	8

***Nota*.** Tomado de NEC Search, Search Engine Watch (2002, citado en Suárez y Plasencia, 2010, p. 4).

Según Bergman (2001), en su escrito The Deep Web: Surfacing Hidden Value; establece que en esa llamada Internet Invisible, existen una serie de sitios web

dedicados a exponer contenidos especializados, donde los mismos son evaluados por comités calificados y en general están conformados por revistas, reportes de investigación, tesis, trabajos de grados, en fin, una serie de documentos de calidad reconocida, incluso muchos de estos son reconocidos por organismos internacionales; pero a pesar de ello existen en La Web Invisible, es decir, lo motores de búsqueda tradicionales no logran llegar a este tipo de información.

Muchos de estos sitios son instituciones como museos, librerías, universidades, entre otros; que han tenido la necesidad de proveer acceso a su información a la mayor cantidad de usuarios posibles; eventualmente La Web es un medio muy atractivo para cumplir este objetivo, creando así repositorios digitales. Sin embargo, de acuerdo a lo dicho por Bergman (2001), este tipo de contenido frecuentemente permanecerá dentro de La Web Invisible, ya que no poseen enlaces a otras páginas ni otras páginas poseen enlaces a ellas, lo que las hace "invisibles" a los algoritmos de búsquedas de los grandes buscadores Web; esto en posición con lo escrito por Interpristor (2010). En este sentido, Boswell (2010) escribió: "La mayoría de la información de La Web Invisible es mantenida por instituciones académicas y tiene más alta calidad que la de los resultados de los motores de búsqueda" (párr. 13); lo que coincide con lo dicho por Bergman (2001) "El total de la calidad de contenido de La Web Profunda es de 1000 a 2000 veces más grande que la de La Web Superficial". (párr. 5).

A raíz de esto, algunas organizaciones se dieron la tarea de discutir cuales serian los métodos y estándares a seguir para considerar que un sitio de contenido digital o repositorio fuese confiable; el más destacado de ellos es Research Library Group (RLG), creado en 1974; y que ha realizado varios estudios y propuesto estándares. Hoy en día es parte de la división de investigación de Online Computer Librarie Center (OCLC), incluso funge junto The Center of Research Libraries (CRL) como ente certificador de sitios considerados Repositorios Confiables; adicionalmente han contribuido estableciendo una serie de parámetros y atributos para determinar si un repositorio se puede determinar cómo confiable.

En otro orden de ideas, Google es una de las principales herramientas usadas

como motor de búsqueda, pero el uso de este no garantiza un contenido académico confiable; los motores más populares usan métodos basados en búsquedas jerárquicas, comúnmente llamados Crawlers, Spiders o Arañas Web; donde recogen de manera automática información sobre los contenidos de los sitios web, “tejiendo” enlaces relacionados de cada página, para lo cual tienen un punto de partida llamado “semilla”, que no es más que una serie de enlaces por donde comienzan a “tejer” el motor de búsqueda.

La arquitectura de cualquier motor de búsqueda está compuesta por varios componentes cuya comunicación normalmente se establece a través del lenguaje de programación por el cual han sido desarrollados. Google, yahoo, bing, entre otros conocidos popularmente, no poseen interfaz pública para poder accesar a sus componentes, como por ejemplo; ir hasta su base de datos y extraer una porción de los enlaces web o usar su Crawler de búsqueda como componente en alguna otra aplicación, etc.; son sistemas que algunos a pesar de ser desarrollados en entornos libres su acceso es restringido. Mucho menos proveen métodos de integración heterogéneos, para dar soporte a cualquier aplicación escrita en otro lenguaje. Que interesante seria que estos componentes pudiesen dar servicio a peticiones realizadas por entes externos, incluso siendo desarrollados en plataformas distintas.

Ante este planteamiento, y tomando en cuenta la premisa de que existen atributos para estimar la confiabilidad del contenido de un sitio web como repositorios confiables, surge la necesidad de formular las siguientes interrogantes: ¿Que algoritmos y métodos son usados en la actualidad por lo buscadores tradicionales?, ¿Por qué no, crear una arquitectura de búsqueda basado en crawler dirigido a repositorios confiables?, ¿Cuales son los requisitos y componentes de un motor de búsqueda dirigido?, ¿De qué manera establecer mecanismos de integración entre los componentes de la arquitectura?, ¿Como modelar dichos componentes para que sean usados por otros sistemas como servicio?, ¿Cuales atributos considerar para estimar la confiabilidad de un dominio?; por último, ¿Qué tipo de estructura definirá la metadata del almacén de datos?.

Finalmente ante la situación planteada, se dará una solución en la presente

investigación, a través de la construcción de un motor de búsqueda de contenidos web, basado en crawlers dirigido o enfocado; enmarcado en una arquitectura orientada a servicio (SOA), en el cual confluyen diversos componentes coordinados y comunicados, por medio de un esquema de información basado en mensajes, mediante un middleware de integración.

Objetivos de la Investigación

Objetivo General

Desarrollar un Servicio de Motor Búsqueda de contenidos en repositorios confiables, basado en Crawlers, enmarcado en una arquitectura orientada a servicio (SOA), usando un bus de integración empresarial (ESB).

Objetivos Específicos

Elicitar el estado actual de los métodos, arquitecturas y componentes de las maquinas de búsqueda tradicionales; y requisitos a través de una análisis orientado a servicio.

Diseñar la arquitectura orientada a servicios, la estructura del almacén de datos por medio de una ontología y los atributos de un repositorio confiable.

Implementar los componentes del motor de búsqueda, y el bus de integración; de acuerdo a los requisitos planteados.

Validar tecnológicamente el motor de búsqueda y su arquitectura de acuerdo a los requisitos funcionales y no funcionales.

Justificación e Importancia

Día tras día, la cantidad de usuarios conectados a La Web aumenta, así como la cantidad de información no calificada; por lo tanto cada vez se hace más laborioso la búsqueda de cualquier tipo de información que sea confiable, sobre todo aquella requerida que se encuentra en La Web Invisible; es allí donde radica la principal importancia y justificación de esta investigación, donde la aplicación de este sistema permitirá la ejecución de una maquina de búsqueda, enfocada a seleccionar páginas Web o contenido de repositorios confiables, lo cual facilitará la localización de información de calidad.

Esta investigación también se justifica desde el punto de vista práctico, ya que la misma propone el diseño de la aplicación desde un enfoque arquitectural orientado a servicio, de forma que otras aplicaciones puedan reusar las funcionalidades que implementa; usando estrategias de integración basadas en mensajes. Igualmente se abordara la estandarización de la metadata siguiendo una ontología; permitiendo que los sistemas de búsqueda en un futuro sean más abiertos.

También desde el punto de vista teórico, esta investigación generará reflexión y discusión tanto sobre el conocimiento existente del área investigada, como dentro del ámbito de las Ciencias Sociales y Naturales, ya que de alguna manera u otra, se confrontan teorías.

Con el desarrollo de la arquitectura orientada a servicios, dará pie al aprovechamiento de cualquier infraestructura que permita el despliegue de aplicaciones distribuidas utilizando La Web; y abrirá nuevos caminos, para la búsqueda de conocimiento en las organizaciones.

Con el desarrollo de la propuesta se sientan las bases para el desarrollo de la línea de investigación y la generación de nuevos trabajos que partan del problema planteado, sirviendo como marco referencial.

Alcances y Limitaciones

La presente investigación, está enmarcada en el desarrollo de una arquitectura orientada a servicio (SOA), junto con los componentes del motor de búsqueda y el despliegue de los mismos en la arquitectura, bajo un bus de integración, a través del intercambio de mensajes estandarizado, dando escalabilidad y reusabilidad, empleando patrones de integración. Este no contempla la puesta en marcha en un entorno de producción de lo anterior descrito; solo se realizarán pruebas locales y estas no abarcarán el potencial de intermediación con otros entes.

Esta investigación se centra en la obtención de información de La Web Invisible en el contexto de las aplicaciones de recopilación de información dirigida, para dar soporte a las necesidades de búsqueda que se plantean. El enfoque propuesto se basará en un proceso de búsqueda dirigido; donde explorará periódicamente las fuentes objetivo para localizar información relevante, y la indexará para su posterior consulta.

La investigación no tratará sobre los requerimientos de seguridad en ninguna etapa del desarrollo, lo cual puede ser tratado como tema aparte o complementario en futuras investigaciones.

El proceso de búsqueda será de experimentación controlado, hacia el rastreo de repositorios especificados por un usuario administrador, el cual configurará las semillas o páginas cuyo calificativo sea confiable; por lo que la herramienta no cuenta con mecanismos que le permitan reconocer automáticamente la confiabilidad de los repositorios. Adicionalmente, solo abarcará repositorios públicos.

El sistema está previsto desarrollarse usando JAVA como lenguaje de programación, por ello solo se podrá implementar en Sistemas Operativos que satisfaga su máquina virtual.

El motor de búsqueda solo contempla la búsqueda de contenido obviando las imágenes o algún otro tipo recurso multimedia. Igualmente, tampoco contempla el desarrollo de los buscadores de la emergente Web 3.0; es decir, no hará búsquedas semánticas.

CAPITULO II

MARCO TEÓRICO

Antecedentes

Son muy diversas y variadas las investigaciones realizadas donde se observan serios intentos y propuestas arquitecturales para resolver el problema de la búsqueda de información de contenidos confiables a través de La Web; así como metodologías que ponen en práctica sin traumas, la implementación del paradigma SOA junto con métodos de integración apropiados.

En esta sección se hará mención de aquellos trabajos que han antecedido a la presente investigación, que de alguna manera u otra han ayudado a la misma y han servido de base para la solución del problema planteado.

Papazoglou y Van Den Heuvel (2005); publicaron un escrito en Very Large Data Bases Journal (VLDB) donde exponen las tecnologías y enfoques que unifican los principios y conceptos de SOA. En el escrito ponen de manifiesto los problemas que tienen las empresas modernas para integrarse tecnológicamente con sus clientes o asociados de negocio, lo que trae consigo la necesidad de integrar también sus procesos de sus negocios, ya sean internos o externos; esto por supuesto requiere de una implantación que sea rápida y que los servicios de los negocios se ensamblen con quien lo necesite de una manera transparente.

El documento se centra en el ESB y describe una serie de funciones que están diseñadas para ofrecer un “backbone” de fácil uso para SOA, basadas en estándares, lo que amplía la funcionalidad del middleware; todo mediante la conexión de componentes y sistemas heterogéneos y además ofrece servicios de integración.

Por último, el documento propone un enfoque para ampliar la SOA

convencional, para así satisfacer los requerimientos esenciales de un ESB, que incluyen capacidades, tales como: orquestación de servicios, enrutamiento "inteligente", aprovisionamiento, integridad y seguridad del mensaje, así como la gestión de servicios.

Como producto final, propusieron una arquitectura SOA, junto con un ESB como middleware, que se puede observar en la *Figura No. 1*; esta propuesta es importante para el estudio ya que da una visión de la arquitectura principal del motor de búsqueda a construir. A pesar que dicha publicación fue realizada en el 2005, aún mantiene vigencia ya que los traumas de diseño de arquitectura SOA, hoy en día continúan siendo un problema para establecer esta filosofía en los sistemas; por ello siguiendo su propuesta de una SOA extendida con un middleware ESB, aunado a su experiencia, garantizan un diseño arquitectural bastante robusto y escalable. Por ello se hará consideración a su propuesta de SOA extendido que se ajustará a nuestro caso de estudio, articulando junto a las propuestas de arquitecturas de motores de búsqueda realizadas por Seeger y Álvarez M.

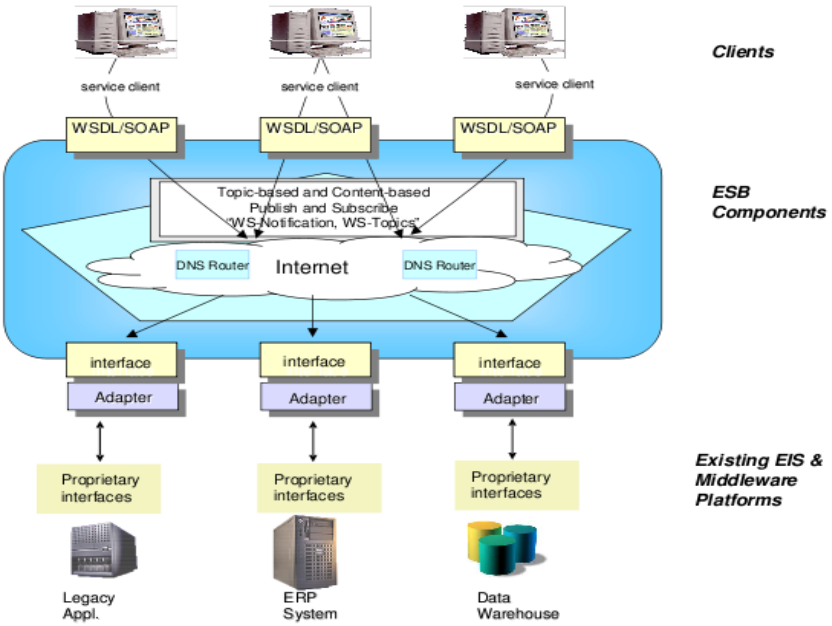

Figura No. 1. **Arquitectura Propuesta SOA y ESB.** Tomado de Papazoglou y Van Den Heuvel (2005, p. 18).

Papazoglou y Van Den Heuvel (2006), como expertos en el área de arquitectura de software; realizaron otra publicación, esta vez en el International Journal of Web Engineering and Technology (IJWET); el cual tiene como objetivo, examinar una metodología de desarrollo de servicios desde el punto de vista tanto de los productores como de los consumidores de servicios, además hacen una revisión de la gama de elementos de esta metodología. En el mismo ponen de manifiesto la dificultad que presentan las empresas para implementar SOA. Por lo que en el artículo ofrecen una visión de los métodos y las técnicas utilizadas en el diseño y desarrollo SOA.

Para darle solución a tales dificultades, luego de revisar distintos métodos y técnicas, Papazoglou y Van Den Heuvel describen una metodología experimental para el diseño y desarrollo de una SOA; según los autores esta metodología refleja un intento en definir los principios del diseño y desarrollo que son aplicados tanto en los WebServices como en los procesos de negocio. La metodología toma en cuenta un conjunto de los modelos de desarrollo y varios escenarios para la realización de servicios.

En contraste con los enfoques tradicionales de desarrollo de software, la metodología en este artículo pone de manifiesto las actividades en torno a la prestación de servicios, la implementación, ejecución y seguimiento. Según los autores estas actividades serán cada vez más importante en el mundo de los servicios, ya que contribuyen al concepto de capacidad de servicio de adaptación, donde los servicios y procesos continuamente puede transformarse ellos mismos para responder a las demandas del medio ambiente y los cambios sin comprometer la eficiencia operativa y financiera.

Es por ello, que para el estudio, con el objeto de seleccionar los servicios e implementar la SOA con mayor facilidad, se seguirá la metodología propuesta por Papazoglou y Van Den Heuvel; donde la metodología describe paso a paso las actividades a realizar para lograr la arquitectura deseada.

Álvarez M. (2007), planteó en su tesis doctoral la construcción de una aplicación de recopilación automática de información de La Web Oculta utilizando un

enfoque dirigido. En ella propuso una arquitectura para Crawling dirigido a la información contenida en La Web Oculta (*ver Figura No. 2*), mediante una investigación de tipo experimental; apoyada en una investigación documental. Álvarez M. pone de manifiesto el problema de los buscadores actuales que se enfocan en toda La Web, y alega que es imposible abarcarla toda por lo que queda una gran cantidad de información sin localizar.

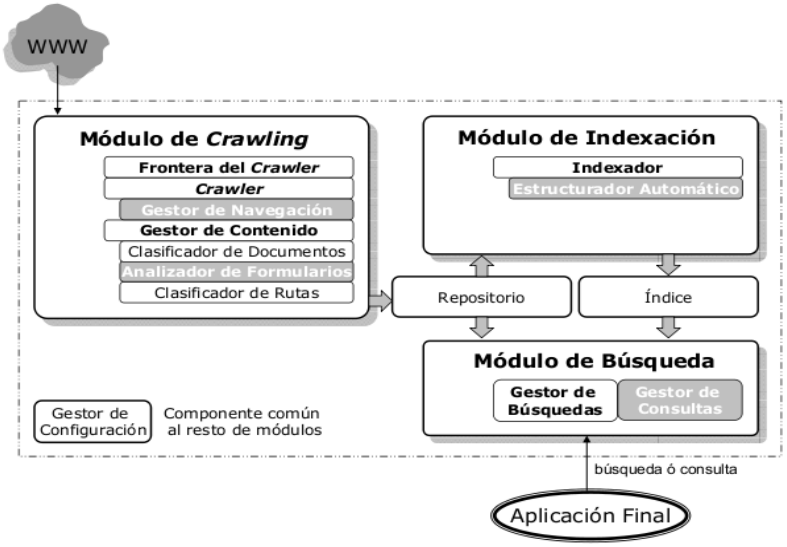

Figura No. 2. **Arquitectura de Motor de Búsqueda Dirigido.** Tomado de Álvarez, M. (2007, p 91).

Además de la arquitectura, también desarrollo un conjunto de técnicas y algoritmos para alcanzar aquellas páginas basadas en Ajax o Scripts que contienen formularios de búsqueda en base de datos.

Para realizar la validación experimental de los algoritmos, realizó un conjunto de pruebas con un prototipo de buscador llamado DeepWeb; en el mismo tomó varios dominios (libro, música y películas) y los resultados fueron medidos usando métricas de minería de datos como precisión, alcance y media armónica, que permitieron demostrar que es posible extraer información de La Web Oculta. Las pruebas se realizaron con fuentes web reales y de diversos tipos y en diferentes dominios de aplicación, de forma independiente para los principales componentes de la

arquitectura propuesta, con el propósito de obtener la efectividad individualmente.

En las mismas se obtuvieron porcentajes de efectividad muy altos, superando en prácticamente todos los casos el 90%, tanto en precisión como en alcance (recall) y consiguiendo el 100% en otros casos.

Álvarez M. concluyo que “Es posible diseñar soluciones de búsqueda dirigido para extracción de información de forma automatizada de La Web Oculta y que los esfuerzos para acceder a la información contenida en La Web Oculta son valiosos y deben de ser continuados.” (p. 165).

Este estudio es importante en el contexto de la presente investigación; debido a que a pesar de que Álvarez M. no se enfoca en la calidad de los contenidos, demuestra que es posible extraer información de La Web Invisible, mediante métodos y algoritmos propuestos en su tesis y además sienta las bases de una arquitectura básica de un motor de búsqueda dirigido que será tomada en cuenta como punto de partida.

Seeger (2010), presentó una tesis cuyo propósito fue la investigación e implementación de una buena arquitectura para colectar, analizar y manejar datos de sitios web sobre la escala de millones de dominios; su principal enfoque era el de tomar data de aquellos sitios basado en Content Managment System (CMS), en especial Drupal.

Para la realización de su tesis, Seeger hizo una serie de investigaciones de tipo documental, donde evaluó algunas posibilidades sobre el tipo de tecnología a usar de cada uno de los componentes de la arquitectura propuesta, determinando sus ventajas y desventajas y en ella baso su decisión de selección.

Al final obtuvo como producto una arquitectura basada en Crawler donde empleó como lenguaje Ruby y como Base de Datos decidió tomar una relacional (MySQL), adicionalmente también agregó métodos de indexación y distribución de balanceo de carga. A este producto se le realizaron una serie de pruebas de validación y verificación donde obtuvo que la implementación de la arquitectura podía procesar 100 dominios por segundo en un solo servidor; al final, luego de 12 días, el sistema logró obtener aproximadamente 100 millones de dominio.

Con estos resultados, Seeger razonó que a través de su arquitectura podría lograr almacenar millones de dominios e indexarlos. A pesar de que usó MySQL como almacén de datos, en sus conclusiones recomienda para futuras investigaciones el uso de una Base de Datos NoSQL como MongoDB, para mejorar la velocidad. Tanto esta recomendación como los resultados obtenidos por Google a través de su Base de Datos BigTable, han sido importantes para la toma de decisión del uso de tecnología NoSQL en el presente estudio.

Otro aspecto importante a considerar en esta tesis, es que Seeger da una idea inicial de como a través de un crawler se puede buscar información en redes sociales como twitter, en su caso específico, usando JSON; sin embargo esto se podría usar en otras redes sociales que aceptan JSON como facebook. Por último es importante considerar el método que se usó para la verificación y validación de la propuesta.

Suárez y Plasencia (2010), realizaron un análisis de la situación actual de los buscadores especializados desde el punto de vista de la vigilancia tecnológica y la inteligencia competitiva para determinar si los que ya existen satisfacen los requerimientos de búsqueda de información especializada y si resulta conveniente la construcción de un buscador especializado en software libre que sea capaz de obtener información actualizada de fuentes especializadas de información en Internet referentes a determinados sectores de actividad; para ello usaron una investigación documental.

Con la revisión de varios estudios e información bibliográfica verificaron que el propósito general de las grandes herramientas de búsqueda como Yahoo, Altavista, Google, Lycos, etc.; es precisamente dar una respuesta general pero tienen limitaciones para encontrar lo que se busca con mayor precisión. Algunos son particularmente efectivos y sofisticados pero ninguno es enteramente integral; además detectaron que la mayoría de los buscadores especializados son propietarios y que tienen altos precios de adquisición, de ahí que pensaron convenientemente, analizar las opciones de código abierto.

De sus resultados, concluyeron que resulta conveniente la construcción de un buscador especializado en software libre capaz de realizar búsquedas en un campo

concreto, por tal motivo platearon un diagrama de cómo debería funcionar el proceso de búsqueda de dicho motor, que se puede ver en la *Figura No. 3.*

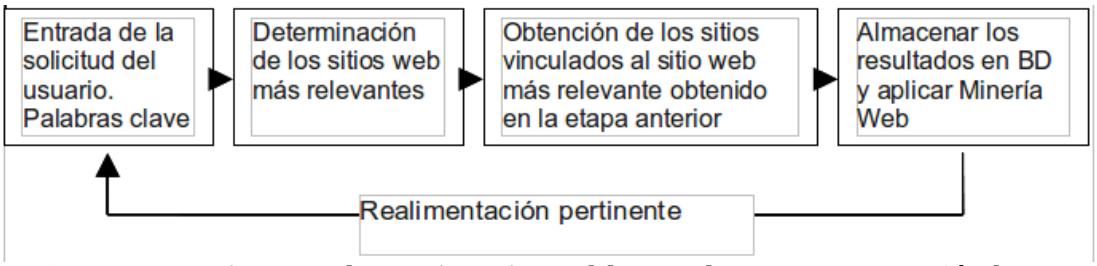

Figura No. 3. **Diagrama de Funcionamiento del Buscador Propuesto por Sánchez y Plasencia.** Tomado de Sánchez y Plasencia (2010, p. 6).

En este estudio Suárez y Plasencia, ven la necesidad de la construcción de un buscador cuyas semillas o páginas de inicio a buscar fueren páginas especializadas, es decir, sitios web que de alguna manera u otra contenga información de calidad de acuerdo a un ámbito que desee buscar el usuario; además como segundo punto plantea la necesidad de que dicho buscador sea desarrollado en software libre. De allí parte la idea de la presente investigación de que el objetivo principal del buscador sean repositorios confiables; además de tomar como base el diagrama de funcionamiento del motor de búsqueda y la idea de incorporar semillas especializadas iniciales. También se toma en cuenta la recomendación para la selección de una plataforma de código abierto como Java en la construcción de la solución.

La Universidad del Sur de la Florida (2010), realizó un trabajo para ayudar a sus estudiantes a analizar la información contenida en la Web; este fue un trabajo de tipo documental, apoyado en una investigación de campo para determinar sus resultados. En el estudio enfatizan el problema que existe en encontrar información de calidad en la Web por tres razones: “1) Cualquier persona puede colocar un Sitio Web y conocer el autor del mismo y sus credenciales, 2) El cambio constante de la información en la Web y 3) Frecuentemente el contenido colocado en la Web no es evaluado antes de su publicación” (p. 3).

El objetivo principal era determinar las características a tomar en cuenta para

determinar si un sitio Web es de calidad y así proveer a sus estudiantes un mecanismo de medición. Una de las premisas del estudio es que “la actividad de búsqueda no solo se debe limitar a encontrar el recurso sino también a evaluarlo; en donde analizar y evaluar el recurso es parte esencial del proceso de búsqueda” (p. 1); para lo cual se enfocaron en dos puntos a analizar: la confiabilidad y la validez del recurso.

Como resultado obtuvieron dos tipos de análisis; uno para analizar fuentes impresas y otras electrónicas; donde elaboraron una serie de preguntas organizadas por características. En el caso del análisis de fuentes electrónicas tomaros tres características principales a medir: Origen de la Documentación, Precisión y Actualidad.

Para verificar la eficacia del resultado del estudio, tomaron una muestra de estudiantes donde una parte busco una serie de tópicos usando los criterios dados y otra parte sin criterios y concluyeron que el 95% de los estudiantes que usaron los criterios encontraron información confiable y el otro grupo solo el 45%.

Para el caso de estudio se tomará en cuenta las características y preguntas las cuales tuvieron como resultado para analizar lo que son las fuentes electrónicas.

Greenwood y Steyn (2011), profesores de la Universidad de British Columbia, realizaron un trabajo de tipo documental, para determinar qué aspectos se deben tomar en cuenta para evaluar la credibilidad y confiabilidad de los recursos en internet.

Partieron del supuesto de que a pesar de que existe mucha información en la Web, no toda la que allí se encuentra es confiable, debido a la oportunidad que esta da para que cualquier persona pueda expresar su idea y además la facilidad de publicar información completa en la Web sin que sea monitoreada.

Como resultado de su trabajo, establecieron una lista de chequeo con una serie de atributos o características que se deben evaluar para determinar la confiabilidad de los recursos en internet y en cada una de ellos establecieron una serie de preguntas; así como un instructivo de cómo deben ser llenado e interpretado. Los atributos a evaluar son: el autor o fuente, precisión, actualización, objetividad, cobertura y propósito; además de describir los tipos de materiales y de fuentes.

La lista de chequeo de este trabajo dará un punto de partida junto con la investigación realizada por La Universidad de la Florida, para determinar la confiabilidad de acuerdo a las características planteadas.

Bases Teóricas

En esta sección se presentará un conjunto de conceptos y definiciones, con el fin de establecer un marco teórico que sustentará el desarrollo de la actual investigación.

El basamento teórico se centra en explicar el fenómeno de La Web Invisible, término del cual parte el problema. De igual forma se definirán términos básicos en cuanto a repositorios digitales, confiabilidad de los repositorios, los esfuerzos que han realizado organismos para linear sus características y el porqué de su creación.

Luego se expondrá cómo funcionan los motores de búsqueda basados en Crawlers, métodos de búsquedas y su arquitectura; explicar los almacenes de datos existentes, cuales son las principales categorías y escalabilidad. Se pretende en este apartado dejar claro lo que significa SOA; así como también el Bus de Servicio Empresarial (ESB) y el papel que juegan en las implantaciones SOA.

Adicionalmente se aborda el tema de los servicios Web y la correspondiente tecnología subyacente. En última instancia se define el tema de las ontologías; los metadatos y cuales son algunos de los estándares internacionales que soportan esta área.

Web Invisible

Pedraza (2009) define La Web como “un sistema hipertexto que funciona sobre Internet, permitiendo la consulta de documentos o páginas web y la navegación a través de las mismas” (*Sección* Evolución de La Web, párr. 1).

La Web o Internet, ha tenido un crecimiento exponencial año tras año; cada vez

se suman nuevas página y/o documentos en este gran mundo interconectado; es por ello, que autores como Bergman (2001), Varian y Lyman (2003) y la Dra. Ellsworth (1994); han dividido La Web en dos; la que llaman Web Superficial (Surface Web) y aquella que llaman Web Invisible o Web Profunda (Invisible Web o Deep Web).

La Web Superficial es aquella que consiste en la cantidad de páginas que son encontradas por los motores de búsquedas globales ya que estas se encuentran dentro de una colección de información interrelacionada unas con otras; frecuentemente esta se visualiza como una gran telaraña web; esto es producido debido a los hipervínculos que contiene cada una de las páginas, los cuales referencian a otras páginas y a su vez las otras páginas referencian a otras, incluso a quienes las referenciaron. Los buscadores globales, como Google, Yahoo y Bing, debido a sus algoritmos de búsqueda, dedica vital importancia a este conjunto de hiperenlaces, colocándole una medida de popularidad; frecuentemente páginas con contenido importante y de calidad no poseen páginas que las enlazan o estas no poseen enlaces a otras páginas catalogadas como populares, en consecuencia estas son poco valoradas por los motores de búsqueda y por ende no están dentro de sus parámetros de búsqueda.

Esta es una de las principales causas que origina a La Web invisible, la misma constituye dentro de La Web todas aquellas páginas que escapan al índice de los motores de búsqueda generalistas. También forman parte de esta web algunas páginas generadas dinámicamente (p.e. como respuesta a una consulta) o aquellas páginas que forman parte de intranets o de sitios web que requieren acceso mediante contraseñas.

Álvarez, M. (2007), comenta sobre La Web Invisible definiéndola como:

> A esta porción de La Web como 'Web Oculta' (Hidden Web) o 'Web Profunda' (Deep Web) debido a que sus contenidos no pueden accederse desde los buscadores actuales, basados en crawlers convencionales. Por oposición, a la porción de La Web compuesta por páginas estáticas suele denominársele 'Web Visible' (Visible Web) o 'Web de Superficie' (Surface Web). (p. 2).

De esta forma, La Web puede, en cierto modo, verse como un iceberg (*ver Figura No. 4*). Existe una gran parte visible pero la parte que no puede verse

directamente es de un tamaño mucho mayor y de más valor; constituye la base, los 'cimientos' del iceberg; La Web Superficial es sólo la punta del iceberg.

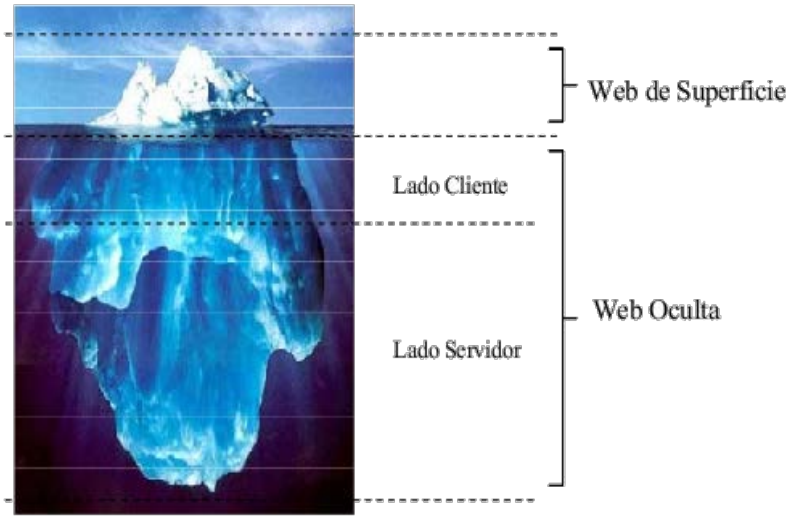

Figura No. 4. **Internet como un Iceberg.** Tomado de Álvarez P. (2008)

Se puede caer en la tentación de pensar que los buscadores más populares, cubren la mayor información posible dentro de La Web; sin embargo, si se observan sitios como PudMed, donde existe miles de documentos con información confiable relacionada a literatura biomédica y que los buscadores populares no pueden encontrar; así como este sitio existen muchos otros en Internet que contienen recursos de información importantes que no son accesibles por los buscadores tradicionales; por ello pasan a formar parte de La Web Invisible.

Chang, He, Li y Zhang (2003, citado por King, 2008, p. 11), estimaron a través de un estudio realizado, que habían alrededor de 450.000 colecciones de La Web Invisible, contenida sobre 550 billones de documentos, mientras que el más largo índice de los motores de búsqueda contenía menos de 9 billones de documentos. Según Bergman (2001) La Web Invisible contenía alrededor de 750 terabytes de datos, a diferencia de La Web Superficial que solo llegaba a 19 terabytes.

En otro estudio, Bergman (2000), quien se ocupó de caracterizar La Web Invisible, llegando a la conclusión de que contiene más del 80% de toda la

información disponible en La Web (entre 400 y 550 veces superior al contenido en la 'Web Superficial'). Además, el contenido de calidad que contiene es entre 1000 y 2000 veces superior (entendiendo por este concepto, información altamente específica y elaborada). A pesar de su alto grado de calidad, el 95% de la información de La Web Invisible está accesible al público en general sin coste alguno.

Estudios más recientes realizados por Chang, He, Li y Zhang (2004, citado por Álvarez M., 2007); corroboran las principales conclusiones del estudio anterior y constatan además el rápido crecimiento del volumen de La Web Invisible: en el lapso que separa ambos estudios (4 años), el tamaño de la misma ha crecido entre 3 y 7 veces.

Estrategias de Recopilación

Pueden distinguirse dos tipos de estrategias de recopilación de información de La Web: recopilación de información global y recopilación de información dirigida. Los buscadores convencionales como Google, Altavista, Yahoo entre otros, utilizan el enfoque global, porque pretenden abarcar el contenido completo de La Web. Por lo tanto, en sus procesos de rastreo o crawling los aspectos de escalabilidad son cruciales para construir un sistema exitoso. Para ello es clave que el componente de crawling realice operaciones de escasa complejidad. Incluso en ese caso, los recursos computacionales necesarios en términos de ancho de banda, espacio de almacenamiento y capacidad de procesamiento son increíbles.

Por el contrario, las tareas de recopilación de información dirigida están orientadas a un propósito específico; donde el volumen de información a explorar y recopilar es mucho más restringido por lo que no son necesarios los recursos computacionales precisos para abarcar toda La Web.

Las técnicas de búsquedas dirigidas, utilizan algoritmos de clasificación de contenidos para guiar los procesos de rastreo hacia contenidos relevantes para un dominio de aplicación específico.

Repositorios Digitales

Los repositorios digitales es uno de los resultados que emergieron, para cubrir las necesidades de los usuarios de buscar información en los sitios web, desplazando así a las bibliotecas tradicionales; esto debido al auge de la internet y a que cada vez su acceso es más fácil y más amplio; es por ello que hoy en día surgen como la principal fuente de acceso a la información, generalmente cobijados en instituciones como universidades, bibliotecas académicas, archivos nacionales, academias, museos, y otras entidades vinculadas a la investigación científica y/o al cuidado del patrimonio histórico y cultural. A pesar de que la internet tiene más de cincuenta años, no es sino desde hace aproximado diez años que se vienen usando recursos electrónicos como método de información.

Algunos piensan que los repositorios digitales son sitios activos, dinámicos y colaborativos, como un sitio de proyecto de software de fuente abierta. Otros, tienen una inclinación distinta, más histórica, viéndolo como lugares donde el contenido digitalizado físico y análogo y el contenido digital original son preservados para siempre. Una definición más adecuada de lo que son los repositorios digitales la dan Jantz y Giarlo (2005), “es simplemente un lugar para almacenar, acceder y conservar objetos digitales. Los objetos digitales pueden ser bastante complejos ya que reflejan la estructura de un artefacto físico e incluyen el contenido de varias secuencias de bytes y un software especial utilizado para obtener resultados dinámicos para el usuario.” (p. 4).

Un repositorio digital debe permitir almacenar todo tipo de objetos digitales, junto con los metadatos que los describen. Un objeto digital puede ser un artículo de una revista electrónica, una imagen, datos numéricos, un vídeo digital, o un libro completo en distintos formatos.

Para poder asegurar al usuario algún tipo de seguridad sobre la información a consultar; RLG (2002) establece que, el repositorio debe proporcionar información sobre sus políticas y tecnologías, y estar abierto a compartir esta información.

Además, debe ser capaz de preservar los documentos digitales por un tiempo considerablemente largo, usando métodos de conservación y actualización

Preservación Digital

El proceso de preservación digital se está convirtiendo en un proceso confiable, sin embargo, aun existe mucho escepticismo, en relación con la viabilidad e incluso el significado de este proceso. Dada la naturaleza de las tecnologías de almacenamiento electrónico y la naturaleza cambiante de las páginas web, muchos dudan de que la preservación digital se convierta en una realidad.

Aún, en tiempos presente la preservación digital crea confusión, ya que los lectores, familiarizados con los enfoques tradicionales, asocian el proceso de conservación con el uso de técnicas para evitar que el documento u objeto original se deteriore, incluso mejorarlo hasta el punto donde se pueda volver a utilizar. Ahora, en concordancia con Jantz y Giarlo (2005) "la preservación digital consiste en métodos, habilidades y resultados muy diferentes y pueden complementar los servicios tradicionales de preservación, al mismo tiempo proporcionan únicos y nuevos usos de la información dinámica" (p. 2).

Es por ello que el informe de RLG (2002, citado en Jantz y Giarlo, 2005, p. 2) define la preservación digital como: "las actividades de gestión necesarias para: 1) Para el mantenimiento a largo plazo de un flujo de bytes (incluyendo metadatos) suficiente para reproducir una adecuada representación del documento original y 2) Para permitir el acceso constante de los contenidos de los documentos a través del tiempo y el cambio tecnológico".

A raíz del problema de encontrar información confiable dentro de La Web, y con este auge por parte de muchos organismos e instituciones de crear repositorios digitales y de publicarla mediante un sitio web; surgió la necesidad, de poder considerar que dicha información allí contenida fuere confiable; por tal motivo, el proceso de preservación es un término muy importante para el desarrollo de esta

nueva definición de repositorios digitales confiables, el cual requerirá de la integración de nuevos métodos, política, estándares y tecnología

Repositorios Digitales Confiables

Debido a lo subjetivo que podría ser determinar si un repositorio es confiable o no; muchos institutos y programas se han dado a la tarea de crear métodos para alcanzar dicho objetivo. En el 2000, el programa RLG fue anexado como una unidad de la OCLC, cuyo principal objetivo ha sido la creación de métodos y métricas para poder medir la confiabilidad de los repositorios digitales contenidos en La Web; desde el 2003 junto al US National Archives and Records Administration (NARA) aunaron esfuerzos para poder establecer mecanismos de certificación de repositorios confiables; y en el 2005 publicaron la documentación requerida para tal fin; en el mismo propiciaron una metodología para auditar los repositorios, creando una lista de chequeo. Con ello podrían determinar la solidez y sustentabilidad de los repositorios, además de su confiabilidad.

Hace pocos años atrás la creación de Repositorios Confiables parecía una utopía pero hoy, es una realidad. En el informe de RLG, toma como punto de apoyo importante los procesos de preservación digital, debido a que proporcionan un punto de partida y un marco para poder establecer el concepto de Repositorio Digital de Confianza, “el cual se basa en dos requisitos principales: 1) el repositorio con las políticas asociadas, estándares, e infraestructura de tecnología proporcionará un marco de trabajo para realizar la conservación digital, y 2) el repositorio debe ser un sistema de confianza , es decir, un sistema de software y hardware que pueda ser confiable para seguir ciertas reglas” (RLG, 2002, citado en Jantz y Giarlo, 2005, p. 2).

En el mismo informe, RLG, define a los Repositorios Digitales Confiables como:

> Un repositorio digital de confianza es uno cuya misión es proporcionar un acceso confiable a largo plazo de gestión de recursos digitales para la

> comunidad, ahora y en el futuro. Los repositorios confiables pueden adoptar diferentes formas: algunas instituciones pueden optar por crear repositorios locales mientras que otros pueden optar por gestionar los aspectos lógico e intelectual de un depósito contratando un proveedor externo para su almacenamiento y mantenimiento. (p. 2).

Sea cual sea la tecnología, métodos o formas de un repositorio digital confiable, en resumen RLG propone las siguientes características que deben poseer dichos repositorios:

- Acepta la responsabilidad por el mantenimiento a largo plazo de los recursos digitales en nombre de sus depositantes y para el beneficio de los usuarios actuales y futuros.
- Diseña sus sistemas de conformidad con convenciones y normas comúnmente aceptadas para garantizar la gestión, acceso y seguridad de los materiales depositados en su interior.
- Establece metodologías para la evaluación de los sistemas que satisfagan las expectativas de la comunidad.
- Debe ser dependiente para cargar con las responsabilidades a largo plazo de los depositantes y usuarios de manera abierta y explícita.
- Las políticas, prácticas y resultados pueden ser auditados y medidos.

Motores de Búsqueda

Los motores de búsqueda, son aquellos utilizados para localizar y consultar la información en La Web. Un concepto dado por Pedraza y Codina (2009, *Sección* Que es Un Buscador, Párr. 1) “son sistemas de información documental que permiten realizar consultas y recuperar información contenida en La Web”. Dado a esa importancia que tienen como objetivo de recuperar información de La Web y al crecimiento de esta última, estas herramientas, han emergido como un método rápido y eficiente para encontrar información electrónica; información sobre cualquier tema, puede ser encontrada rápidamente. Como ese proceso de generar información y

contenido en La Internet, sigue en crecimiento, cada vez este tipo de herramientas serán más importantes para la vida diaria de los usuarios.

Existen dos formas básicas de obtener información en La Web, a través de la navegación por directorios y por medio de una selección de los Motores de Búsqueda. Los directorios son índices creados y mantenidos mediante esfuerzo intelectual; en cambio, los buscadores usan métodos automáticos mediante programas informáticos; es por eso que los resultados de ambos métodos son distintos, mientras en un directorio solo se encuentran sitios web; mediante los motores de búsqueda se encuentran además de sitios web, resultan documentos en cualquier formato.

La mayoría de los motores de búsqueda son basados en "índice invertido"; que es una estructura de índice que almacena una relación desde las palabras a sus localizaciones en un documento o conjunto de documentos, lo que permite una búsqueda total de texto. Debido a su eficiencia es la estructura de datos más popular usada en sistemas de recuperación de documentos.

Historia de los Buscadores

La presente información histórica se realizó por medio de una síntesis y resumen de información publicada por Wall (2003), Sullivan (2005) e Isopixel (2010).

Los buscadores WEB, han sido desarrollados casi desde la aparición de la Internet, y es que con la gran necesidad de los usuario de encontrar recursos, los buscadores han ido desarrollando diferentes métodos. Archie, la primera herramienta usada como buscador fue creado en 1990 y consistía en una lista descargable que contenía la información de sitios en internet vía FTP; más tarde en 1991 nace Gopher, muy parecido a Archie, solo que manejaba indexación, característica que le permitía a programas de búsqueda como Verónica y Jughead encontrar sitios web a través de la introducción de palabras claves, sin embargo su información debía mantenerse manualmente; por lo que aún en 1993 había carencia de un verdadero motor de búsqueda. Ya en esta época surgen casi a la par los dos principales métodos de búsqueda; aquellos basados en Crawlers y los basados en directorios.

A pesar de que los Crawlers fueron los primeros desarrollados, no obtuvieron importancia, ya que computacionalmente era muy costosos; al contrario de los directorios, que es una tecnología barata y no requiere mucho recurso informático, en cambio requiere más soporte humano y mantenimiento.

Con el desarrollo de súper servidores, uso de nano tecnología y aumento en la capacidad de procesamiento y almacenamiento, los Crawlers o Arañas Web han llegado a convertirse en parte esencial de los métodos de búsqueda preferido por los grandes buscadores como Google, Yahoo, Bing, etc.; estos a pesar de no ser un nuevo concepto en el ámbito de la informática, no fue sino hasta 1993 donde Matthew Gray crea un Crawler llamado Wanderer con el propósito de explorar toda La Web y crear un índice llamado Wandex. Más tarde a finales de 1993, Jonathon Fletcher desarrolló el primer buscador que combina los tres elementos esenciales (rastreo o crawling, indexación y búsqueda), llamado JumpStation; debido a las limitantes de la plataforma donde fue implantado, solo indexaba los títulos y encabezados de las páginas.

El primer buscador basado en Crawlers y que podía buscar una palabra en el texto entero de una página Web, fue creado en 1994 por la Universidad de Whashington por Brian Pinkerton, que se llamó WebCrawler y se convirtió en un estándar para los buscadores actuales. Casi a la par fue lanzado Lycos, un buscador con las mismas características que WebCrawler, solo que este fue el primero en tener gran popularidad, debido al gran esfuerzo comercial que hizo la compañía. Luego de esto comenzaron a emerger una serie de Crawlers que tuvieron una gran popularidad tales como: Excite, Infoseek, Inktomi, Nothern Light y Altavista, sin embargo, Yahoo a pesar de ser un buscador de directorios, gozaba de la mayor popularidad de los buscadores, donde los usuarios preferían realizar sus búsquedas navegando a través de los directorios.

No fue sino hasta finales de los noventa que los motores de búsqueda basados en Crawlers, adquirieron una notoria popularidad, desplazando así a los directorios, fueron conocidos como “la estrella más brillante” en La Web. Las compañías comenzaron una competencia en este mercado generando así lo que se llamo “The

Search Engine Size War" o en español "La Guerra del Tamaño de los Motores de Búsqueda" que comenzó en 1997 y que aún continúa.

En 1997 nace el buscador Google, hasta esa fecha un buscador más de La Web; pero a mediados del 2000, Google revolucionó el área de los motores de búsqueda, teniendo los mejores resultados, esto gracia a su nuevo algoritmo PageRank. Algoritmo cuya finalidad es colocar un valor de ponderación a las páginas de acuerdo a la importancia que esta tenga dentro de La Web. Así como Google, cada uno de los motores de búsqueda poseen su propio algoritmo, a pesar de esto, ninguno da importancia a la calidad del contenido que posean las páginas que son indexadas.

Funciones de Un Motor de Búsqueda

Las principales funciones de un motor de búsqueda genérico son las siguientes:

- Acceder a sitios web, localizar y descargar documentos.
- Extraer el contenido textual (y multimedia) de los documentos descargados.
- Analizar e indexar el contenido de los documentos para construir los índices del motor.
- Realizar el análisis de enlaces de cada página y otorgar alguna medida de popularidad.
- Permitir la formulación de consultas mediante palabras clave.
- Facilitar el acceso a los resultados de una consulta ordenándolos conforme a unos criterios de relevancia.

Introducción al Rastreo o Crawling

De acuerdo a Castillo (2004), existen dos características a tomar en cuenta en La Web que generan un escenario difícil para la ejecución del rastreo o crawling; uno es el gran volumen de información y otro es la tasa de cambio; ya que todos los días

una gran cantidad de información es adicionada, actualizada o eliminada. También la mejora en la velocidad de la red no va a la par con los avances que se han hecho en cuanto a procesamiento y almacenamiento. Este amplio volumen de recursos web implica que una maquina de búsqueda puede solo abarcar una fracción de La Web.

La tasa de cambio de los recursos de La Web; trae como consecuencia que mientras un Crawler puede estar descargando la información de un sitio, probablemente se esté adicionando, actualizando o eliminando información de ese sitio. Por lo que ciertamente es imposible que un buscador represente La Web en cualquier instante en el tiempo; por ello las últimas páginas que se ha encontrado, probablemente estas en ese momento, sean precisas; pero la primera página que fue encontrada, al pasar el tiempo, tiene una alta probabilidad de haber cambiado su contenido.

Edwards, MacCurley y Tomlin (2001, citado por Castillo, 2004) señala: "Hay que tener en cuenta que el ancho de banda para llevar a cabo el rastreo no es infinito ni libre, lo cual es esencial para rastrear La Web no solo de una manera escalable pero si eficiente; solo si alguna medida de calidad razonable es mantenida"(p. 27).

El comportamiento de un rastreador web es el resultado de una combinación de políticas por lo que Castillo (2004), establece las siguientes políticas como las principales que un buscador debe elegir:

- Una política de selección para determinar qué información descargar.
- Una política, para visitar nuevamente la información obtenida para revisar sobre algún cambio.
- Una política para evitar la sobrecarga de información
- Una política para coordinar el rastreo distribuido paralelamente.

A continuación se explican en detalle cada una de estas políticas:

Políticas de Selección. Dado al tamaño de La Web, y que es imposible para un motor de búsqueda cubrirla completamente; como lo demuestra Lawrence y Giles (2000, citado por Castillo, 2004, p. 28), solo el 16% de la información de La Web

puede ser cubierta por los buscadores; por ello es conveniente que la fracción a buscar dentro de ella sean sitios con contenido de calidad y no sea un simple criterio al azar de búsqueda. Estudios realizados por Chen, Chung, Ramsey y Yang (1998), Najork y Wiener (2001) y Abiteboul, Preda y Cobena (2003); en resumen, los tres estudios concuerdan que se puede abarcar una parte importante de La Web, sabiendo tomar cuales serán los sitios iniciales o "semillas". De allí parte el termino lo que Chakrabarti, Van Der Berg y Dom (1999, citado por Castillo, 2004, p. 29), introdujeron de motores de búsqueda con Propósito Especifico o dirigidos.

Políticas de Re-Visita. Debido a la naturaleza dinámica de La Web y tomando en cuenta el tiempo que toma a un buscador en recorrer la información; cuando este termina de rastrear, muchas modificaciones han ocurrido. Por esta razón, desde el punto de vista de un motor de búsqueda, existe un costo asociado al no poder cubrir los eventos que ocurren, lo que implica no tener la información actualizada.

He aquí dos variables que se toman en consideración como son "Freshness" y "Age"; donde "Freshness" es la medida que determina si la información es precisa o no; y "Age" indica cuan anticuada es una copia del repositorio.

La evolución de estas dos variables se observan en la *Figura No. 5*.

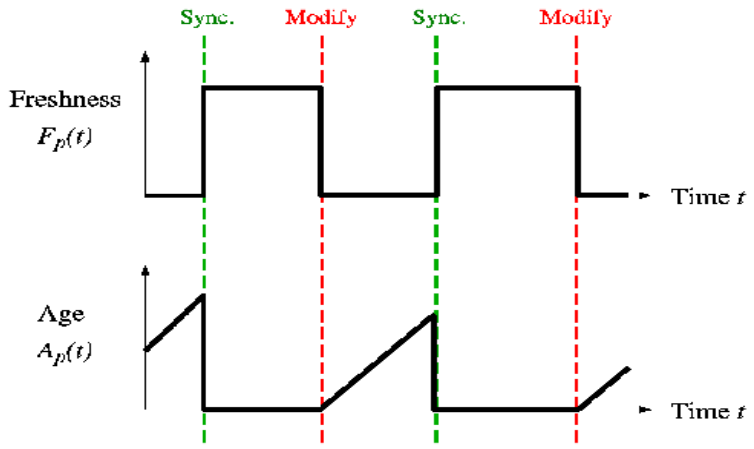

Figura No. 5. **Evolución de Variables Age y Freshness.** Tomado de Castillo (2004, p. 31).

El objetivo de un motor de búsqueda es mantener el promedio de "Freshness" tan alta como sea posible y mantener "Age" tan baja como sea posible. Para lograr este objetivo Cho y Garcia (2003, citado por Castillo, 2004, p. 32), estudiaron dos políticas simple de revisita:

1. Política de Uniformidad, la cual involucra revisitar todas las páginas de la colección con la misma frecuencia, sin tomar en cuenta su tasa de cambio.
2. Política Proporcional, que consiste en revisitar con más frecuencia las páginas que más cambian.

En el estudio encontraron un sorpresivo resultado, haciendo pruebas tanto en simulación como en el mundo real; donde la política de uniformidad generó mejores resultados midiendo el promedio de páginas actualizadas. Debido a que cuando una página cambia con demasiada frecuencia, el motor pierde mucho tiempo tratando de rastrear nuevamente la página que cambió y aun así no tendrá la información actualizada de dicha página; por lo que concluyeron "para mejorar la Actualización, se deben penalizar los elementos que cambian con demasiada frecuencia" (p. 32).

Por consiguiente, para mantener un óptimo promedio de "Freshness se debe ignorar las páginas cuyos cambios son demasiados frecuentes. Ahora para mantener un óptimo promedio bajo de "Age" se debe accesar a la información de manera frecuente que monótonamente aumenta con la tasa de cambio de cada página. En ambos casos lo más óptimo es más parecido a la Política de Uniformidad; tal como concluye Coffman y Liu (1998, citado por Castillo, 2004, p. 32) "con el fin de minimizar el tiempo de obsolescencia, los accesos a cualquier página en particular deben ser lo más iguales posible".

Políticas de Sobrecargar. El uso de los buscadores en La Web trae consigo un costo computacional; Koster (1995, citado por Castillo, 2004, p. 32), presentó sus resultados de acuerdo al uso de los motores en La Web, donde comprobó que traían consigo un alto consumo de recursos de red, necesitando un ancho de banda considerable para poder cumplir con su función eficazmente; sobrecarga de servidores, sobre todo si la frecuencia de acceso al servidor es muy alta; también encontró buscadores mal desarrollados que podían colapsar un servidor, estos

conocidos como “spams” o “troyanos”.

Aunque estos estudios se realizaron en 1995 y a pesar que se han realizado muchos avances como el nacimiento de la Internet 2 y aumento de las capacidades de los componentes de red; igualmente hay que tomar en consideración dichos resultados; ya que de igual manera aumenta la información y tamaño de La Web.

A partir de los problemas encontrado, Koster; propone dos soluciones; donde primero sugiere la creación de un protocolo de exclusión para los buscadores, para que los administradores de servidores puedan establecer a que recursos podría buscar información; y segundo, considera que el tiempo de visita a un servidor debería ser por lo menos 60 segundos, de intervalo entre uno y otro.

Políticas de Rastreo Paralelo. Un buscador paralelo consiste en aquel que puede ejecutar múltiples procesos al mismo tiempo; con el objetivo de maximizar la cantidad de información que pueda obtener, minimizando la sobrecarga de recursos computacionales a través de la paralelización y evitar la redundancia de información.

Para evitar la redundancia, es decir, evitar el problema que dos procesos del Crawler descarguen los mismos recursos al mismo tiempo; Cho y Garcia, proponen dos métodos: asignación dinámica y asignación estática.

Castillo concluyen que:

> Sea cual fuere el método a seleccionar hay que tener en cuenta tres aspectos; en primer lugar cada proceso del Crawler debe alcanzar la misma cantidad de sitios; en segundo lugar, si el numero de procesos crece, el numero de sitios asignados a cada proceso debe reducirse; y por último, el método de asignación debe ser capaz de añadir y eliminar procesos dinámicamente. (p. 34).

Arquitectura de un Motor de Búsqueda

Un motor de búsqueda consta de un conjunto de programas que trabajan cooperativamente. En la *Figura No. 6* se muestra el esquema básico de una arquitectura de búsqueda basada en Crawlers; se distinguen los tres principales componentes: crawling (rastreo), sistema de recuperación de información o

indexación y la interfaz consulta; cada uno de ellos se detallan a continuación:

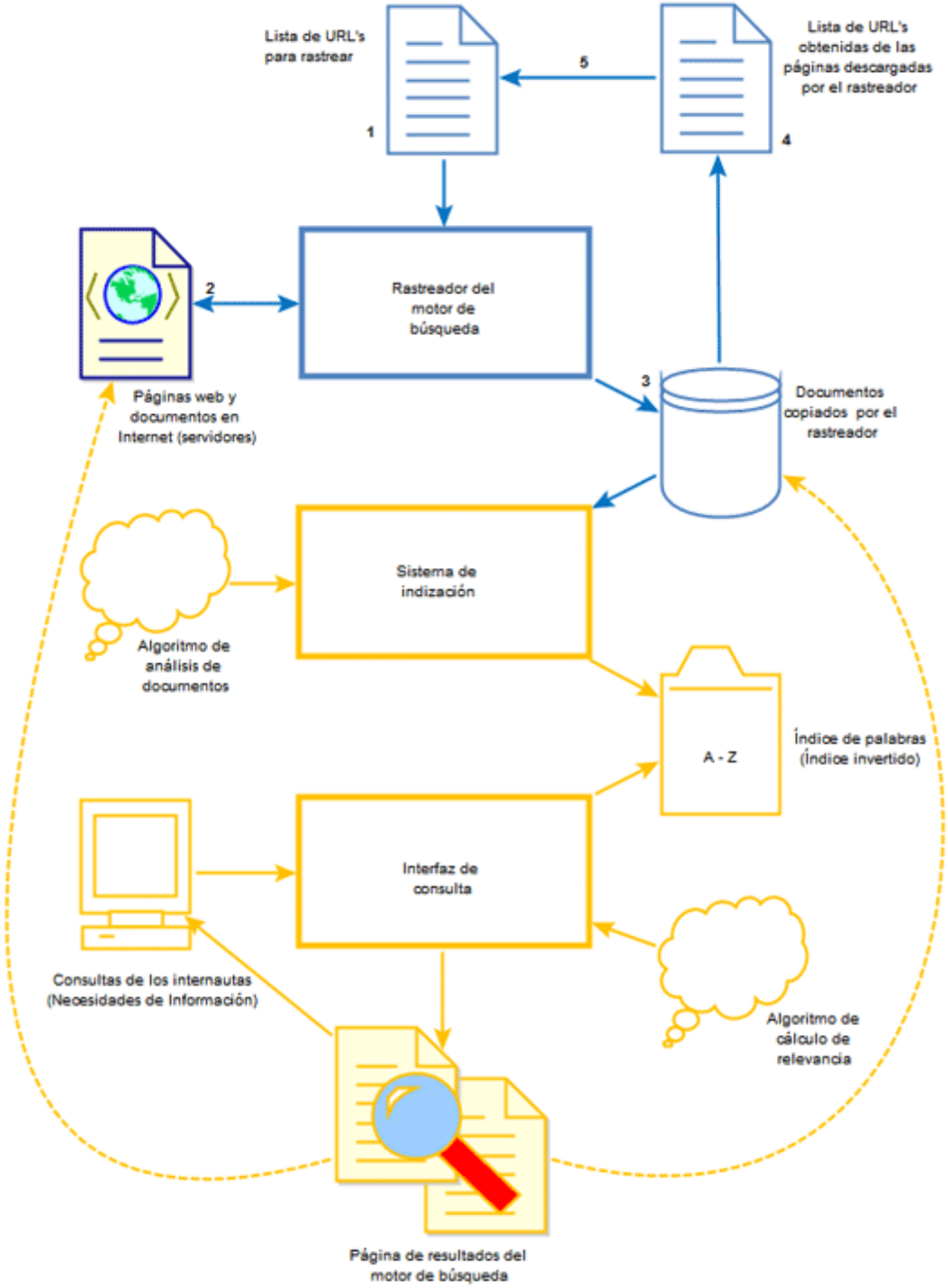

Figura No. 6. **Arquitectura General de un Motor de Búsqueda.** Tomado de Codina (2008, p. F092-3).

Rastreador del Motor de Búsqueda o Crawler. Es el principal componente, tal vez el más importante, y el que caracteriza a los motores de búsquedas, también llamado Spider, es un pequeño software, que recorre los enlaces que existen entre las páginas Web de Internet de forma automática y sistemática.

Los Crawlers se utilizan mucho hoy en día. Su principal uso es en motores de búsqueda, pero éste no es el único uso para el que se pueden usar; pueden utilizarse para multitud de tareas. Por ejemplo, se podría crear un Crawler que invite a todos los miembros de una comunidad o se podría crear uno que garantice que todos los enlaces de un sitio web apunten a páginas activas.

Montero (2009) define a Los Crawlers como "un tipo especializado de webbot - robot de La Web - que se encarga de llevar a cabo un tipo concreto de tareas. En particular, se encarga de recorrer las páginas Web de Internet, descargarlas al ordenador local, parsearlas y procesarlas" (párr. 2).

El Crawler explora La Web en busca de información contenida en los documentos, se obtiene la ubicación y las páginas a las que se hace referencia. La ubicación de un documento web se define por su dirección en Internet conocida como Uniform Resource Locator (URL).

Entonces el modulo del Crawler rastrea todos los enlaces que existen entre las páginas web, tomando información de las mismas de su HTML; este modulo recibe como entrada un conjunto de URLs o direcciones Web, también llamadas "semillas", que apuntan a documentos que hay que descargar de La Web. Utilizando como base para su funcionamiento la estructura de hiperenlaces existente entre los diferentes documentos, extrae los URLs que aparecen en las páginas recuperadas y envía esa información al módulo de control del Crawler; recursivamente continua buscando a través de los enlaces que contienen a otras páginas; esta relación de hiperenlaces entre las páginas Web se puede ver en la *Figura No. 7* en forma de grafos. El objetivo es que, a través de Crawler, indexar La Web y centralizar la información en un índice para luego almacenar las páginas recuperadas en un repositorio.

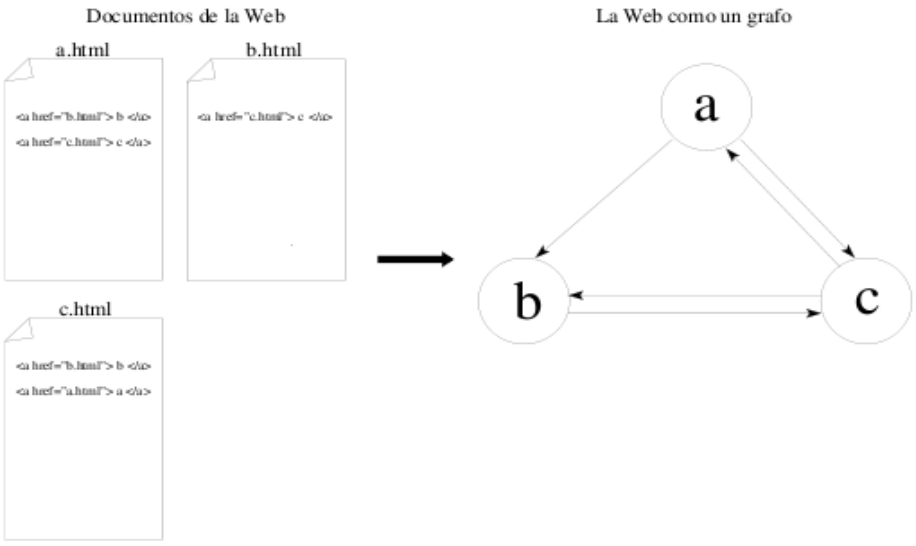

Figura No. 7. **Representación de La Web como un Grafo.** Tomado de Álvarez P. (2008, p. 9).

El módulo de control del Crawler determina qué enlaces visitar y el orden de los mismos, y alimenta con ellos al módulo de Crawlers; el cual es responsable de gestionar la dirección de exploración del Crawler y establecer un criterio de parada. El algoritmo básico de crawling puede admitir diversas configuraciones; por ejemplo, en función de la amplitud que se le quiera dar al crawling (es decir, el porcentaje de La Web a abordar) o de alguna temática en particular, además puede configurarse para visitar tantos sitios como sea posible, obviando sitios que se encuentren a niveles más profundos de los sitios analizados; o puede configurarse para analizar sólo sitios de un dominio.

Cada buscador posee su propia implementación de Crawler; cada cual con una estrategia distinta. Por ejemplo algunos Crawlers, desestiman aquellas páginas cuya información es generada por una base de datos, accesada a través de lenguajes dinámicos como PHP o ASP, dejando así una cantidad de información importante sin indexar. También existen Crawlers donde se puede configurar que tipo de información va a rastrear si solo contenido HTML, información contenida en PDF, Word o solo imágenes.

Se definen dos tipos de sistema de Crawler, dependiendo del propósito de

alcance del motor de búsqueda; aquellos de propósito general y los de propósito especifico. En la *Figura No. 8* se puede observar ver su subdivisión.

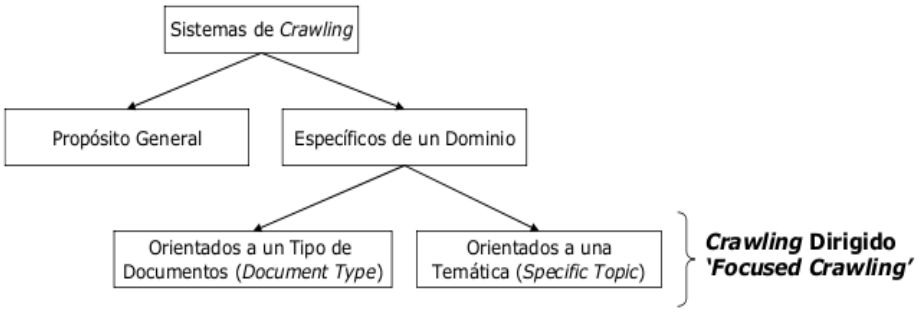

Figura No. 8. **Clasificación de los Sistemas de Crawling.** Tomado de Álvarez M. (2007, p. 19).

1. Propósito General: En esta categoría se incluye aquellos que recopilan documentos sin considerar algún tópico de búsqueda. Tienen como objetivo principal localizar y registrar el mayor número posible de páginas en La Web. Estas páginas pueden abordar una gran variedad de temas. Chau y Chen (2003, citado por Álvarez, 200, p. 17) consideran que los motores búsqueda que utilizan Crawlers de propósito general no satisfacen todas las necesidades de los usuarios al buscar información más específica. Los motores de búsqueda como Google, AltaVista o Yahoo son de propósito general y es por eso que usualmente muestran miles de páginas como resultado de las consultas, muchas de estas páginas son irrelevantes para el usuario, esto según Chau y Chen.

2. Propósito Especifico: Chakrabarti, Van Der Berg y Dom, acuñaron el nombre de crawler especifico (focused crawler); según Álvarez (2007) “el cual localiza y obtiene páginas relacionadas con un conjunto de temáticas determinadas, que representan segmentos relativamente limitados de La Web” (p. 20).

A diferencia de los de propósito general, llevan una orientación en el proceso de recopilación, y sólo las páginas relacionadas con un dominio previamente definido son consideradas en el proceso de exploración de La Web. Durante el proceso, un “clasificador” determina qué enlaces seguir para cada página obtenida, en función de

su relevancia potencial para la temática objetivo. En última instancia, el clasificador es el encargado de decidir la dirección de exploración del sistema de crawling. Para ello el proceso de crawling se inicia con unas semillas o páginas, y el clasificador no sale de los dominios especificados.

Es importante acotar que debido al enorme tamaño de La Web, los sistemas de propósito general no son capaces de satisfacer las necesidades de acceso a información de todos los usuarios; sin embargo también es cierto que no es necesario que un sistema de búsqueda recorra toda La Web para ser efectivo.

El recorrido en cuanto a extensión, permite al sistema de propósito específico obtener como mínimo, las páginas de inicio de los sitios web enlazados por las páginas exploradas. A pesar de su simplicidad, se utiliza de forma extensiva en sistemas especializados, debido a que es fácil de implementar y rápido en ejecución. Intuitivamente si un URL es relevante para un dominio destino, es probable que las páginas en su vecindad también lo sean. Esta idea está basada en los resultados obtenidos por Kumar, Raghavan, Rajagopalan y Tomkins (2002, citado por Álvarez, 2007), donde comparan La Web con una sociedad; en dicho modelo, las páginas no apuntan a otras al azar, sino que reflejan páginas interesantes o relevantes para ésta, según el autor de la misma.

Sistema de Indexación o Indexador. El módulo de indexación extrae todas las palabras de cada página y le asocia el URL en el que cada palabra ha aparecido. El resultado es una tabla de búsqueda generalmente de gran tamaño, que proporciona todos los URLs que apuntan a páginas donde una palabra dada aparece. Lógicamente, el tamaño de la tabla estará limitado a las páginas obtenidas por el proceso de crawling. Debido a las dificultades inherentes en el tratamiento de volúmenes de información tan grandes y con frecuentes tasas de cambio, normalmente se suele hacer uso de varios índices. Por ejemplo, suele utilizarse un índice separado para almacenar la estructura de enlaces entre los diferentes documentos. El módulo de análisis de páginas es el responsable de la creación de estos índices.

El repositorio de páginas representa la colección (posiblemente temporal) de páginas que son descargadas por el crawler. Esta colección de páginas suele

mantenerse mientras no se finaliza el proceso de crawling/indexación. En algunos casos se mantienen más tiempo, para utilizarlas a modo de caché de documentos y acelerar el acceso a los documentos contenidos en las páginas de resultados de búsqueda.

En el índice las páginas tienen una importancia dado por un peso que es otorgado por el algoritmo del motor de búsqueda. Cada uno posee su propio algoritmo y métodos para ponderar las páginas. Este es normalmente el secreto mejor guardado por los buscadores, ya que es allí donde gastan gran parte de su intelecto e inversión financiera.

Los buscadores pueden excluir documentos que pueden considerar irrelevantes o de poco valor; también el indexador puede borrar ciertas páginas como por ejemplo, aquellas con contenido ofensivo o sitios Web que pueden alterar el Índice de los motores, mejor conocidos como “spams”.

La optimización de los motores de búsqueda, en cuanto a la relevancia de los resultados, es hoy en día una industria muy lucrativa; y es que la mayoría de los sitios Web desean ser primero en los resultados de búsqueda; por ello, algunos sitios Web han optado por, de alguna manera u otra, manipular la clasificación de los resultados; creando páginas “ficticias” que tengan hipervínculos con otra que desean colocar de primero en las búsquedas o también creando un tráfico irreal en dicha página. Incluso varios buscadores tienen la opción de pago a cambio de relevancia dentro de las búsquedas.

El modulo de indexación, es una pieza fundamental para el sistema; ayudando en el rastreo ya que provee información sobre la clasificación de la páginas; así el Crawler puede ser más selectivo y tratar de buscar primero en las páginas más importantes.

Existen dos tipos de índices que se usan en el proceso de indexación; el Índice Directo y el Índice Invertido. Las consultas las resuelven los índices invertidos, mientras que los índices directos, se utilizan como elementos de gestión y control internos.

1. Índice Directo: En un índice directo se tiene la lista de documentos (o de

registros) en un orden cronológico (el documento más antiguo primero, por ejemplo) o numérico (del documento número 1 hasta el último). La *Cuadro No. 2* ilustra esta clase de índices:

Cuadro No. 2
Representación de un Índice Directo.

Id. Documento	Contenido
0001	Titulo: Manual de Planificación Tributaria y Control Interno. Autor: Osuna, Adriana.
0002	Titulo: Optimización evolutiva bicriterio. Autor: Álvarez, Pavel
.....	.....
35450	Titulo: Estudio sobre los niveles de ventas de las bombas centrifugas. Autor: Carrasco, María.

Fuente: El Investigador

Como se observa en la tabla, con este índice, para saber si hay un documento con las palabras “bicriterio” y “centrifugas”, se tendría que recorrer decenas de miles de entradas del índice (específicamente 35.450). Lo más crucial es que si el índice completo tuviera, por ejemplo, cien mil entradas, habría que recorrer las cien mil entradas del índice para saber si hay más de un documento que cumpla la condición anterior.

Es fácil suponer que esta clase de índices no mejora mucho en relación al supuesto rastreo en tiempo real de La Web. Por tanto, un motor de búsqueda necesita complementar este índice con un índice invertido, que es el que se utiliza realmente para responder a las consultas (mientras que el índice directo se utiliza para aspectos de gestión y administración internos).

2. Índice Inverso: La estructura de un índice invertido es exactamente la inversa a la de índice directo. Consiste en una lista ordenada de todas y cada una de las palabras que aparecen en los distintos documentos asociadas a los documentos

concretos en los que aparecen. La estructura básica de un índice invertido, se observa en la *Cuadro No. 3*; el cual está compuesto principalmente de un vocabulario, representada por la columna "Termino Único" y por una lista de ocurrencia, representada por la columna "Frecuencia".

Cuadro No. 3
Representación de un Índice Invertido

Termino Único	Frecuencia	Ubicación
Estudio	110	(35450,01,01)
.....	.....	.....
Bicriterio	233	(0002,02,03)
.....	.....	.....
María	214	(35450,02,02)
.....	.....	.....
Osuna	6	(0001,02,01)
.....	.....	.....

Fuente: El Investigador

De acuerdo a la *Cuadro No. 3*, en la columna "Término único" aparecen las distintas palabras de los documentos, pero solamente aparece una fila por cada palabra; aunque en el conjunto de los documentos aparezca muchas veces. En la columna "Frecuencia" se observa el número total de veces que aparece cada término. Por último, en "Ubicación" se tiene una clave en forma de vector donde aparece el número de documento, la zona o campo donde aparece la palabra y el orden de la palabra. Habrá un vector por cada ocurrencia.

Con más detalle, se puede tomar el término "Estudio"; donde se observa que la columna de "Frecuencia" señala: 110. Esto significa que "Estudio" aparece 110 veces en el conjunto de los documentos, y que habrá por tanto 110 vectores distintos en la

columna Ubicación. Esta columna de “Ubicación”, es muy importante cuando se usa para consultas complejas, como consultas para frases exactas o consultas aproximadas. En la *Figura No. 9*, se tiene un ejemplo con consultas.

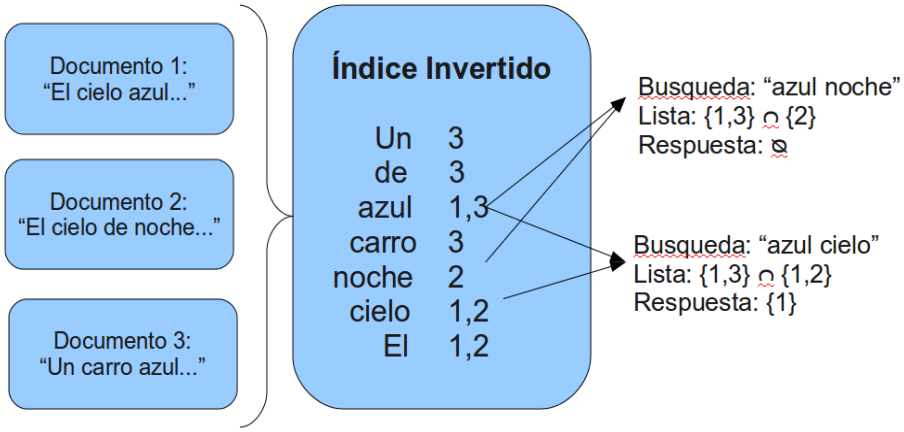

Figura No. 9. **Demostración de Índice Invertido Con Criterios de Búsqueda**.
Fuente: El Investigador

Castillo (2004) afirma que “Las búsquedas a través de este tipo de índice son muy rápidas porque normalmente se usan estructuras tipo Hash en memoria para el vocabulario y además la lista de ocurrencia son pre-ordenadas por algún criterio” (p. 19).

Otros índices auxiliares ayudan al indexador a recorrer la lista de términos únicos con un pequeño número de operaciones de comparación. Lo mejor de todo es que esta clase de índices hace que el tiempo de respuesta sea virtualmente independiente del número de términos que aparecen en la lista. El problema principal con estos índices es la cantidad de recursos de hardware que requieren, particularmente, si se debe dar servicio a miles de usuarios simultáneamente.

Castillo (2004) también, establece 4 etapas que debería tener el proceso de indexación, que se puede ver en la *Figura No. 10.*

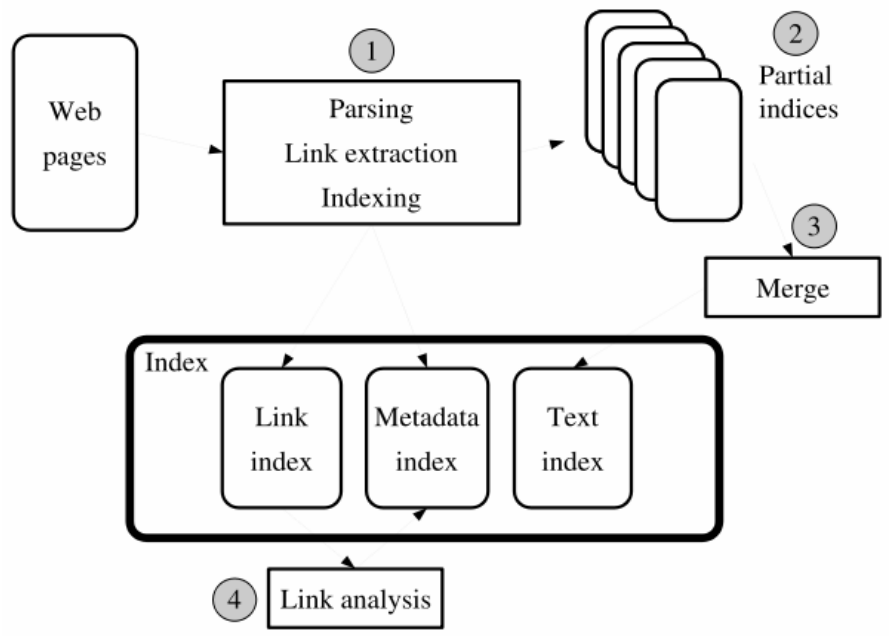

Figura No. 10. **Etapas del Proceso de Indexación.** Tomado de Castillo (2004) "(1) Las páginas son analizadas, enlazadas y extraídas (2) Índices parciales son escritos en disco cuando la principal memoria es agotada. (3) Los índices son combinados en un gran Índice (4) Los enlaces guardados en el Índice pueden ser usados para calcular las ponderaciones de importancia" (p. 20).

Un aspecto importante a considerar, es el uso de los recursos de Hardware como la memoria y el CPU; ya que todos estos índices normalmente son guardados en Base de Datos o en alguna otra estructura de almacenamiento. Por cada adición o eliminación, implica una actualización de todo el índice; especialmente en aquellos casos donde el motor de búsqueda hace que todo el texto de una página pueda ser indexado.

Interfaz de Consulta. Finalmente, una vez que los datos son indexados, el motor de búsqueda se encuentra listo para generar resultados; por lo que cada motor de búsqueda provee una interfaz para realizar dicha operación. Para cumplir con este proceso se apoya en un programa de extracción de información, el cual a través de un algoritmo, busca la información en los índices y devuelve el resultado en una interfaz de resultados.

Existen varios métodos de selección de los resultados. Cada buscador posee un

método de clasificación y ordenamiento distinto, incluso pueden aplicar filtros. Tanto la cantidad como la calidad de la información de los resultados de una búsqueda son importantes, atributos que un usuario diferencia para la elección entre un motor y otro.

En general, la interfaz de consulta sirve para enviar unos criterios, insertados por el usuario, a otra parte del sistema que compara los términos de los criterios con el índice invertido y filtra de este modo las páginas web que contienen los términos. Luego el motor de búsqueda debe presentar los resultados de la forma más clara y eficiente posible a través de la página de resultados del motor de búsqueda. Por último, pero posiblemente lo más importante de todo, los resultados deben mostrarse en algún orden significativo, y de entrada se puede descartar el orden alfabético o el cronológico, dada su escasa utilidad con la inmensa cantidad de documentos que hay en Internet. Aquí interviene el denominado algoritmo de cálculo de relevancia.

Con respecto al cálculo de relevancia; Pedraza y Codina (2009), "Su función es la ordenación de los resultados, es decir, la construcción del ranking de resultados en función de unos criterios de relevancia. Algunos de ellos son: enlaces de entrada y URL; contenido, propiedades de la página, frecuencia, ubicación, formato de las palabras clave, etc. y resultados pagados y resultados no pagados" (*Sección* Componente 3: Software de Relevancia, párr. 1).

Codina (2008), plantea como puede suceder este cálculo de relevancia.

> En recuperación de información se considera que los distintos documentos de un fondo presentan un grado de relevancia diferente para cada pregunta o necesidad de información. Ante una pregunta dada, en un extremo tendremos a los documentos cuya relevancia será cero (por no tener ninguna relación con la pregunta). En el otro extremo tendremos (con suerte) documentos cuya relevancia será total (o casi total). En zonas intermedias tendremos documentos con diversos grados de relevancia. Si la expresamos con tanto por ciento, tendremos un espectro que irá del 0% de relevancia al 90% o al 100% de relevancia con zonas intermedias donde habrá documentos con el 10%, el 25% o el 70% de relevancia, etc. (p. F092-8).

Tomando en cuenta lo anterior, lo más lógico será presentar la página de resultados con los documentos ordenados según su grado de relevancia. De este modo,

aunque haya miles y miles de documentos relevantes en la respuesta, se podrá limitar la consulta.

Codina (2008), considera que “los motores de búsqueda actuales suelen combinar dos grupos de criterios para determinar la relevancia de una página web: Criterios internos o intrínsecos y Criterios externos o de popularidad” (p. F092-8).

Donde los Criterios internos o intrínsecos, Codina (2008) los define como:

> Aquellos que se refieren a los aspectos estadísticos o de frecuencia de ocurrencia de las palabras clave de la búsqueda Entonces aquella información de La Web que tengan mayor ocurrencia de las palabras clave, serán más relevantes. Otro punto que se toma en cuenta es determinar si las palabras claves aparecen rodeadas de etiquetas como <h1> o <h2>, esta condición también otorga mayor importancia relativa a la página correspondiente, y sobre todo, al hecho de que la palabra clave forme parte de la URL de la página. (p. F092-9).

Los Criterios externos o de popularidad, como su nombre lo dice, toma en cuenta las condiciones externas de cada página y se refieren al resultado que pueda arrojar el análisis de los enlaces de entrada de la página considerada.

Cada motor de búsqueda tiene sus propios conjuntos de criterios y sus propias reglas para asignar pesos a cada criterio, pero en general, el número de enlaces que recibe una página suele ser uno de los más importantes, al menos a igualdad de los otros factores.

Por último, la página o interfaz de resultado, es la que presenta los resultados de la búsqueda; Pedraza y Codina (2009), dan algunas recomendación de que información y opciones debería mostrar.

La página de resultado debería ofrecer la siguiente información:

- Título de la página (o del documento).
- El tipo del documento (cuando no es HTML).
- Unas líneas de descripción del contenido del documento.
- URL de la página.
- Tamaño de la página web.

Y opciones:

- Obtener una versión traducida de la página con traducción automática (en general muy deficiente).
- Ver la página en la caché.
- Buscar páginas con contenidos similares.
- Navegación secuencial entre los resultados o yendo a una página de resultados concreta (hasta la página 90 más o menos).
- Restringir la siguiente búsqueda a los resultados obtenidos.

Almacenes de Datos

Los almacenes de datos o DataStores, se refieren a sistemas de almacenamiento de datos ya sean base de datos relacionales (RDBMS) o no relacionales (NoSQL). En un motor de búsqueda existe la necesidad de guardar la información encontrada; por lo que es necesario seleccionar algún almacén de datos para guardar la estructura de la información, ergo el almacén se debe ajustar a la estructura.

Un almacén de datos debe garantizar una redundancia mínima de los datos; acceso concurrente por parte de múltiples usuarios; optimización de consultas; seguridad de acceso; respaldo y acceso desde lenguajes de programación.

El propósito general de los almacenes de datos es el de manejar de manera clara, sencilla y ordenada los datos que posteriormente se convertirán en información importante para el usuario final.

En la actualidad existen dos principales enfoques de almacenes de datos los relacionales y los no relacionales. Las Base de Datos Relacionales surgieron a partir de las bases postuladas en 1970 por Edgar Codd en su Modelo Relacional; estos almacenes de datos se componen principalmente de tablas y relaciones; cada tabla posee registros y campos, además de restricciones que garantizan la integridad de los datos.

Más tarde, y originalmente motivado por la oleada de aplicaciones de La Web 2.0, donde las mismas son diseñadas orientadas a alcanzar millones de usuarios, en contraste de las aplicaciones que usan el modelo relacional. Comenzó una escalada de

creación de almacenes de datos no relacionales o comúnmente llamas NoSQL. Este nuevo paradigma fue diseñado principalmente para proveer escalabilidad horizontal para las operaciones de lectura/escritura de una base de datos, distribuida por distintos servidores. A diferencia de las relacionales, que no proveen la habilidad de escalar horizontalmente.

En concordancia con Cattell (2011, p. 1), los sistemas NoSQL, persiguen principalmente las siguientes seis características:

- Habilidad de escalar horizontalmente, a través de muchos servidores.
- Habilidad de replicar y distribuir los datos a través de muchos servidores.
- Una sencilla interfaz de llamado o protocolo.
- Un modelo más débil de concurrencia de transacciones ACID que el de las base de datos relacionales.
- Uso eficiente de índices distribuidos y memoria RAM para el almacén de datos.
- Habilidad de adicionar nuevos atributos a la información guardada dinámicamente.

NoSQL

En 1998, surge el término NoSQL, para referirse al nuevo paradigma presentado por algunos programadores de Base de Datos distribuidas de código abierto, no relacionales y que no usan el lenguaje de consulta SQL (Structured Query Lenguaje). Según Requena (2010) dice de NoSQL lo siguiente: “con el término NoSQL nos referimos a una multitud de bases de datos que intentan solventar las limitaciones que el modelo relacional encuentra en entornos de almacenamiento masivo de datos, y concretamente al momento de escalar, donde es necesario disponer de servidores muy potentes y de balanceo de carga” (párr. 8). Y es que las base de datos NoSQL al no poseer relaciones hace que las búsquedas y tiempo de repuesta sean mucho más rápido; como por ejemplo, “Cassandra (una Base de Datos NoSQL)

es capaz de escribir en disco 50GB de datos en tan sólo 0.12 milisegundos, 2500 veces más rápido que MySQL (Manejador de Base de Datos Relacional)" (Lakshman y Malik, 2009, p. 21).

Google tomando en cuenta este nuevo paradigma; en el 2004 comienza el desarrollo de BigTable, una Base de Datos NoSQL desarrollada en su propia compañía, que es capaz de almacenar grandes cantidades de datos, además dispone de un buena manejo de balanceo de carga.

> Google decidió crear Bigtable porque los sistemas de bases de datos tradicionales no le permitían crear sistemas lo suficientemente grandes. Las bases de datos relacionales, como pueden ser MySQL, PostgreSQL, Firebird u Oracle se diseñaron pensando que se ejecutarían en una solo servidor con mucha potencia. Jamás se pensó en la posibilidad de que estuviesen distribuidas en miles de servidores. (Ruiz, 2009, p 48).

En el 2010, Google crea Caffeine como nueva infraestructura de búsqueda basada en BigTable, Metz (2010) señala que Caffeine hace una especie de modelo de programación de base de datos que permite realizar cambios en su índice sin volver a generarlo desde el principio. Antes de Caffeine el índice era construido nuevamente desde el principio usando MapReduce, que consistía en un conjunto de procesos de operación en lote, este recibía una cantidad enorme de datos recogidos por el Crawler de Google para actualizarlos en la Base de Datos; también debía determinar nuevamente la ponderación de cada sitio a través de PageRank. Elsar Lipkovitz, Director de Google, en una entrevista realizada en 2010 (citado por Metz, 2010, párr. 7) dijo lo siguiente: "Nosotros debíamos comenzar con esta larga colección de datos y procesarlos; luego de ocho horas o más, tomábamos las salidas de este largo proceso y copiábamos estos a nuestros servidores. Y esto lo hacíamos continuamente".

En esa misma entrevista Lipkovitz nos habla sobre los beneficios que ahora tienen usando Caffeine y BigTable:

> Este es completamente incremental Cuando una nueva página es rastreada, Google puede actualizar su índice con los cambios necesarios en vez de reconstruir todo. Antes, para actualizar los índices viejos, nosotros analizábamos toda La Web, lo que significaba que había una espera

> significante cuando encontrábamos una página y la hacíamos disponible para ustedes. En cambio, con Caffeine, nosotros analizamos La Web en pequeñas proporciones y actualizamos nuestro índice sobre una base continua, globalmente. Cuando vamos encontrando nuevas páginas, o nueva información existente en las páginas, nosotros podemos agregarlos directamente al índice. Esto significa que ustedes pueden encontrar la información más fresca que nunca antes, sin importar cuando o donde esta fue publicada. (párr. 11).

El Dr. Max Glaser (2010) en su blog, manifiesta que: "Sin duda la gran ventaja tecnológica, proporcionada por Caffeine, que actualmente tiene Google sobre sus competidores ha ayudado de gran manera al desarrollo de la nueva interfaz del buscador" (párr. 1). Esto refiriéndose a Google Instant, que fue exhibido en septiembre de 2010, en donde el buscador procesa una gran cantidad de peticiones que generan los navegadores a medida que los usuarios teclean sus consultas.

Categorías

Uno de los principales problemas de un sistema de búsqueda es el almacén de datos. Alrededor de cien millones de dominio y un alto número de procesos lectura/escritura debe ser soportado por el almacén de datos. Acá se verán las principales tecnologías para almacenar los datos persistentes.

Cuando se trata de almacenar datos de forma persistente se pueden distinguir cinco principales categorías de almacenes de datos. Todas ellas ofrecen una manera de hacer la información consistente a través de una aplicación y de hacerla accesible a través de la red.

Sistemas de Base de Datos Relacional. Las Base de Datos Relacionales, parten de la idea que los datos pueden ser representados en una colección de tablas relacionadas entre sí, en donde almacenan sus datos en filas y columnas llamadas tuplas.

La mayoría de ellas se enfoca en ser capaz de proporcionar datos consistentes, la capacidad de hacer cumplir específicas restricciones sobre los tipos de datos, las restricciones sobre relaciones y la capacidad de hacer consultas. Para grandes

conjuntos de datos, estas suelen usar indexación con una estructura de datos llamada b-tree.

El lenguaje SQL (Structured Query Lenguaje); es usado por las Base de Datos Relacionales, como interfaz de los desarrolladores para poder extraer datos desde el almacén a través de consultas. Las transacciones y consistencia de datos son características que son bien manejadas por las Base de Datos Relacionales; pero tienen un impacto negativo en el rendimiento de las operaciones de lectura/escritura

Almacenes Basados en Columna (Column Store). Se refiere a los almacenes de datos cuyos datos son guardados en columnas; he aquí una principal diferencia con el paradigma tradicional relacional que guarda los datos en filas; lo que significa que los datos guardados no necesitan de un esquema en común. Otra diferencia es que las Column Store no se enfocan en las propiedades ACID o restricciones entre los diferentes conjunto de datos.

Los proyectos que se basan en este tipo más populares son Hadoop/Hbase, Cassandra e Hypertable.

Almacenes Documentales (Document Stores). "Los sistemas que se encuentran en esta categoría, son aquellos que en realidad tienen conocimiento de los datos que almacenan. Estos además de proveer las operaciones básicas, también permiten operaciones map/reduce. Estos sistemas, al contrario de los Column Stores pueden hacer uso de índices secundarios" (Seeger, 2010, p. 25). Los proyectos más populares de este ámbito son CouchDB y MongoDB; esta última además utiliza una estructura avanzada de Hash y Arrays.

Almacenes Clave-Valor (Key-Value Stores). Estos almacenes usan un modelo de datos sencillo que consiste en un índice clave-valor. Cattel (2011) amplia este concepto: "Generalmente proveen un mecanismo para persistencia y funciones adicionales como: replicación, versiones, bloqueo, transacciones, ordenación, entre otras; la persistencia es provista a través de arreglos asociativos y la interfaz del cliente provee inserciones, borrados y una búsqueda de índices" (p. 5).

En vista que los almacenes Key-value no ofrecen características complejas como filtros de búsqueda, joins, claves foráneas, ordenación o triggers; su

rendimiento es predecible y se basa en función de la cantidad de keys almacenados. Estos como tampoco ofrecen relaciones entre las keys, hacen que su escalabilidad vertical sea mucho más fácil que la de los modelos relacionales. Muchas veces los almacenes de tipo Key-value se usan en combinación con formatos de serialización de objetos tales como protocolos Buffers, Thrift, BERT, BSON y JSON; que ayudan a almacenar objetos complejos.

Almacenes de Gráficos (Graph Stores). Seeger (2010) los define como "son almacenes de datos que normalmente consisten de 3 bloques: Nodes, Edges y Properties; estos son combinados para representar los datos en el mundo real. Los almacenes de gráficos son optimizados para asociar conjunto de datos; estas son una buena opción para etiquetados o anotaciones de metadatos; por lo que sería interesante usarlo junto con un motor de búsqueda" (p. 26), ergo haría básicamente un marcado de los nodos de los dominios con datos diferentes de distintos atributos. Uno de los mayores proyectos de código abierto de base de datos gráfica es Neo4j.

Escalabilidad

Un atributo que se considera importante cuando se evalúan los almacenes de datos, es la habilidad de estos en tratar con una gran cantidad de documentos. Existen dos principales métodos para hacerlo: escalabilidad vertical y escalabilidad horizontal.

Escalabilidad Vertical. Este tipo de escalabilidad implica aumentar los recursos del sistema (como procesadores, memoria, discos y adaptadores de red) al hardware existente o reemplazar hardware existente por otro con mayores recursos de sistema (Microsoft, 2005, párr. 11). Esta escalabilidad se enfoca completamente en un solo nodo; es decir, se enfoca en añadir recursos a un solo nodo dentro de un sistema.

La escalabilidad vertical es idónea cuando se desea mejorar el tiempo de respuesta de los clientes, como en una configuración de equilibrio de carga de red de un servidor de aplicaciones o por ejemplo, si el hardware actual no ofrece un rendimiento adecuado para los usuarios, donde se puede considerar la posibilidad de agregar memoria RAM o más unidades centrales de procesamiento (CPU) a los

servidores de un clúster para satisfacer esa demanda.

Escalabilidad Horizontal. Según Seeger (2010) "la escalabilidad horizontal describe la habilidad de un sistema en distribuir los datos en más de un simple servidor" (p. 28); complementando esta definición, este tipo de escalabilidad, además permite agregar servidores para atender la demanda en un clúster de servidores de servicios, esto significa agregar nodos al clúster; por lo que el sistema debe tener la capacidad de distribuir los datos sobre todos los nodos del clúster de servidores. Esto debería hacerse en un manera transparente que no requiera tiempo o tareas administrativas complejas.

Los sistemas de búsqueda que tienen la gran tarea de almacenar millones de datos en un solo servidor, deberían de enmarcar su almacén de datos en una escalabilidad horizontal para que pueda permitir la ejecución de las tareas almacenamiento a través de varios servidores y así no sobrecargar sus recursos.

Estructuras de Datos

En esta sección se dará un resumen sobre las estructuras de datos que generalmente usan lo almacenes de datos y las implicaciones que resulta al usarlos. Es importante analizar estas estructuras ya que de ellas, en gran manera depende la rapidez con que realizan las operaciones los almacenes.

Árbol-B (B-trees). En general, los B-trees son estructura de arboles balanceados que son optimizados para situaciones en donde no hay suficiente memoria RAM; para mantener toda la estructura de los datos en memoria. Los Árbol-B permiten una eficiente inserción, actualización y eliminación de ítems que son identificados por una clave (*Ver Figura No. 11*).

Solar y Figueroa (2010) sugieren que, "estos surgen de la necesidad de mantener índices en almacenamiento externo para acceso a bases de datos" (Sección*:* Introducción, párr. 1), ya que existe un grave problema de la lentitud de estos dispositivos, donde se pretende aprovechar la gran capacidad de almacenamiento para

mantener una cantidad de información muy alta, organizada de forma que el acceso a una clave sea lo más rápido posible.

La representación más común de un Árbol-B en un almacén de datos es un Árbol-B+. En este, en contraste a un Árbol-B, toda la información se guarda en las hojas. Los nodos internos sólo contienen claves y punteros.

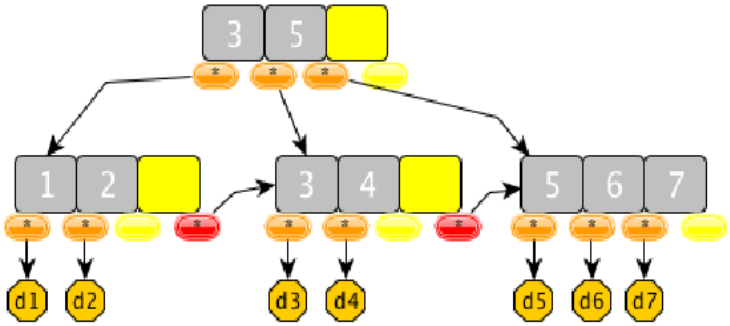

Figura No. 11. **Árbol B+ que enlaza los elementos 1 al 7 a valores de datos d_1-d_7.** Fuente: El Investigador.

Mapa Hash - Tabla Hash (Hash Map – Hash Table). Meneses (2003) la define como "una estructura de datos que intenta hacer eficientes todas las operaciones básicas, de manera que su comportamiento sea casi constante" (p. 3), esta estructura asocia llaves o claves con valores (*Ver Figura No. 12*). La operación principal que soporta de manera eficiente es la búsqueda; es decir puede permitir el acceso a un grupo de datos almacenados a partir de una clave generada.

Las tablas hash se suelen implementar sobre vectores de una dimensión, aunque se pueden hacer implementaciones multi-dimensionales basadas en varias claves. Comparada con otras estructuras de arrays asociadas, las tablas hash son muy útiles cuando se almacenan grandes cantidades de información; sin embargo como la información es almacenada sin algún tipo de orden, la ordenación de los mismos suele ser bastante lenta.

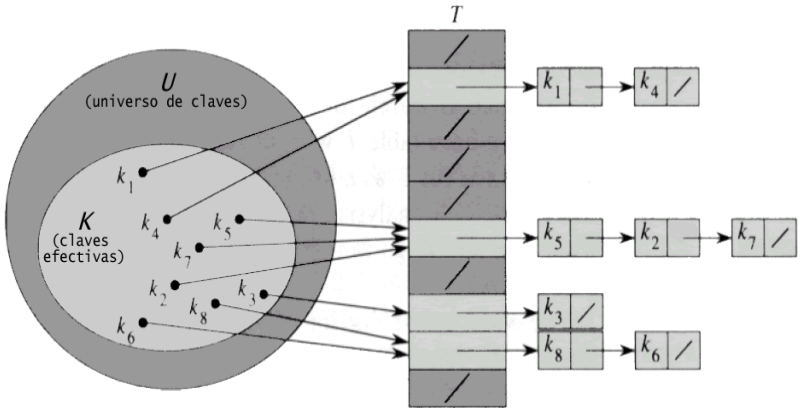

Figura No. 12. **Representación de una Tabla Hash**. Tomado de Zanarini (2010, p. 4).

Árbol-R (R-Tree-Based). Guttman (1984, citado por Calvillo, 2010) los define como "Los árboles-R son estructuras de datos de tipo árbol similares a los árboles-B, con la diferencia de que se utilizan para métodos de acceso espacial, es decir, para indexar información multidimensional; por ejemplo, las coordenadas (x, y) de un lugar geográfico." (p. 18). (*Ver Figura No. 13*).

Así, los algoritmos de búsqueda utilizan los conjuntos límite para decidir en qué nodo buscar. De este modo, la mayoría de los nodos del árbol nunca son examinados durante una búsqueda. Esto hace que este tipo de árboles (como los árboles-B) sean idóneos para el trabajo con bases de datos, debido a su rapidez.

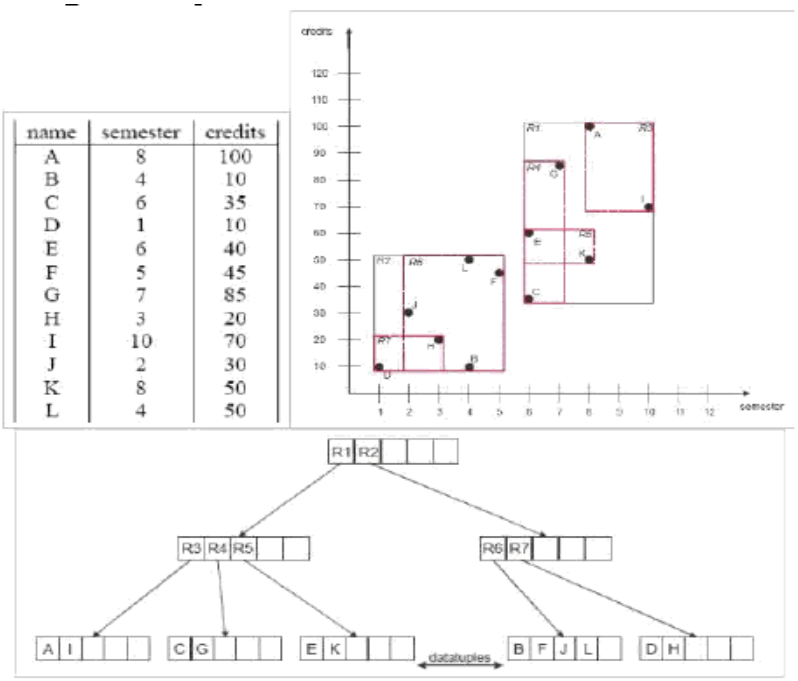

Figura No. 13. **Representación de un Árbol-R**. Tomado de Neumman (2001, citado por Calvillo, 2010, p. 19).

Árbol Hash (Hash-Tree-Based). También conocido como Árbol de Merkle, debido al apellido de la persona quien propuso este ingenioso método; inicialmente fue concebido para almacenar datos codificados para sistemas de seguridad; sin embargo para métodos de almacenamiento no es usado, por su eficiencia; sino más bien en sistemas de almacenamiento distribuidos, para detectar inconsistencia entre las replicas y también para minimizar la cantidad de datos transferidos; ya que es una estructura que permite codificar información resumida de grandes cantidades de datos en forma de árbol.

Árbol Prefijo (Prefix-Tree). Un Árbol de Prefijo parte de la idea de que muchas palabras tienen prefijos comunes, en el cual dicho conjunto se puede representar como un Árbol (ver *Figura No. 14*).

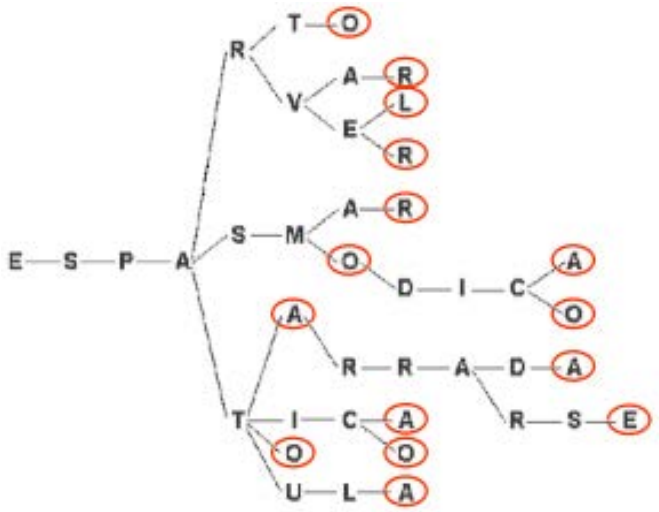

Figura No. 14. ***Representación*** **de Conjuntos Mediante Arboles.** Tomado de García (2010, p. 5).

Por ende se puede decir que un Árbol Prefijo es una estructura de datos de árbol que utiliza las partes de la clave para organizar y buscar entre su colección de datos. Se puede observar (*Ver Figura No. 15*) que la clave no se almacena entera en un nodo, sino en un camino y que todos los descendientes de un nodo tienen un prefijo común.

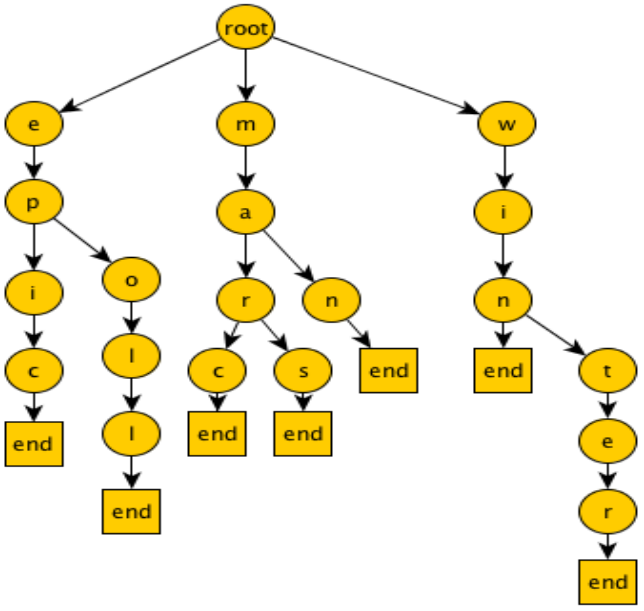

Figura No. 15. **Representación de un Árbol Prefijo.** Tomado de Seeger (2010, p. 35). “Datos: epic, epoll, marc, mars, man, win, winter”.

Por su sistema, estos pueden adaptarse perfectamente a sistemas de diccionarios de corrección ortográfica o en casos de autocompletar entradas de caracteres. Un ejemplo de ello es Lucene, que es una librería para contenidos de búsqueda e indexación.

BitMap. Es un índice que usan las bases de datos relacionales y es usada para datos que tienen una cantidad limitada de posibles valores. Un BitMap usa como principal estructura un bit arrays, lo que ayuda a responder a las búsquedas con operaciones lógicas (*Ver Figura No. 16*).

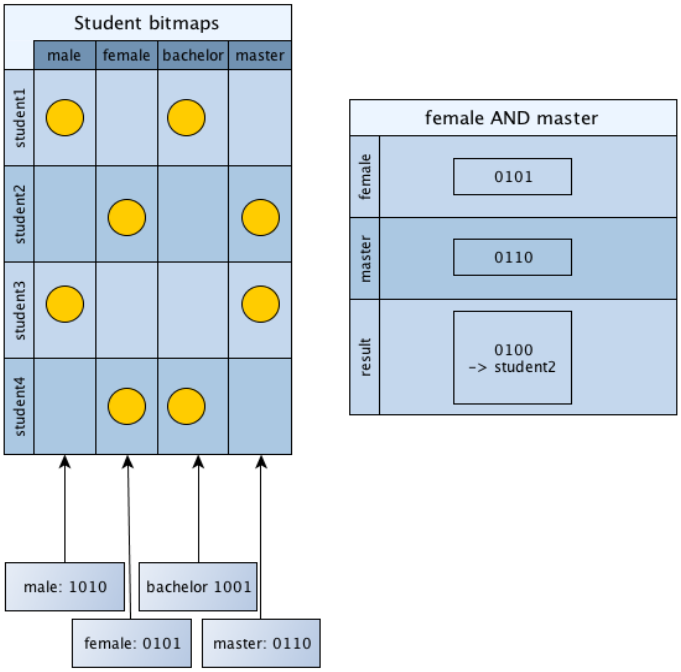

Figura No. 16. **Representación de un Índice Bitmat y Búsqueda Lógica AND.** Tomado de Seeger (2010, p. 37).

Patrones

Según Valera (2010) “En el sentido más general, un patrón es un modelo a seguir que describe una solución exitosa de un problema particular en un contexto

dado” (p. 15).

Buchman, Meunier, Rohnert, Sommerlad y Stal (1996), afirman que un patrón “describe un problema de diseño recurrente que surge en contextos específicos de diseño, y presenta un esquema de solución genérico bien probado” (p. 8). Dicho esquema es delimitado mediante la descripción de los elementos que lo constituyen, sus responsabilidades, relaciones y la manera en que estos colaboran entre sí.

Los patrones envuelven diferentes rangos de escala y de abstracción; además, ayudan a la estructuración de un software en subsistemas y a refinar dichos subsistemas y componentes, así como las relaciones entre ellos. Según Buchman et al (1996), los componentes y sus relaciones no siempre son tan atómicos como parecen ser en un principio; por esta razón, a pesar de que un patrón resuelve un problema particular, su aplicación puede plantear nuevos problemas, lo cuales pueden ser resueltos por otros patrones; de esta manera, componentes individuales dentro de un patrón, puede ser descrito por patrones más pequeños, integrados por el patrón más grande en el que están contenidos.

Los patrones deben ir de la mano con los objetivos de la metodología de desarrollo, para que pueda apoyar en la elaboración, mantenimiento y evolución del sistema; así mismo, deben ser compatibles con los frameworks a usar en la implementación. Cada patrón ayuda a diseñar un aspecto particular de la aplicación, por lo que con un único patrón no se construye la arquitectura completa del software.

Los patrones abarcan desde dominios independientes hasta dominios específicos, dependiendo de su utilidad; por esta razón existen distintas categorías, por lo que en este apartado se abarcarán los patrones arquitecturales y de integración.

Patrones Arquitecturales

De acuerdo a Buchman, Meunier, Rohnert, Sommerlad y Stal (1996) “Un patrón arquitectónico expresa un esquema de organización estructural fundamental para sistemas software. Proporciona un conjunto predefinido de subsistemas,

especifica sus responsabilidades, e incluye las reglas y directrices para la organización de las relaciones entre ellos" (p. 12).

Buchman et al (1996) establecen que los patrones arquitecturales representan el nivel más alto del sistema de patrones. Estos ayudan a construir arquitecturas de software viables de acuerdo con algunos principios generales de estructuración; ya que especifican las propiedades estructurales de todo el sistema, y tienen un impacto en la arquitectura de sus subsistemas o componentes. Por tanto, la decisión del patrón correcto es fundamental en el desarrollo del software.

Buchman et al (1996), agrupan en 4 categorías los patrones arquitecturales, que se explicarán a continuación:

De Estructuración (from mud to structure). Son patrones que soportan una descomposición adecuada de la tarea general de un sistema en subtareas cooperativas. Entre estos están:

1. Layers: patrón que ayuda a estructurar las aplicaciones en grupos de subtareas; en donde las mismas se encuentran en un nivel de abstracción.

2. Pipes and Filters: proporciona una organización para aplicaciones que procesan un flujo de datos. Cada proceso se encapsula en un componente "filtro" y los datos se pasan a través de "tubos".

3. Blackboard: Varios subsistemas especializados comparten conocimiento para construir una solución parcial o aproximada.

Sistemas Distribuidos. Aquellos que proporcionan infraestructuras para sistemas que tienen componentes situados en procesos diferentes o en varios subsistemas y componentes. Hay un patrón en esta categoría:

1. Broker: Muy usado en sistemas de aplicaciones distribuidas con componentes desacoplados que interactúan por llamados de servicios remotos; mediante un componente responsable de coordinar la comunicación, transmisión de resultados y excepciones.

Sistemas Interactivos. Referente a que ayudan a estructurar sistemas con interacción hombre-máquina.

1. MVC: divide un sistema en tres componentes; el modelo que contiene los

datos y las funcionalidades; la vista que muestra la información al usuario y el controlador que responde a las entradas de los usuarios.

2. PAC: este patrón define una estructura para los sistemas interactivos en forma de jerarquía de agentes que cooperan entre sí; en donde cada uno es responsable de una tarea específica de las funcionalidades del sistema, por cuanto constan de presentación, abstracción y control.

Sistemas Adaptables. Los cuales proporcionan infraestructuras para la extensión y adaptación de aplicaciones en respuesta a requisitos funcionales que cambian y evolucionan.

1. Reflection: provee un método para cambiar la estructura y el comportamiento de los sistemas de forma dinámica. Este patrón se divide en dos partes, un nivel meta que proporciona información sobre las propiedades del sistema y un nivel base que incluye la lógica de la presentación. Cada uno están interrelacionados, de manera que los cambios en la información meta influye en el comportamiento del nivel base.

2. Microkernel: Son útiles a sistemas capaces de adaptarse a los cambios de requisitos; separando los componentes en tareas especificas y orquestando su colaboración.

Estilo Arquitectónico

De acuerdo a Garlan y Show (1994), un estilo arquitectónico es “Una familia de sistemas en términos de un patrón de organización estructural. Más específicamente, un estilo arquitectónico determina el vocabulario de los componentes y conectores que se pueden utilizar en instancias de ese estilo, junto con un conjunto de restricciones sobre cómo pueden ser combinados” (p. 6).

Microsoft (2009), definen ocho estilos, albergados en 4 categorías, como se muestra en la *Cuadro No. 4* y en la *Cuadro No. 5* se muestra las definiciones de cada estilo arquitectural.

Cuadro No. 4
Estilos Arquitectónicos

Categoría	Estilo Arquitectónico
Comunicación	• Service-Oriented Architecture (SOA) • Message Bus
Implementación	• Client/Server • 3-Tier / N-Tier
Dominio	• Domain Driven Design
Estructura	• Object-Oriented • Component-Based • Layered

Nota. Tomado de Microsoft (2009, Patterns & Practices).

Cuadro No. 5
Descripción de Estilos Arquitectónicos

Estilo	Descripción
SOA (Comunicación)	Se refiere a las aplicaciones que exponen y consumen funcionalidad como un servicio a través de contratos y mensajes.
Message Bus (Comunicación)	Prescribe el uso de un sistema de software que puede recibir y enviar mensajes usando uno o más canales de comunicación, de modo que las aplicaciones puedan interactuar sin necesidad de conocer los detalles específicos de cada uno.
Client/Server (Implementación)	Segrega el sistema en dos aplicaciones, donde el cliente realiza peticiones al servidor. En muchos casos, el servidor es una base de datos con la lógica de la aplicación representada en procedimientos almacenados.

Estilo	Descripción
3-Tiers / N-Tiers (Implementación)	Segrega funcionalidad en segmentos separados en gran parte de la misma manera como el estilo de capa, pero con cada segmento en un nivel situado en un equipo separado físicamente.
Domain Driven Design (Dominio)	Un estilo orientado a objetos que se centra en el modelado del dominio de negocio y en la definición de objetos de negocio basado en las entidades en el ámbito empresarial.
Object-Oriented (Estructura)	Basado en la división de responsabilidades para una aplicación o sistema en cada objetos reutilizables y auto-suficientes, cada uno con los datos y el comportamiento de los relevantes para el objeto.
Component-Based (Estructura)	Se descompone en el diseño de aplicaciones en funcionalidad reutilizable o componentes lógicos que exponen interfaces de comunicación bien definida.
Layered (Estructura)	Partición de los asuntos de la aplicación en grupos apilados (capas).

Nota. Tomado de Microsoft (2009, Patterns & Practices).

Patrones de Integración

Frecuentemente las empresas poseen sistemas de distintas índoles, que operan en múltiples niveles de las diferentes plataformas. Existen muchos escenarios de integración, en donde se encuentran aplicaciones de desarrollos propios, aplicaciones compradas a proveedores, aplicaciones que se ejecutan en varios equipos y pueden estar geográficamente dispersas, aplicaciones accesadas fuera de la empresa por

socios comerciales o clientes, etc. En ocasiones surge la necesidad de integrar estas aplicaciones a pesar de que no fueron diseñadas para ello y no se pueden cambiar. Estos temas y otros, son los que hacen difícil la integración de aplicaciones

El objetivo de la integración de es hacer que las aplicaciones separadas trabajen colaborativamente para producir un conjunto unificado de funcionalidad. Es allí, donde los patrones de integración exploran las opciones disponibles para la integración de aplicaciones.

Los patrones de integración, definen diseños comunes en el desarrollo de funcionalidades relacionadas con la integración de aplicaciones. Especifican una manera estándar de realizar tareas y ayudan a conocer con un lenguaje común determinadas funciones.

Hohpe (2010) establece que:

> Los patrones son una forma de capturar el conocimiento de expertos en las áreas donde no hay simple respuesta de solución que encaje para todos, como la arquitectura de aplicaciones, diseño orientado a objeto o integración orientado a mensajes. Cada modelo plantea un problema de diseño específico, sobre las consideraciones en torno al problema, y presenta una elegante solución que equilibra las distintas fuerzas o manejadores. En la mayoría de los casos, la solución no es la primera aproximación que viene a la mente, pero que ha evolucionado a través del uso real en el tiempo. Como resultado, cada modelo incorpora como base la experiencia que los desarrolladores de integración y arquitectos han ganado en la construcción de soluciones, aprendiendo de los errores. Esto implica que no son patrones inventados, sino descubiertos y observados en la práctica real. (Secc. Como los patrones pueden ayudar)

De acuerdo a Hohpe y Woolf (2003), hay más de un enfoque de integración de aplicaciones. Cada enfoque se dirige a criterios de integración mejor que otros. Los diferentes enfoques se pueden resumir en cuatro estilos principales de integración:

File Transfer. Este estilo resume aquellos patrones en donde según Hohpe y Woolf (2003) “la aplicación produce un archivo compartido de datos para que otros lo consuman y además consume los archivos que otros han producido” (p. 65).

Los archivos son estándares de almacenamiento universal, que es previsto por cualquier sistema operativo y accesible desde cualquier lenguaje; por lo que la idea

sería integrar las aplicaciones que utilizan archivos. En la *Figura No. 17*, se puede observar el estilo.

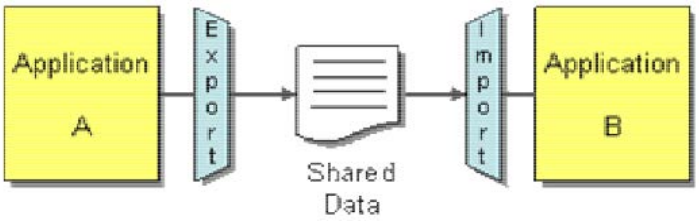

Figura No. 17. **Estilo File Transfer.** Tomado de Hohpe y Woolf (2003, p. 66).

En este estilo, una aplicación proporciona el archivo. El contenido y el formato del archivo se negocian con los integradores; entonces se realizan las transformaciones necesarias para las aplicaciones consumidoras, o se delega la función para que dichos consumidores decidan cómo quieren manipular y leer el archivo. Como resultado se obtienen aplicaciones desacopladas, donde cada aplicación puede hacer cambios sin afectar a otras aplicaciones, siempre y cuando produzcan el mismo contenido y formato.

Shared Database. Se refiere a aquellos patrones que se usan en situaciones donde lo que se desea compartir entre las aplicaciones es la base de datos. La transferencia de archivo permite a las aplicaciones compartir datos (File Transfer), pero carecen de actualización de los datos en tiempo real; una de las metas de la integración es compartir la información rápida y consistente.

En la *Figura No. 18* se puede observar cómo funciona el presente estilo de patrones.

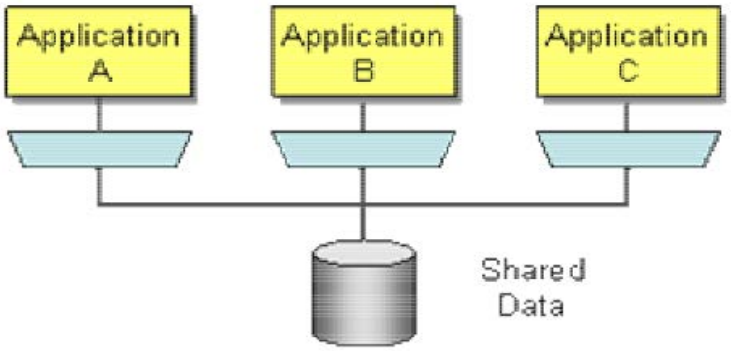

Figura No. 18. **Estilo Shared Database.** Tomado de Hohpe y Woolf (2003, p. 69).

De acuerdo a Hohpe y Woolf, como todas las aplicaciones comparten la misma base de datos, esto trae consigo problemas de disonancia semántica, lo que a su vez lleva a grandes cantidades de datos incompatibles; al igual que el acceso a los datos de manera recurrente causa cuellos de botella, motivado a que cada aplicación debe bloquear los datos mientras son modificados.

Remote Procedure Invocation (RPC). Dichos patrones aplican el principio de encapsulamiento para la integración; en donde las aplicaciones necesitan información provista por otra, haciéndolo a través de llamadas, cada una manteniendo la integridad de sus propios datos; sin tener que afectar los datos de las otras aplicaciones.

Los RPC necesitan de un mecanismo para invocar una función de una aplicación, pasando los datos que necesitan ser compartidos y dicha función determina cómo procesar los datos. Cada aplicación se desarrolla como un componente con datos encapsulados; proporcionando una interfaz para permitir la interacción. Este método se puede observar en la *Figura No. 19*

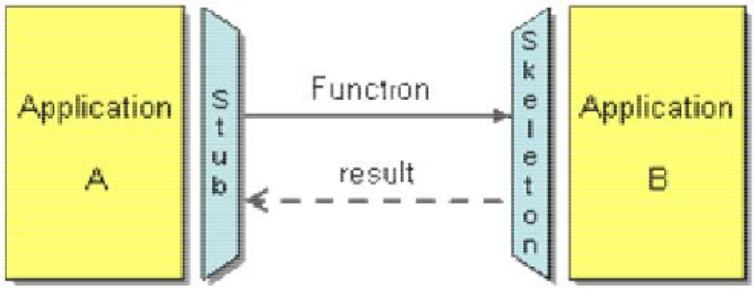

Figura No. 19. **Estilo Remote Procedure Invocation**. Tomado de Hohpe y Woolf (2003, p. 69)

Existen varias implementaciones RPC como CORBA, COM, .Net Remoting, Java RMI, entre otras; hoy en día lo más usado son los WebServices.

Aunque el encapsulamiento ayuda a reducir el acoplamiento de las aplicaciones, mediante la eliminación de una gran estructura de datos compartidos, las aplicaciones son aún acopladas. En particular según Hohpe y Woolf, la secuencia de acciones entre las aplicaciones puede hacer que sea difícil cambiar los sistemas de forma independiente, sobre todo si se usan distintas plataformas de desarrollo. Los desarrolladores diseñan métodos de integración desde el punto de vista de una aplicación, no previniendo que las reglas del negocio son susceptibles a cambios.

Message. En este estilo de patrones, las aplicaciones se conectan por medio de un sistema de mensajes, intercambiando los datos. File Transfer y Shared Database permiten compartir los datos pero no la funcionalidad, por otro lado, Remote Procedure Invocation permite compartir la funcionalidad pero estas se encuentran estrechamente vinculadas a los procesos. El funcionamiento de Message se puede observar en la *Figura No. 20.*

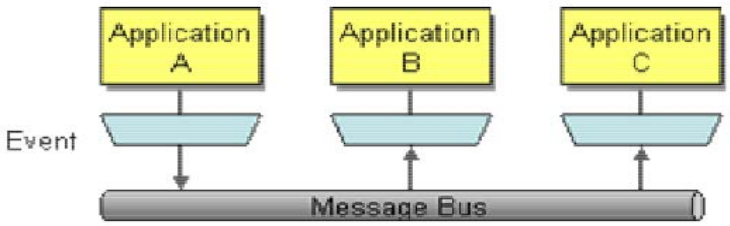

Figura No. 20. **Estilo Messaging**. Tomado de Hohpe y Woolf (2003, p. 73).

Uno de los objetivos de la integración es que los sistemas colaboren en tiempo real, sin estar acoplados. Hohpe y Woolf señalan que el uso de mensajes para transferir datos frecuentemente, inmediatamente, confiablemente y asíncronamente, usando formatos configurables; ayudan a la integración de aplicaciones. Además agregan que "El mensaje asíncrono es fundamentalmente una reacción practica a los problemas de sistemas distribuidos" (p. 73); esto debido a que todas las aplicaciones no necesitan estar en ejecución al mismo tiempo; incluso, pueden colaborar aplicaciones cuyo tiempo de respuesta es lento, lo que fomenta el diseño de componentes con alta cohesión y de baja adherencia. Los mensajes también permiten el bajo acoplamiento y este puede ser transformado en un punto intermedio de la comunicación sin que el emisor o el receptor estén involucrados en dicho proceso. El sistema de mensajes es el responsable de la entrega de los datos, por tanto, las aplicaciones se centran en que data deben compartir pero no en cómo deben compartirla.

Según los autores, "Los Mensajes son más inmediatos que File Transfer, mejor encapsulado que Shared Database y más seguro que Remote Procedure Invocation".

Los siguientes son los patrones de mensajería propuestos por Hohpe y Woolf (2003), asociados en 7 grupos.

1. Messaging Channels: Las aplicaciones se conectan por medio de mensajes. **Patrones**: Point-to-Point Channel, Publish-Subscribe Channel, Datatype Channel, Invalid Message Channel, Dead Letter Channel, Guaranteed Delivery, Channel Adapter, Messaging Bridge, Message Bus.

2. Message Construction: Las aplicaciones conectadas por Messaging Channel intercambian una pieza de información. **Patrones**: Command Message, Document Message, Event Message, Request-Reply, Return Address, Correlation Identifier, Message Sequence, Message Expiration, Format Indicator.

3. Pipes and Filters: El procesamiento complejo por medios de mensajes es mejorado, manteniendo independencia y flexibilidad.

4. Message Router: El procesamiento individual es desacoplado, a tal forma que los mensajes puedan ser pasados a diferentes filtros dependiendo de un conjunto

de condiciones. **Patrones**: Content-Based Router, Message Filter, Dynamic Router, Recipient List, Splitter, Aggregator, Resequencer, Composed Message Processor, Scatter-Gather, Rounting Slip, Process Manager, Message Broker.

5. Message Translator: Los sistemas usan datos en distintos formatos para comunicarse usando mensajes. **Patrones**: Envelope Wrapper, Content Enricher, Content Filter, Claim Check, Normalizer, Canonical Data Model.

6. Message Endpoint: Las aplicaciones se conectan al canal de mensajes para enviar y recibir mensajes. **Patrones**: Messaging Gateway, Messaging Mapper, Transactional Client, Polling Consumer, Event-Driven Consumer, Competing Consumers, Message Dispatcher, Selective Consumer, Surable Subscriber, Idempotent Receiver, Service Activator.

7. System Management: Las soluciones deben ser monitoreadas para conocer la cantidad de mensajes enviados y cuanto es el tiempo de procesamiento. **Patrones**: Control Bus, Detour, Wire Tap, Message History, Message Store, Smart Proxy, Test Message, Chennel Purger.

Cada estilo tiene sus ventajas y desventajas. Dos aplicaciones pueden integrarse con múltiples estilos de tal manera que cada punto de integración aprovecha el estilo que se adapte mejor; del mismo modo, una aplicación puede utilizar diferentes estilos de integración con diferentes aplicaciones. Algunos métodos de integración pueden resultar como un híbrido de varios estilos.

Hohpe y Woolf, (2003) señalan sobre la mensajería “creemos que es a menudo el mejor estilo para la solución de muchas oportunidades de integración. Es también el menos conocido de los estilos de integración y una tecnología madura con patrones que ayudan rápidamente cómo hacer buen uso de ella” (p. 65). La mensajería es la base para muchos productos de EAI como Mule, ServiMix, OpenESB, BizTalk, WebSphere ESB, entre otras**.**

Arquitectura Orientada a Servicio (SOA)

Las organizaciones necesitan adaptarse a los distintos cambios de su entorno; por lo tanto sus procesos siempre están en contaste cambio, lo que a su vez implica que estos deben ser dinámicos y adaptados a las condiciones como única forma de mantener y mejorar su competitividad. Estos procesos constantemente son soportados por sistemas, que por la misma naturaleza cambiante significa que también deben adaptarse fácilmente a los cambios para responder de forma ágil a las necesidades del negocio. Es a partir de este planteamiento que nace SOA (Service Oriented Architecture), para dar respuesta a este tipo de retos.

Según Martin (2004, citado por la revista Computing, 2004) define SOA como "Un nuevo paradigma de arquitectura para construir servicios y soluciones. Busca abordar el desarrollo de aplicaciones, pero pensando desde el primer momento en la integración entre esas aplicaciones".

De acuerdo a la W3C (2004) "Una Arquitectura Orientada a Servicio es una forma de arquitectura de sistemas distribuidos que se caracterizan por las siguientes propiedades: Vista Lógica, Orientado a Mensaje, Orientado a Descripción, Granularidad, Orientado a Red y Plataforma Neutral" (p. 61).

Antes de dar una definición más amplia, es importante aclarar que SOA no es una tecnología ni un producto que se pueda comprar e instalar; sino como lo especifica Borja (2004, citado por la revista Computing, 2004) "también hay que hablar de SOA como una filosofía y conjunto de guías de desarrollo, patrones y buenas prácticas, es decir, una nueva metodología" (p. 5). En este sentido, SOA supera tecnológicamente a paradigmas como Java RMI o .NET Remoting que no permiten la interoperabilidad entre distintas tecnologías.

A pesar de que los componentes son la mejor forma de implementar servicios, se debe entender que una aplicación correctamente basada en componentes, no necesariamente es una aplicación correctamente orientada a servicios. Por ello, BPS (2006), determina que "la clave para comprender esta diferencia radica en ver como una arquitectura orientada a servicios (SOA) implica una capa adicional de

arquitectura (una nueva abstracción) implementada con una granularidad más “gruesa” y ubicada más cerca del consumidor de la aplicación.” (p. 9).

La importancia arquitectural de SOA está en la exposición de interfaces abstractas que aíslan la implementación particular de cada pieza de software. Existen muchas maneras de implementar SOA como arquitectura, pero una de ellas para conseguir este objetivo, es el uso de WebServices. Sin embargo, hay que tener en cuenta que es posible tener WebServices y no por ello tener SOA. Es de vital importancia entender que SOA es un problema de diseño y no de uso de una determinada tecnología o lenguaje de desarrollo.

Problemas de Integración de Aplicaciones

Los dos grandes enemigos de la integración son el acoplamiento y la no adecuación de estándares. Se dice que un sistema está fuertemente acoplado cuando resulta muy complicado o incluso imposible modificar alguna de sus partes sin que no resulten afectadas las demás. Es decir, cada módulo o subsistema conoce y necesita demasiada información del entorno en el que opera, resultando por tanto poco autónomo y muy dependiente. En términos de integración, cuanto más acopladas estén las distintas partes de un sistema más difícil resulta integrarse con alguna de ellas sin tener que hacerlo con las demás o con el sistema completo.

El otro enemigo de la integración es la no adecuación de estándares. Los sistemas que están bien diseñados, siguen una metodología y patrones de desarrollo, presentando interfaces de integración que permiten el diálogo con otros sistemas ya sean dentro de la misma organización o con sistemas de terceros, aliados, proveedores y organismos reguladores. El uso de estándares facilita enormemente este diálogo dado que minimiza la necesidad de desarrollar software específico para llevar a cabo esta integración. Si cada módulo del sistema se desarrolla teniendo en mente los estándares se está preparando para una larga vida interviniendo en los procesos de negocio de la organización.

Bus de Integración de Servicio (ESB)

La competitividad, los nuevos retos y los cambios en los procesos del negocio hacen indispensable la innovación tecnológica; estos cambios siempre requieren el apoyo de sistemas, que a su vez deben también adaptarse a los cambios producidos en su entorno. La innovación normalmente trae consigo la proliferación de sistemas de información que a menudo genera un estado de entropía tecnológica. La respuesta no puede ser la renuncia a la innovación sino que debe dar pie a abordar seriamente el problema de la integración entre sistemas, lo que se denomina comúnmente EAI o Enterprise Application Integration.

Vázquez (2010) define "EAI es el intercambio sin restricciones de datos y procesos de negocio entre cualquier aplicación y fuente de datos existente en una empresa" (párr. 6). Este concepto siempre ha estado en las organizaciones desde la explosión de la era tecnológica; por lo que ha dado a pie a procesos de integración; las primeras soluciones de integración se hacían arquitecturalmente a través de Sistemas Punto a Punto o a través de Base de Datos.

Aún hoy en día, tal vez por su simplicidad y bajo costo, se siguen usando estos métodos como soluciones de integración; sin embargo, a medida que la complejidad tecnológica aumenta, estos métodos no responden a las necesidades, por ello se han diseñado nuevos métodos de integración más sofisticados y ambiciosos como:

1. Hub-and-Spoke: todos los sistemas se conectan a un punto central o hub como en el caso de Microsoft BizTalk. El problema es que este punto central se convierte en extremadamente importante para la organización. Un fallo en el hub podría comprometer todos los procesos de negocio de la empresa.
2. Enterprise Message Bus o broker de mensajes: en este caso no hay un punto central sino que los conectores están desacoplados gracias al intercambio de mensajes a través de un broker.

Es aquí donde el Bus de Servicio Empresarial juega un papel importante y donde ayuda a las organizaciones en el problema de integración de sistemas que

puedan presentar. La definición de un Bus de Servicio Empresarial o ESB de acuerdo a Caponi, Rodríguez y Zamudio (2008) es la siguiente:

> Es una plataforma de integración de aplicaciones basada en estándares, que combina entre otras cosas: mensajería, servicios, ruteo y transformación de mensajes, con el objetivo de coordinar y conectar de manera confiable un número significativo de aplicaciones. Está construido sobre un canal común de mensajes (bus) altamente distribuible, multiprotocolo, basado en estándares y provee la columna vertebral para implementar una Arquitectura Orientada a Servicios (SOA). (p. 28).

En la *Figura No. 21* se ilustra la infraestructura de este tipo de plataformas.

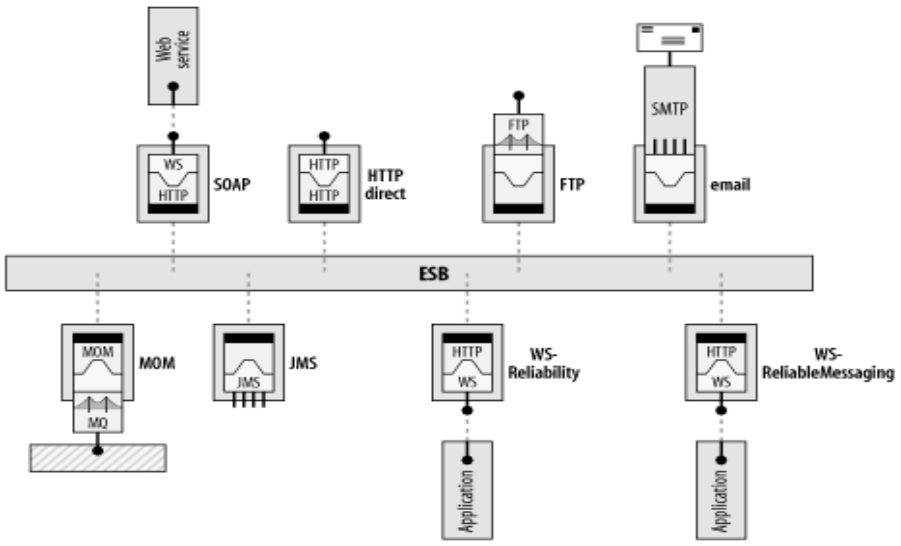

Figura No. 21. **Capacidades de integración multiprotocolo de un ESB.** Tomado de Caponi, Rodríguez y Zamudio (2008, p. 28)

Un ESB al igual que SOA, parte de la idea de desacoplamiento. Un ESB usa estándares abiertos e interoperables, sobre una capa de transporte, típicamente HTTP. De esta manera los conectores no definen la implementación sino la capa de transporte y una interfaz de servicio. El aporte que hacen los ESB al proceso de integración, radica en que puede distribuirse a lo largo de la organización, no necesitando un punto central de integración, y permitiendo la interoperabilidad entre

sistemas implementados en las más diversas tecnologías. Las características básicas, según Vázquez (2010), que debe presentar un ESB son las siguientes:

- Enrutamiento y redireccionamiento de mensajes.
- Estilo de comunicación síncrono y asíncrono.
- Multiplicidad de tipos de transporte y protocolos de enlace.
- Transformación de contenido y traducción de mensajes.
- Orquestación y coreografía de procesos de negocio.
- Procesamiento de eventos.
- Presencia de adaptadores a múltiples plataformas.
- Herramientas de diseño de la integración, de implementación y despliegue.
- Características de garantía de la calidad del servicio (QoS), como transaccionalidad, seguridad y persistencia.
- Auditoría, registro y métricas.
- Gestión y monitorización.

Existen varias implementaciones de ESB ya sean de código abierto o propietarios como Sonic ESB, Oracle Enterprise Service Bus, Apache ServiceMix y Mule.

La implantación de soluciones ESB, conllevan a ventajas significativas, sobre todo en organizaciones que poseen varias aplicaciones en plataformas distintas y que necesitan de un proceso de integración; resolviendo gran parte del proceso, permitiendo tener un buen control del mismo.

Componentes Básicos de un ESB

De Acuerdo a Pacheco (2009) los principales componentes de un ESB son los siguientes:

1. Invocación: Se encarga de proporcionar apoyo a los protocolos de transporte de manera síncrona y asíncrona.

2. Routing: Responsable para el envío de la información para un lugar u otro, lo hace de un modo estático o dinámico. Se puede usar el enrutamiento basado en reglas con Drools por ejemplo.

3. Mediación (Trasformación): Se encarga de proporcionar la transformación de protocolos, por ejemplo entrar en una http y salir en un SFTP.

4. Mensaje: Se encarga de proporcionar el tratamiento, procesamiento y el refuerzo de mensajes. Por ejemplo, si lee un xml el ESB debe ser capaz de añadir o eliminar información de ese XML antes de llegar al destino.

5. Orquestación / Coreografía: Se refieren a procesos complejos de BPMN/BPEL.

Uso de un ESB Dentro de una Arquitectura SOA

Es muy importante destacar, y que normalmente es una concepción errada, es que el uso de un ESB no necesariamente significa tener SOA. Es vital aclarar que un ESB se centra solo en la integración de los sistemas, específicamente de los protocolos, plataformas y formas de acceso; en cambio SOA trata de contratos y reutilización de sistemas.

Se debe tener en cuenta que la integración es una parte muy importante, pero SOA es mucho más que integración; lograr SOA va a depender de la forma en que se modela y que se concibe un sistema. ESB juega un papel primordial dentro de SOA como componente indispensable en la solución ya que toda integración de sistemas pasa a través de él.

Ahora bien, a pesar de que un ESB puede llegar a ser una parte vital para una SOA, este puede no ser la mejor solución al problema, se debería usar cuando se tienen varios sistemas que necesitan conversar de diferentes maneras y a veces están distribuidos a través de La Web. Si sólo se dispone de 2 sistemas Java en una empresa no valdría la pena usar un ESB.

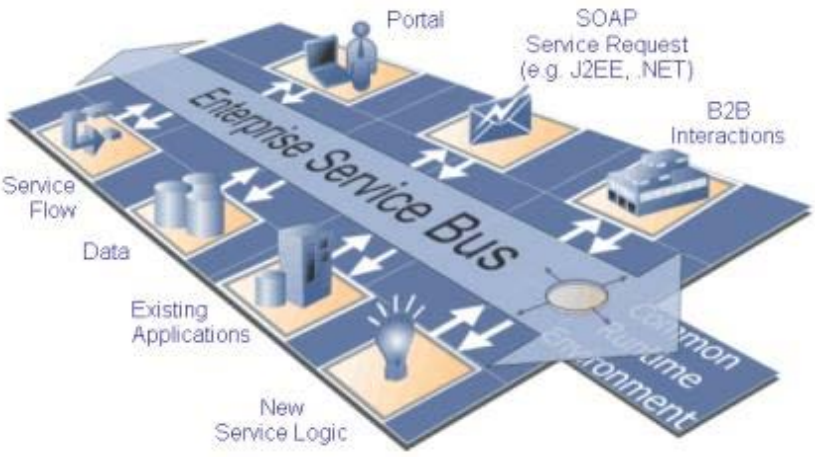

Figura No. 22. **Representación de un Bus de Integración.** Tomado de Urrutia (2006).

Se puede ver en la *Figura No. 22*, el rol de integración con las demás aplicaciones que juega un ESB en una arquitectura SOA; este no trata con lógica de negocio, trata con lógica de servicios, esto es resolución de servicios, monitorización de servicios, acceso de servicios, homogeneización de los mismos; el nivel de codificación en la capa ESB es menor, esa es la ventaja de este tipo de Middlewares.

¿Por qué Usar ESB en una Arquitectura SOA?

Como se explicó en la definición, ESB funge como una especie de orquestador de servicio, que permite interactuar de mejor forma y rápidamente los componentes técnicos y de información con aquellos relacionados a los procesos de negocio de las capas superiores. Una arquitectura SOA, usada junto con un ESB permite mantener la flexibilidad necesaria en los procesos de negocio a nivel técnico y de codificación.

El aporte de un ESB va más allá de una herramienta que facilite integraciones del tipo WebServices; sino que habilita un middleware de servicios orientados a mensajería como JMS, administra colas y prioriza datos, y comunicaciones de contenidos; administrando los estados de estas integraciones entre sistemas distribuidos.

En resumen un ESB no implementa una solución SOA, pero si la da las herramientas para facilitar ese tipo de arquitecturas y además proveer otros mecanismos de integración, como mensajería, adaptadores hacia repositorios, listeners y otros medios de administración de servicios.

WebServices

Booth et al (1986, citado por Caponi, Rodríguez y Zamudio, 2004), plantea los WebServices como "uno de los principales mecanismos para lograr interoperabilidad entre plataformas heterogéneas; a su vez, son una tecnología clave para implementar una Arquitectura Orientada a Servicios (SOA)" (p. 32).

De acuerdo a esta definición, en sentido amplio, los WebServices ayudan en la forma de comunicar aplicaciones, ya sea que se encuentren en plataformas distintas. En la actualidad, existen dos metodologías de servicios principales, aquellas basadas en SOAP y REST. En la *Cuadro No. 6*, se puede ver algunas diferencias entre estos dos estilos.

Cuadro No. 6.
Diferencias entre SOAP y REST

	REST	**SOAP**
Formato de Mensajes	XML	XML
Definición de Interfaz	Ninguna	WSDL
Transporte	HTTP	HTTP, FTP, MIME, JMS, SMTP, etc.

Fuente. El Investigador.

Además, en la *Cuadro No. 7*, se puede observar las características, ventajas y desventajas de cada estilo de acuerdo Navarro (2007).

Cuadro No. 7.
Características, Ventajas y Desventajas entre SOAP y REST

	REST	SOAP
Características	- Las operaciones se definen en los mensajes. - Una dirección única para cada instancia del proceso. - Cada objeto soporta las operaciones estándares definidas. - Componentes débilmente acoplados.	- Las operaciones son definidas como puertos WSDL. - Dirección única para todas las operaciones. - Múltiple instancias del proceso comparten la misma operación. - Componentes fuertemente acoplados.
Ventajas	- Bajo consumo de recursos. - Las instancias del proceso son creadas explícitamente. - El cliente no necesita información de enrutamiento a partir de la URI inicial. - Los clientes pueden tener una interfaz “listener” (escuchadora) genérica para las notificaciones. - Generalmente fácil de construir y adoptar.	- Fácil (generalmente) de utilizar. - La depuración es posible. - Las operaciones complejas pueden ser escondidas detrás de una fachada. - Envolver APIs existentes es sencillo. - Incrementa la privacidad. - Herramientas de desarrollo.
Desventajas	- Gran número de objetos. - Manejar el espacio de nombres (URIs) puede ser engorroso. - La descripción sintáctica/semántica muy informal (orientada al usuario). - Pocas herramientas de	- Los clientes necesitan saber las operaciones y su semántica antes del uso. - Los clientes necesitan puertos dedicados para diferentes tipos de notificaciones. - Las instancias del proceso son

	REST	SOAP
	desarrollo.	creadas implícitamente.

Nota. Tomado de Navarro (2007).

SOAP WebServices

La arquitectura que envuelve a estos servicios está basada en estándares y especificaciones que proveen un marco de trabajo para construir aplicaciones interoperables sobre una red. De acuerdo a W3C, estos tienen una interfaz descrita en un formato procesable llamado WSDL; donde los sistemas interactúan de una manera prescrita por su descripción utilizando mensajes SOAP, típicamente transmitidos a través de HTTP con una serialización XML. Los estándares están enteramente basados en XML con el objetivo de que sean independientes de la plataforma y del lenguaje de programación. Así, el corazón de la arquitectura consiste de tres especificaciones, Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP), y Universal Description, Discovery, and Integration (UDDI). La misma se puede ver en la *Figura No. 23.*

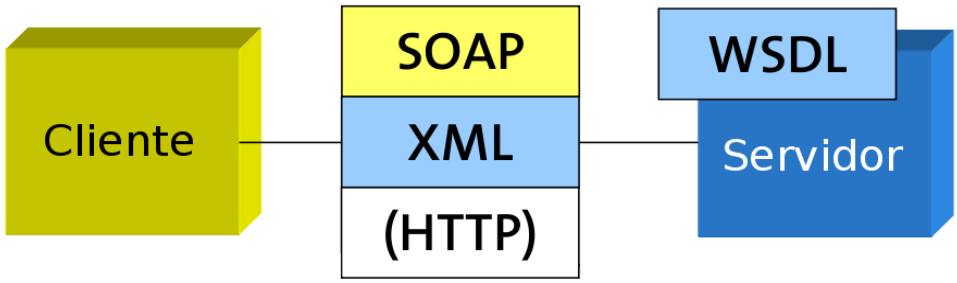

Figura No. 23. **Arquitectura SOAP WebServices.** El Investigador.

WSDL es definido por Chinnici, Gudgin y Moreau (2003, citado por Venzke, 2003) como “el lenguaje para describir la interfaz de un Web Service, esto comprende: su ubicación física, protocolo de comunicación, operaciones disponibles, formatos de

mensajes que componen las operaciones, y tipos de datos utilizados en los mensajes" (p. 2).

Gudgin, Hadley, Mendelshon, Moreau y Frystyk (2003, citado por Parra, Sánchez, Sanjuán y Joyanes, 2005) definen SOAP como "un protocolo para intercambiar información estructurada en un ambiente descentralizado y distribuido" (p. 2).

Según Bellwood, Clement y Von Riegen (2003, citado por Parra, Sánchez, Sanjuán y Joyanes, 2005) establece que UDDI "es un servicio para localizar Servicios Web para los clientes así como un mecanismo para publicarlos" (p. 4).

Los estándares de SOAP conformaron en principio, los denominados Web Services de primera generación, donde se definieron unidades funcionales de negocio bien definidas, pero con limitaciones requeridas en aplicaciones empresariales. Estas limitantes entre otras son: confiabilidad, transaccionalidad, seguridad. Para solucionarlas surge lo que se denomina segunda generación de Web Services. Esta nueva generación define especificaciones construidas sobre las anteriores y compatibles entre sí; especificaciones que en conjunto se suelen denominar WS-*.

REST Web Services

La definición dada por el creador de REST, Fielding (2000), es la siguiente:

> Es una abstracción de los elementos arquitectónicos dentro de un sistema de distribución de hipermedia. REST ignora los detalles de la implementación del componente y la sintaxis de protocolo con el fin de centrarse en las funciones de los componentes, las limitaciones de su interacción con otros componentes, y su interpretación de los elementos de información significativa. Abarca las limitaciones fundamentales sobre los componentes, conectores, y los datos que definen la base de la arquitectura Web, y por lo tanto la esencia de su comportamiento como una aplicación basada en la red. (p. 86).

Por tanto, REST define un estilo arquitectural de la Web, usando la escalabilidad y métodos del protocolo HTTP como GET, POST, PUT y DELETE;

siendo más fácil de entender y usar que SOAP. A continuación en la *Cuadro No. 8*, se puede ver el uso de cada método, para las operaciones CRUD de los recursos. Este estilo arquitectónico describe el funcionamiento de la WEB, siendo un subconjunto de la WWW, basado en HTTP.

Cuadro No. 8.
Relación métodos HTTP con CRUD

CRUD	REST	Descripción
CREATE	POST	Crea un recurso.
READ	GET	Recupera el estado actual de un recurso.
UPDATE	PUT	Inicializa o actualiza el estado de un recurso dado por el URI
DELETE	DELETE	Libera el recurso, luego que el URI no es valido

Fuente. El Investigador.

En la *Figura No. 24*, se puede observar la arquitectura REST.

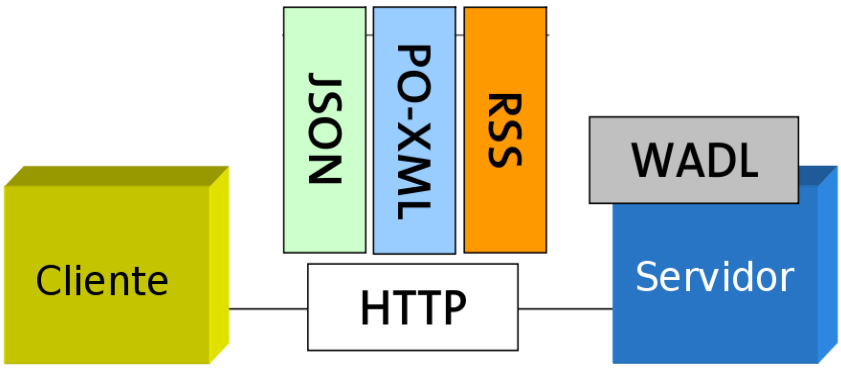

Figura No. 24. **Arquitectura REST WebServices.** El Investigador.

REST, abarca un diseño sin estados previos cliente-servidor, en donde los servicios se visualizan como recursos y se pueden identificar por su URL. Cada

petición del cliente debe contener toda la información necesaria para que el servidor la comprenda y no necesite algún dato almacenado en el contexto de la comunicación; por tal motivo el estado de la sesión se guarda íntegramente en el cliente. Esta condición mejora la visibilidad, eficiencia y escalabilidad.

La visibilidad porque el servidor no tiene que ocuparse de observar en otros sitios ni realizar más operaciones para comprender la naturaleza de una simple petición. La eficiencia mejora porque es más fácil recuperarse de errores parciales. Por último, la escalabilidad se ve también afectada porque al no hacer falta almacenar los estados entre las peticiones, los componentes pueden liberar recursos rápidamente.

Adicionalmente, las respuestas a las peticiones en REST pueden ser etiquetadas como cacheable o no-cacheable. Si una respuesta es cacheable, entonces al cliente cache tiene permiso para reutilizar la respuesta más tarde si se hace una petición equivalente. Por tal razón, se evitarán determinadas peticiones al servidor, mejorando así la eficiencia y escalabilidad, reduciendo el tiempo medio de espera de una serie de interacciones.

Se puede decir que REST es una descripción analítica de las arquitecturas web existentes, por lo tanto la interacción con el protocolo HTTP es muy natural. A pesar de ello, Tyagi (2006) establece “el estilo REST y el protocolo HTTP son mutuamente excluyentes, y REST no requiere HTTP”; entonces, no se debe confundir esta interacción entre REST y el protocolo HTTP.

Ontología

El termino de ontología fue admitido por la informática con el objetivo de representar y organizar el conocimiento, dado a la gran cantidad de información no estructurada que producen los sistemas; en este sentido, esto constituye un problema al momento de manejar y recuperar la misma, provocando falta de precisión y exhaustividad. Así, una de las definiciones más conocida de ontología, es la proporcionada por la IEEE (2003) “Una ontología es similar a un diccionario o

glosario, pero con mucho más detalles y estructuras que permiten a la computadoras procesar su contenido" (párr. 2).

Se puede decir que una ontología es una conceptualización o representación de un dominio, acordada y formalizable computacionalmente; la cual favorece la comunicación que pueda existir entre personas, organizaciones y aplicaciones porque proporciona una comprensión común de un dominio de modo que se eliminan las confusiones conceptuales y terminológicas. Además muestra una forma de representar y compartir el conocimiento utilizando un vocabulario común, permitiendo que el desarrollo de los procesos se lleve a cabo de una manera organizada y estandarizada proporcionando un protocolo específico de comunicación, logrando que los integrantes del dominio entiendan y manejen los mismos términos y conceptos.

La ontología ha sido usada en La Web para convertir la información en conocimiento, mediante metadatos con un esquema común convenido sobre algún dominio; de allí nace la Web Semántica denominada por Berners-Lee, en donde las aplicaciones serán capaces de comprender y relacionar la información contenida en La Web; para ello, la misma debe estar caracterizada y estructurada, a tal modo que su significado exacto pueda ser inferido.

Una parte fundamental de la web semántica son los metadatos que contendrán las páginas web. Esencialmente los metadatos son datos sobre los datos que contiene cada página, los cuales ayudan a describir los contenidos del documento. La web semántica requiere que los metadatos sean relacionales; esto es, metadatos que describen la manera en como las descripciones de un recurso instancian definiciones de clases y la manera en cómo están semánticamente ligadas por propiedades.

Las ontologías proporcionan dos ventajas en este escenario. Ayudan a unir la información que normalmente se encuentra aislada en varias descripciones por separado; y proporciona los conocimientos básicos que permite a los no expertos realizar consultas desde su punto de vista. La falta de relaciones semánticas genera mayores esfuerzos por parte de los usuarios para vincular documentos unos con otros, a fin de filtrar y clasificar los resultados de una consulta a partir de un dominio

específico.

Es importante aclarar que una ontología no es una base de datos ni un programa, ya que tienen sus propios formatos internos; no es una conceptualización, porque no es una especificación, y tampoco es una tabla de contenidos. Las ontologías son acuerdos, en un contexto social, para cubrir una serie de objetivos; que pueden ayudar para describir la funcionalidad de componentes mediante un formalismo de representación del conocimiento.

Guarino (1998), clasifica en cuatro la ontología de acuerdo con su nivel de dependencia de una determinada tarea:

- Ontologías de Alto Nivel o Genéricas: Describen conceptos más generales, como los espacios, tiempos, materia que son independiente de un problema en particular o de un dominio.
- Ontologías de Dominio: Describen un vocabulario relacionado con un dominio genérico, mediante la especialización de los términos introducidos en la ontología de nivel superior.
- Ontologías de Tareas o de Técnicas básicas: Describen una tarea, actividad o artefacto.
- Ontologías de Aplicación: Describen conceptos que dependen tanto de un dominio específico como de una tarea específica y, generalmente son una especialización de ambas.

Componentes de una Ontología

Según Gruber (1993) las ontologías tienen los siguientes componentes que servirán para representar el conocimiento de algún dominio:

- Conceptos: son las ideas básicas que se intentan formalizar. Los conceptos pueden ser clases de objetos, métodos, planes, estrategias, procesos de razonamiento, etc.
- Relaciones: representan la interacción y enlace entre los conceptos del

dominio. Suelen formar la taxonomía del dominio. Por ejemplo: subclase-de, parte-de, parte-exhaustiva-de, conectado-a, etc.

- Funciones: son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología. Por ejemplo, pueden parecer funciones como categorizar-clase, asignar fecha, etc.
- Instancias: se utilizan para representar objetos determinados de un concepto.
- Axiomas: son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología. Por ejemplo: "Si A y B son de la clase C, entonces A no es subclase de B", "Para todo A que cumpla la condición C1, A es B", etc.

Lenguajes para uso de Ontologías

Para poder desarrollar modelos ontológicos es necesario el uso de lenguajes estandarizados que proporcionan un marco para la gestión, integración, compartición y reutilización de datos. Estos normalmente son lenguajes de marcado apropiados que representan el conocimiento de las ontologías. Actualmente, los estándares más usados son XML (Extensible Markup Language), RDF (Resource Description Framework) y OWL (Web Ontology Language) que pueden representar algunas facetas sobre conceptos de un dominio y permite, mediante relaciones taxonómicas, crear una jerarquía de conceptos; ayudando a representar los conocimientos que contienen las ontologías.

XML (Extensible Markup Language). De acuerdo a W3C "es un Lenguaje de Etiquetado Extensible muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos" (Secc. ¿Qué son las Tecnologías XML?). Fue creado con el objetivo de poder compartir datos a través de la WEB, por medio de datos estructurados, uniformes e independientes de las aplicaciones; permitiendo definir nuevos lenguajes de presentación, y la creación de metadatos, etiquetas y estructuras propias; describiéndolas por medio de DTD (Document Type Definition).

RDF (Resource Description Framework). Marco de Descripción de Recurso

(RDF) es un lenguaje general para representar información en la Web; en resumen, no es más que una descripción conceptual. Lamarca (2009) amplia este concepto, diciendo sus ventajas ya que ofrece una arquitectura de metadatos para la Web que supone un gran avance para construir la infraestructura de la Web Semántica. RDF fue diseñado para soportar la reutilización y el intercambio de vocabularios. En realidad, RDF es una capa adicional sobre XML que permite simplificar la reutilización de términos de vocabularios a través de namespaces (espacios de nombre).

OWL (Web Ontology Language). Para W3C (2004) "OWL está diseñado para ser utilizado por aplicaciones que necesitan procesar el contenido de la información en lugar de presentar la información a los seres humanos" (Secc. Abstract). OWL utiliza los identificadores URI y RDF para agregar los atributos clave de las ontologías que les permiten ser utilizados en los sistemas que interactúan con la web semántica. Este describe las propiedades y clases, y las relaciones entre estas clases. El concepto de un marco de información compatible con OWL que comparte atributos tales como apertura, escalabilidad y extensibilidad determinan las bases para una ontología de dominio.

En este sentido, OWL añade un vocabulario para describir propiedades y clases; las relaciones entre clases y la cardinalidad. También se ha diseñado para su uso en aplicaciones que necesitan procesar el contenido de la información en lugar de presentar la información al usuario final; siendo este, un mejor mecanismo de interpretabilidad de contenido web que los mecanismos admitidos por XML, RDF, y esquema RDF (RDF-S), proporcionando vocabulario adicional junto con una semántica formal. OWL tiene tres sublenguajes, con un nivel de expresividad creciente: OWL Lite, OWL DL, y OWL Full.

- OWL Lite, suficiente para los usuarios que tan sólo piden posibilidades de clasificación en la jerarquía de conceptos (clases) de la ontología y restricciones simples.
- OWL DL (Description Logic) es el lenguaje indicado para los usuarios que requieren el máximo de expresividad pero exigiendo completitud computacional y

decibilidad. Incluye todos los constructores de OWL, pero sólo se pueden usar con restricciones.

- OWL Full se dirige a aquellos usuarios que necesitan la máxima expresividad y la libertad sintáctica de RDF pero sin garantía computacionales. Permite, por ejemplo, aumentar el significado de vocabulario predefinido (en RDF o en OWL), por lo que es muy improbable que ningún software de razonamiento sea capaz de soportar razonamiento completo para cualquier característica de OWL Full.

Metadatos

Con esta gran diversidad y cantidad de información que se encuentra contenida en La Web, se hizo necesario mecanismos que pudieran de alguna manera estandarizarla para el entendimiento de su extracción, y por ello se establecieron métodos para etiquetar, clasificar, ordenar y describir dicha información; a esto se refieren los Metadatos.

Un metadato no es más que un dato que posee una estructura definida y estandarizada sobre la información. Su significado literal es información sobre información, o datos sobre datos. Según Lamarca (2009) “Los metadatos en el contexto de La Web, son datos que se pueden guardar, intercambiar y procesar por medio del ordenador y que están estructurados de tal forma que permiten ayudar a la identificación, descripción, clasificación y localización del contenido de un documento o recurso web y que, por tanto, también sirven para su recuperación” (*Sección:* Metadatos, párr. 2).

De acuerdo a Lamarca (2009), concluye que hay que darle un sentido más exacto a esta definición, debido a que los metadatos solo son posibles dentro de La Web, ya que acá es donde pueden ser usados de acuerdo a la función que los caracteriza. Berners-Lee (1997, citado por González y Sadier, 2006) da una definición concreta de los metadatos en este contexto digital "Los metadatos son información inteligible para el ordenador sobre recursos Web u otras cosas" (p. 3).

Hay varios modelos de metadatos que se usan en la actualidad, cada uno posee

su propia estructura y descripción; en todos ellos, cada objeto se describe por medio de atributos y el valor que toman dichos atributos son los que al final ayudan para recuperar información.

Los metadatos al ser datos, pueden también ser almacenados en una base de datos con una referencia al documento completo, haciendo una especie de relación de índice. Los motores de búsqueda generalmente hacen uso de estos metadatos para caracterizar las páginas encontradas por el Crawler.

Lamarca (2009), resalta la importancia del uso de los Metadatos “Las grandes ventajas del uso de metadatos radican en que se usa el mismo contenido del documento como un recurso de datos y que los metadatos valen también para recursos que no tienen únicamente la morfología de texto, sino para cualquier tipo de morfologías tales como vídeo, audio o imágenes" (*Sección:* Metadatos, párr. 6).

A pesar de que uno de los objetivos que persigue el uso de metadatos es la estandarización de la información para su fácil recuperación; aun se están haciendo esfuerzos para normalizarlos; muchos organismos han dedicado tiempo para lograrlo; el ejemplo tal vez más significativo de ello es Dublin Core, creado por las iniciativas de las asociaciones de bibliotecarios norteamericanos, y en concreto por la Online Computer Library Center (OCLC).

Clasificación de los Metadatos

Se han establecido muchas maneras de clasificar los Metadatos, dependiendo de aspectos como su forma, funcionalidad, nivel de estructuración de datos, persona o entidad que lo origina, etc. De forma general según Lamarca (2009) se pueden dividir en los siguientes grupos de acuerdo al dominio en que apliquen:

1. Metadatos para describir recursos de información en La Web.
2. Metadatos para la descripción archivística.
3. Metadatos para la descripción museística.

4. Metadatos para definir registros catalográficos en bibliotecas y centros de documentación.

5. Metadatos para recursos geográficos y espaciales.

6. Metadatos para describir recursos de información gubernativa y administrativa.

Sin embargo Rodríguez (2002, citado por Lamarca, 2009), ofrece una visión mucho más detallada y precisa de las tipologías existentes, donde se basa en la riqueza estructural, semántica y en su complejidad, agrupan los metadatos y los esquemas de metadatos en 3 tipos que denominan bandas o zonas (*Ver Figura No. 25*)

	BANDA 1		BANDA 2		BANDA 3	
Características	Formatos simples Sistemas propietarios Indización a texto completo		Formatos estructurados Estándares de facto Estructura de campos		Formatos ricos Estándares internacionales Etiquetas elaboradas	
Formatos	Lycos AltaVista Yahoo, etc.	Etiquetas **<META>** en HTML	DC/DCMI IAFA RFC 1807 SOIF LDIF	Edna AGLS etc.	ICPSR CIMI EAD TEI MARC	DC-AP DIG35 IMS etc.

Figura No. 25. **Tipos de metadatos en bandas (basado en Dempsey y Heery).** Méndez (2002, citado por Lamarca, 2009)

Se puede observar la clasificación realizada por Méndez. Estas bandas o zonas permiten agrupar los metadatos; irían desde una menor riqueza y complicación (banda 1) hasta una mayor riqueza y complejidad en la descripción (banda 3). Además, esta tipología englobaría tanto aspectos referidos a los atributos de los metadatos, como a los distintos modelos de metainformación que comportan.

Asignación de Metadatos

Para asignar metadatos, existen varios modos de asociar metadatos con recursos digitales; Lamarca (2009) sugiere tres formas:

1. Incrustando los metadatos dentro del propio documento: Esto implica que los metadatos deben ser creados al mismo tiempo que se crea el recurso. Generalmente se almacenan dentro de la cabecera del documento y eso permite que sea transportada a la vez que se transporta el contenido del documento.
2. Asociando los metadatos: por medio de archivos acoplados a los recursos a los que describen. La ventaja de los metadatos asociados se deriva de la facilidad relativa de poder manejar los metadatos sin cambiar el contenido del recurso en sí mismo.
3. Metadatos independientes: los metadatos se mantienen en un depósito separado, generalmente una base de datos mantenida. De esta forma, es mucho más fácil gestionar tanto los metadatos como los recursos.

Esquema

Según Lamarca (2009) "un esquema (schema) es, simplemente, un vocabulario compartido procesable por máquina y que los esquemas proveen un significado para definir la estructura, contenido y semántica de un documento, esto es un esquema está formado por los elementos y reglas que constituyen un modelo de metadatos" (*Sección:* Metadatos, párr. 33).

Existe cierta confusión entre los términos ingleses schema y scheme ya que se traducen de formal igual en castellano; pero estos tienen distintos significados. Un schema es un modelo de metadatos formado por los elementos y reglas que lo constituyen y un scheme es una lista de valores posibles que puede contener una metaetiqueta concreta.

CAPITULO III

MARCO METODOLÓGICO

Existen muchas formas de investigar; desde siempre el ser humano se ha inclinado por descubrir nuevas cosas, investigar el porqué de muchos eventos, etc.; de manera que ha descubierto anomalías, fenómenos o casos raros que surgieron en dichas investigaciones; y es que al seguir su inquietud y luego de un trabajo de investigación, el científico analiza grandes interrogantes, algunas de utilidad para la humanidad, otras más teóricas, que han impulsado el conocimiento general.

Durante este proceso todo investigador enfrenta errores y va mejorando sus métodos y técnicas, trabajando de manera sistemática. Desde luego, en el 1998, Münch y Ángeles (citado en Guardo y Pentón, 2008) define la ciencia como "un conjunto sistemático de conocimientos con los cuales, al establecer principios y leyes universales, el hombre explica, describe y transforma el mundo que lo rodea" (p. 5); en donde dicho mundo es real y suele ser observable y en algunos casos medibles, obteniendo los resultados a través del método científico; que es la base de toda investigación

Así, todo estudio requiere de la definición de la metodología a ser empleada para su desarrollo, debido a que se considera que es la base de todo proyecto de investigación. En consecuencia, el marco metodológico de la presente investigación que desarrolla un sistema de búsqueda, basados en crawler, que permita el rastreo de información en repositorios confiables, enmarcado en una arquitectura SOA; se definen los aspectos metodológicos relativos al tipo de estudio y su diseño de investigación, incorporados en relación a los objetivos establecidos, las técnicas empleadas en la recolección de los datos, incluyendo sus características.

Con la finalidad de que esta investigación se realice de una forma sistemática y

reúna las características de organización y validez necesaria, se procede a enmarcarla en los protocolos establecidos para la investigación científica.

Naturaleza de la Investigación

Altuve y Rivas (1998) asegura que el diseño de una investigación, "... es una estrategia general que adopta el investigador como forma de abordar un problema determinado, que permite identificar los pasos que deben seguir para efectuar su estudio" (p. 231). Para ello, el trabajo se enmarca dentro de la modalidad de Proyecto Especial; porque el producto de su desarrollo coadyuvará en la búsqueda de contenidos de calidad en La Web, a fin de que docentes, estudiantes, investigadores y usuarios en general puedan satisfacer sus necesidades de búsqueda de información, estando confiados en la calidad de los recursos encontrados por la aplicación.

Al respecto de esta modalidad, el Manual de Trabajos de Grado de Maestría y Tesis Doctorales de la Universidad Pedagógica Experimental Libertador (UPEL) (2006), define el proyecto especial como: "Trabajos que lleven a creaciones tangibles, susceptibles de ser utilizados como soluciones a problemas demostrados, o que respondan a necesidades e intereses de tipo cultural". (p. 14).

Del mismo modo, la Universidad Centroccidental "Lisandro Alvarado" (UCLA) (2002), en su Manual para la Elaboración del Trabajo Conducente a Grado Académico de Especialización, Maestrías y Doctorado, la presente investigación se ubica en la modalidad de Estudios de Proyectos, el cual consiste en "...una proposición sustentada en un modelo viable para resolver un problema práctico planteado, tendente a satisfacer necesidades institucionales o sociales y pueden referirse a la formulación de políticas, programas, tecnología, métodos y procesos" (p. 63).

De esta manera, todo proyecto especial debe incluir la demostración de la necesidad de la creación o de la importancia del aporte; por lo que la presente metodología se apoya, en su fase de análisis, en una investigación documental. Así como lo afirma Cázares, Christen, Jaramillo, Villaseñor y Zamudio (1980), "La

investigación documental depende fundamentalmente de la información que se recoge o consulta en documentos, entendiéndose este término, en sentido amplio, como todo material de índole permanente, es decir, al que se puede acudir como fuente o referencia en cualquier momento o lugar, sin que se altere su naturaleza o sentido, para que aporte información o rinda cuentas de una realidad o acontecimiento", (p. 18).

El Manual de la UPEL, establece que una investigación documental, de acuerdo a los objetivos del estudio o de la temática, pueden ser:

> ...Revisiones críticas del estado del conocimiento: integración, organización y evaluación de la información teórica y empírica existente sobre un problema, focalizando ya sea en el progreso de la investigación actual y posibles vías para su solución, en el análisis de la consistencia interna y externa de las teorías y conceptualizaciones para señalar sus fallas o demostrar la superioridad de unas sobre otras, o en ambos aspectos... (p. 13).

En este sentido, se evaluarán los distintos métodos y tecnologías de las maquinas de búsqueda, así como los problemas que subsisten en este entorno, definido en el planteamiento de la presente investigación; comparando dichos métodos, arquitecturas y tecnología para proponer una solución; apoyándose en estándares reconocidos.

Método de Recolección de Información

En la presente investigación, es imprescindible el uso de herramientas o instrumentos que permitan una recolección de datos efectiva y eficaz, que garantice una base sólida para el análisis y el éxito del proyecto. Por cuanto, en el desarrollo de esta investigación es necesario utilizar herramientas que permitirán recolectar el mayor número de información necesaria, con el fin de obtener un conocimiento más amplio de la realidad.

No obstante, la metodología aplicada permitirá evaluar los siguientes aspectos relacionados con los principales motores de búsqueda, entre estos:

- Estudio de los métodos actualmente usados por los buscadores.
- Determinación de los problemas asociados a la confiabilidad de la información encontrada, de la amplitud de búsqueda y la Web Invisible.
- Revisión de las Tecnologías de software usados para el desarrollo de motor de búsqueda como lenguaje de programación, base de datos, estructuras de almacenamiento.
- Identificación de las oportunidades de mejora.
- Análisis de las arquitecturas y componentes usados.
- Inspección de ontologías asociadas al almacenamiento de estructuras de documentos no estructurados.
- Exploración de las tecnologías Web Services, beneficios de una implementación SOA y patrones de integración

Para ello, se aplicarán técnicas de recopilación documental, semántica documental y análisis de contenido; haciendo uso de recursos de documento escritos, como libros, revistas y tratados; documentos electrónicos como páginas web, revistas digitales, presentaciones y conferencias. Cuyo objetivo principal es el acopio de los antecedentes relacionados con la investigación, en donde para tal fin se consultaron documentos escritos formales que se tomarán como base y enfoque para la presente investigación; incluso se admitirán ideas, mecanismos, propuestas y software ya probados en investigaciones y/o proyectos anteriores.

Fases de la Investigación

A fin de cumplir con los requisitos involucrados en la modalidad, la cual se establece como Proyecto Especial y en función de los objetivos planteados en la investigación, se determinaron 4 fases para el estudio. Adicionalmente, como el presente trabajo establece el desarrollo de un software; entendiendo que dicho proceso es por naturaleza creativo y que la ingeniería del software trata de

sistematizar este proceso con el fin de limitar el riesgo del fracaso en el logro de los objetivos, por medio de diversas técnicas que han sido estudiadas y evaluadas. Por tal motivo las fases de la investigación irán acopladas bajo los conceptos y principios de esta disciplina; tomando un modelo de desarrollo iterativo y creciente ágil como Extreme Programing (XP); con el objeto de proveer un resultado de calidad.

Fase I: Fase de Análisis

En esta fase se desarrolla el conocimiento de la situación existente en la realidad objeto de estudio, a fin de describir la plataforma tecnológica y las funcionalidades requeridas por el sistema de motor de búsqueda, apoyado en material bibliográfico; a través de un proceso sistemático de indagación, recolección, organización, análisis e interpretación de información.

El propósito de esta fase es el de analizar la situación existente de la problemática de la Web Invisible, los métodos de búsquedas actuales, tecnologías de integración y servicios web existentes para así, a través de una análisis orientado a servicio, describir el sistemas y elicitar los requisitos funcionales y no funcionales del motor de búsqueda; para ello haciendo una especificación de requisitos, por medio de los casos de uso y metáfora del sistema, tal como lo especifica la ingeniería de requisitos.

Este proceso se realizará documentado en publicaciones realizadas en revistas, conferencias, especificaciones, reportes técnicos, tesis de grado y casos de éxitos; lo que ayudará en el diseño de la arquitectura, la elección de las implementaciones de software a usar; como también algunas ideas de cómo atacar el problema de La Web Invisible y de pasos a seguir para implantar un sistema bajo enfoque SOA con un ESB.

Fase II: Fase de Diseño de Arquitectura

En esta segunda etapa, tendrá como insumo el resultado de la fase de análisis, ya que dará una información amplia y concisa de las características, recursos disponibles, descripción y requisitos del sistema.

Se determinaran los componentes y servicios de la máquina de búsqueda, siguiendo los pasos propuestos por Papazoglou y Van Den Heuvel, a través de un diseño orientado a servicios, donde el mismo es acoplado en un middleware de integración. También se seleccionarán los atributos de los documentos descargados por el proceso de crawling que se guardarán en el almacén de datos, tomando como base estándares y ontologías para uso de metadata.

La meta en esta fase es la de realizar el diseño de la arquitectura orientada a servicio del motor de búsqueda, teniendo presente los componentes y servicios seleccionados, especificando su interacción y comunicación; usando patrones de arquitectura e integración; apoyado en diagramas de secuencia y despliegue. Además, tomando en cuenta las propuestas en los trabajos consultados, como las evaluaciones realizadas por producto de software; se hará la selección de la tecnología a usar. Incluso, se hará el diseño de las propiedades que pueden determinar la confiabilidad de un repositorio, apoyado en trabajos realizados por organizaciones.

Fase III: Fase de Implementación del Motor de Búsqueda

Comprende la construcción de los componentes de la arquitectura, contemplados en la fase anterior, haciendo uso de patrones de desarrollo. En esta fase se evidencia la importancia de las dos fases anteriores, según lo determina la ingeniería de software, con un levantamiento de requisitos apropiado y un buen diseño de la aplicación, existe menos probabilidad de error al momento del desarrollo y de re trabajo.

La implementación de cada componente se desarrollara conforme al paradigma orientado a servicio, bajo los principios de bajo acoplamiento, reutilización e

interoperabilidad; para ello usando tecnología REST Services e implementando los mecanismos de integración basado en mensajes; conectando un bus de integración, empleando Mule como framework. Asimismo se realizarán las interfaces de administración y de búsqueda; siendo los mecanismos de interacción entre el sistema y los actores. Los diagramas de interacción, clases y actividad serán muy importantes en esta etapa.

Como se desarrollará bajo metodología XP, se establecerán las iteraciones para el desarrollo de cada componente, haciendo pruebas al final de cada iteración y validando con los requisitos de cada uno. Con este enfoque se reduce al máximo los posibles errores y fallas que pueda tener el producto final.

Fase IV: Fase de Validación Tecnológica

En esta última etapa de la investigación; se evaluará tecnológicamente el producto desarrollado; esto se elaborará a través de métodos de prueba de software de verificación y validación, desarrollados como etapa final en las metodologías de ingeniería de software. Con estas técnicas se puede determinar si el resultado final cumple con los requisitos y diseño estipulados en las dos primeras fases. Esta viene a comprender la última iteración de la metodología XP, antes de la “muerte del proyecto”.

Adicionalmente, esta fase comprende, una etapa de ajustes, ya que al encontrar algún error o falla; se procede a repararlo haciendo un proceso de retroalimentación; hasta que el requisito quede cumplido. En último lugar se establecerán pruebas comparativas utilizando ámbitos de búsqueda para el sistema; cotejando los resultados con dos buscadores tradicionales, teniendo en cuenta la confiabilidad del contenido; haciendo uso de los criterios evaluativos que surjan de la fase de análisis.

CAPITULO IV

PROPUESTA DEL ESTUDIO

Introducción

Toda actividad de creación de software, necesita ser organizado y distribuido en actividades para el cumplimiento de las necesidades que indujeron su desarrollo; para ello se han establecido una serie de modelos de desarrollo de software que ayudan en esta tarea y por ende logran obtener un producto de calidad. Estos modelos, se centran en definir procesos de desarrollo racionales y controlables, además de imponer un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente; trayendo consigo un patrón a seguir el cual puede ser adaptado a las necesidades del proyecto. Ninguno de estos modelos pretenden ser único y universales; existen múltiples variantes, por lo que representan una guía de cómo y en qué orden deben realizarse cada una de las actividades, las cuales establecen el ciclo de vida del software.

El producto final de la presente investigación será un software que realice búsqueda de información a través de enlaces de la web, usando “robots” comúnmente llamados Crawler, inicializado con una serie de repositorios confiables como punto de partida; construido bajo un enfoque SOA. Para el cumplimiento del mismo es necesario adoptar un modelo de desarrollo de software que brinde herramientas y técnicas para su elaboración; estructurando las actividades a seguir. En este sentido, se plantea usar el modelo de desarrollo eXtreme Programming o Programación Extrema (XP).

XP es una metodología de desarrollo catalogada como Ágil y es adecuada para

equipos de desarrollo pequeños, diseñada principalmente para responder a los cambios y lograr entregas continuas, rápidas y de valor. Esta metodología establece un ciclo de vida que se utilizará para dirigir el planteamiento y el desarrollo de la propuesta.

El modelo XP, estable seis etapas del desarrollo: Exploración, Planificación de la Entrega de una Versión, Iteraciones, Producción, Mantenimiento y Muerte del Proyecto. En esta investigación se realizarán las fases de exploración, planificación de entregas y las iteraciones necesarias para lograr el producto de software que se quiere, no se implementaron las fases de producción y mantenimiento.

Fase I: Fase de Análisis

En esta etapa, además de realizar un análisis de la problemática y la situación actual de los buscadores; de acuerdo a la metodología adoptada XP, se desarrollarán las fases de Exploración y de Planificación de las Entregas; para lo cual, se procedió a realizar una descripción del sistema y recoger las historias de usuario como herramientas para el descubrimiento de requisitos; que en este caso serán las historias de los procesos del motor de búsqueda, ya que el mismo no tiene un campo de aplicación donde intervienen usuarios finales y se basó en el análisis orientado a servicio, a fin de determinar los requerimientos y tareas.

La Web Invisible

La Web Invisible es un fenómeno que influye como una de las principales causas de la dificultad para encontrar información de calidad en La Web. Autores como Bergman (2000) y Shestakov (2008) en la Web Invisible se encuentran muchos más recursos de calidad que en la Web Superficial, siendo esta última aquella que puede ser indexada por los buscadores más populares; al contrario de la Web Invisible cuyos recursos no son indexados o son mal indexados por los motores de búsqueda.

Interesante son los resultados mostrados por BrightPlanet (2011), donde encontraron que: “La calidad total del contenido de la Web Invisible es al menos 1000 hasta 5000 veces mayor que la de la Web Superficial. El contenido de la Web Invisible es altamente relevante para la información que se necesita” (p. 3); también aseveran que: “El 95% de la Web Profunda es de información pública accesible” (p. 3). Se puede concluir con estos datos, aunado aquellos presentados en el planteamiento del problema, que dentro de la Web Invisible existe una gran cantidad de datos confiables que no pueden ser alcanzados por los motores de búsqueda tradicionales. En este sentido es interesante la aseveración que hace Gruchawka (2005) “Hoy en día los motores de búsqueda son herramientas maravillosas; pero sin embargo sus búsquedas frecuentemente producen más basura que tesoros” (p. 5).

Gruchawka (2005), realizó un cuadro comparativo resumiendo los resultados obtenidos por Bergman (2001), que respalda el hecho de que la información contenida en la Web Invisible es evaluada por expertos, al contrario de la Web Superficial, como se puede observar en la *Cuadro No. 9*.

Cuadro No. 9
Comparaciones entre la Web Superficial y Web Invisible

Web Superficial	**Web Invisible**
1 Billón de Documentos	550 billones de Documentos
19 Terabytes	7750 Terabytes
Cobertura Superficial Amplia	Cobertura Vertical Profunda
Resultados Contienen Publicidad	Resultados No Contienen Publicidad
Contenido No Evaluado	Contenido Evaluado Por Expertos

***Nota*.** Tomado de Using The Deep Web: A How-To Guide for IT Professionals. Gruchawka (2005).

Se pueden observar otros datos sobre el contenido de la información de la Web Invisible; como es, que estas no tienen publicidad, además de la diferencia en cuanto a la cantidad de documentos contenidos que es considerablemente amplia.

Es significativo resaltar los casos de estudio realizado por Gruchawka (2005), en la cual evaluó dos búsquedas con las palabras claves “network security” y “PDA security”; para ambos casos usó las herramientas de Google y un buscador llamado Educause; los resultados fueron que usando Google generó millones de entradas, de las cuales se tomó el tiempo en revisar solo los 100 primeros. En ellos, encontró que muchos sitios eran de vendedores y solo una docena de sitio podrían tener buena información. El tiempo que empleó fue de 30 minutos, adicionado al tiempo que pudo emplear al verificar si esa docena de sitios era realmente de información confiable.

Al contrario los resultados obtenidos usando Educause, generó 2620 entradas, de las mismas también tomó las primeras 100, y resultó que ninguna eran de vendedores y tenían bastantes recursos en PDF que contenían información de calidad. Al final concluyó que los resultados obtenidos con Educause tienen más substancia y credibilidad.

Ahora bien, ¿por qué ocurre este fenómeno?; según Shestakov (2008) son tres las causas del mismo. Primero porque existe información totalmente privada. Segundo porque existe información contenida en bases de datos para lo cual se tendría que hacer una búsqueda a través de las interfaces que proveen dichos sitios; en este sentido, se han realizado algoritmos que ayudan a hacer búsquedas sin usar las mismas pasando la información vía URL, sin embargo algunos usan frameworks y establecen mecanismos de seguridad que no permiten estos métodos. Y por último, existen aquellos sitios cuya información no es referenciada o es muy poco referenciada por otras páginas, al igual que dichas páginas no referencian o hacen poca referencia a otras páginas.

Clasificación Actual de los Motores de Búsqueda

Todos los buscadores funcionan de la misma manera a nivel general; donde a partir de un conjunto de semillas iniciales comienzan a operar iterativamente, es decir, extraen y procesan el contenido de estos sitios iniciales, encontrando allí nuevos enlaces a los cuales en la siguiente iteración nuevamente el componente Crawler extraerá y procesará su contenido encontrando más enlaces para la siguiente iteración y así sucesivamente. Estos se diferencian dependiendo de algunos métodos que les permiten cumplir su objetivo, los cuales se describen a continuación.

Por Frecuencia de Actualización

Tomando en cuenta la frecuencia de actualización de la información extraída; existen Motores de Búsqueda Periódicos e Incrementales. El Periódico es aquel donde se establecen políticas de extracción y actualización, es decir, se debe establecer cuando parar el proceso de extracción y luego comenzar a operar la actualización del contenido ya almacenado; el problema es que el proceso de actualización tarda mucho desde la última vez que el contenido fue extraído, por ello habrá más probabilidad que el contenido almacenado esté obsoleto.

A diferencia del Incremental, donde el componente Crawler en cada iteración además de operar sobre los sitios nuevos a extraer, también visita nuevamente el contenido ya extraído para su actualización. El problema se evidencia cuando en cada iteración el nivel de procesamiento se hace más y más grande y por ende más lento; sin embargo, la probabilidad de que el contenido extraído esté actualizado es mayor al del Periódico.

Existe una alternativa aportada por Cho y Garcia-Molina (1999), llamada Crawler Activo, la cual es una modificación del método Incremental. Los autores establecen que el contenido almacenado de las páginas rastreadas debe tener

adicionalmente un campo de ranking o llamado PageRank para determinar la frecuencia de actualización de cada página; por lo tanto en cada iteración adicional a la extracción del contenido de los nuevos enlaces encontrados, el buscador actualizará también las páginas ya encontradas, solo que no todas, sino solo las que cumplen los criterios del ranking, también en cada iteración evaluará y actualizará dicho ranking.

Esta propuesta será tomada en cuenta para la construcción del sistema, ya que además, Cho y Garcia-Molina, en su trabajo hicieron una prueba de campo, la cual arrojó mejores resultados.

Por Método de Extracción

De Acuerdo, al método de extracción de información existen los motores de búsqueda paralelos; aquellos que proveen paralelismo al momento de hacer la búsqueda en cada página y los secuenciales, donde en cada iteración espera por el resultado de la búsqueda de una URL para comenzar con la próxima en la cola.

Castillo, Marin, Rodríguez y Baeza-Yates (2004), realizaron un trabajo donde concluyeron que los procesos de extracción en paralelo son más eficientes por tanto más rápidos; la desventaja de este método es que se requiere de un buen ancho de banda; por ello, se debe considerar el número de tareas que estarán paralelas. Por otra parte es necesario considerar que no se debe extraer información paralela en un mismo sitio Web ya que satura el ancho de banda del mismo.

En tal sentido, este método se tomará en cuenta para la elaboración del producto. Cabe destacar que también se debe seleccionar un lenguaje de programación capaz de soportar el paralelismo de tareas de una manera eficiente.

Por Objetivo de Búsqueda

En cuanto, al objetivo de búsqueda del motor existen los Globales, y los Dirigidos o Enfocados. El Global, es aquel cuya amplitud de búsqueda es toda la Web,

es decir, tratan de buscar todos los enlaces que puedan encontrar en todo el Internet; a diferencia del Dirigido que solo ejecuta la tarea de extracción en los diferentes sitios para los cuales se configuró. Simplemente se tiene que definir unos filtros de enlaces, que restrinjan la descarga de documentos que pertenezcan a un nombre de dominio concreto.

Así, podemos observar en la *Figura No. 26*, que el objetivo del estudio no es abarcar todo el contenido de la Web Superficial; haciendo una analogía, no es tirar una red y arrastrar hasta conseguir la mayor cantidad de peces posibles.

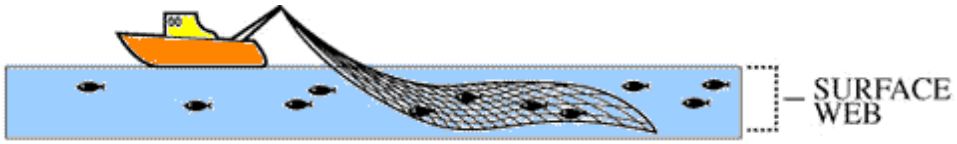

Figura No. 26. **Web Superficial**. Tomado de Bergman (2001).

Más bien, el objetivo es enfocarse en sitios cuyo contenido sea confiable, por lo tanto se desarrollará el concepto de Motor de Búsqueda Dirigido. En la *Figura No. 27*, se detalla como el fin será pescar con anzuelo aquellos peces “confiables” que en su mayoría están en la Web Invisible.

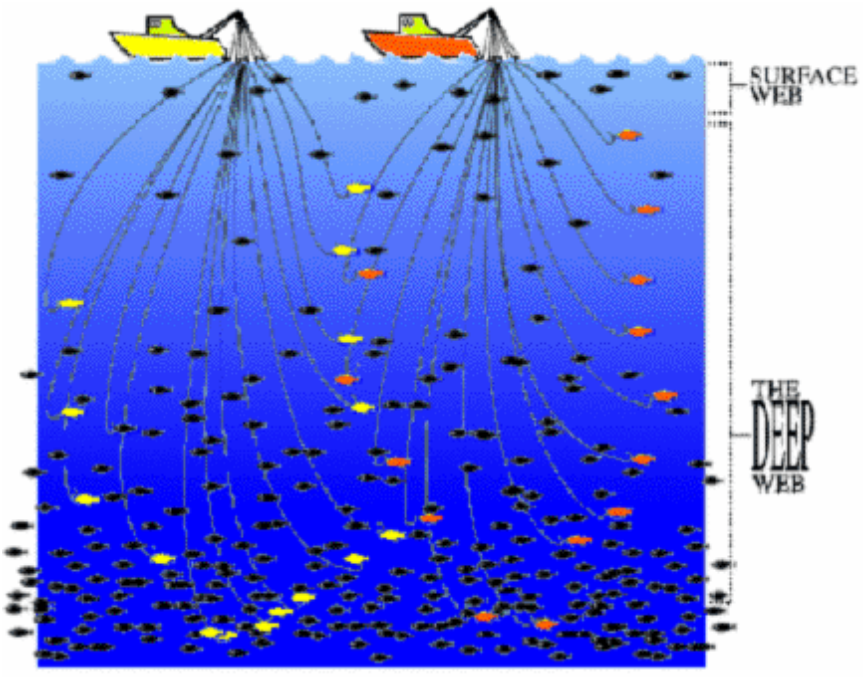

Figura No. 27. **Web Invisible**. Tomado de Bergman (2001).

Componentes del Motor de Búsqueda

Tal como se destacó en los antecedentes, la arquitectura propuesta por M. Álvarez (2007), será tomada como base para el proyecto; la misma se puede observar en la *Figura No. 2*. A continuación se describen sus componentes.

El Componente Gestor de Configuración es común a todos los módulos, y contiene la información de arranque del buscador y la definición de los dominios de aplicación. Existen otros componentes que sirven de unión entre los distintos módulos, como son el Componente Repositorio y el Componente Índice. En tiempo de ejecución, se distinguen tres fases, cada una correspondiente a uno de los módulos principales. Las dos primeras se corresponden con la etapa de obtención de información del sistema y la última con la de ejecución de consultas.

La operación de los distintos componentes de dicha arquitectura, es de la siguiente manera: cuando el motor de búsqueda comienza su ejecución, lee la configuración del componente Gestor de Configuración. El siguiente paso consiste en

inicializar la Frontera con la lista de sitios iniciales, así como la inicialización del conjunto de procesos.

El Componente Frontera es el responsable de mantener la lista global de rutas a ser accedidas, y que es compartida por todos los procesos. Una vez que se han inicializado todos los procesos, cada uno de ellos obtiene una ruta de la Frontera. Luego, el componente de crawling carga el objeto de sesión asociado a la ruta y descarga el documento asociado (utiliza el gestor de navegación para seleccionar el manejador adecuado para el documento, como PDF, MS Word, etc.). Como salida del componente de crawling se obtiene el documento modelado.

El Componente Gestor de Contenido analiza el documento aplicando una cadena de filtros para verificar si el documento posee errores. El clasificador de rutas selecciona y prioriza las rutas que deben ser añadidas a la frontera para ser accedidas en el futuro. Adicionalmente, también posee filtros para almacenar documentos con su meta-información. El proceso de extracción del motor de búsqueda finaliza cuando no quedan rutas en la Frontera.

A partir de la información contenida en el Componente Repositorio, el módulo de indexación recorre los diferentes documentos. En todos los casos se indexan esos contenidos en el componente Índice y el componente de estructuración automática obtiene los diferentes registros de datos contenidos en la página y los almacena en un índice multicampo para permitir la realización de consultas estructuradas sobre él.

Por último, el componente de búsqueda permite la realización de consultas sobre el índice de documentos, para traer los registros de datos que han sido identificados en las diferentes páginas de resultados.

Para este último componente, se tomará en cuenta el diagrama de obtención de resultados propuesto por Suárez y Plasencia (2010); que se puede observar en la *Figura No. 3*, de los antecedentes.

Almacén de Datos

Es necesario para el sistema, realizar una selección de un almacén de datos donde se guardarán todos los documentos recuperados por el proceso de crawling dentro del buscador; esto permitirá aplicar procesos de Recuperación de Información (IR), lo cual dará rapidez en el proceso de consulta.

Para determinar que Almacén de Datos a usar, primero hay que establecer la estructura de datos de sistema más conveniente, así como el modelo y el tipo de objeto a guardar; ya que de estos factores dependerá el tiempo que se emplea al manipular los datos.

Estructura de Datos de Sistema

De acuerdo a lo investigado en las Bases Teóricas; destacan aquellas estructuras de tipo de árbol y entre ellas los arboles B-Tree+, ya que una de sus grandes ventajas son sus métodos de inserción, actualización y eliminación, debido a que mantienen balanceado el árbol a diferencia de los tradicionales. El B-Tree+ siempre mantiene una forma piramidal gracias a su balanceo.

Tipos de Almacén de Datos

Por otra parte, hoy en día existen dos tendencias para la construcción de repositorios de datos; las relacionales y las no relacionales (NoSQL); por ello Bartholomew (2010) en su escrito SQL Vs NoSQL, manifiesta: “Ninguno es mejor que el otro, todo depende las necesidades de implantación y de los requerimientos” (conclusions, párr. 6). Sado que el objetivo del sistema es guardar los documentos extraídos de la Web, el modelo del mismo tendrá entidades y atributos donde no es necesario que estén fuertemente relacionadas entre sí; además que se requiere una

escalabilidad horizontal. Es importante considerar la rapidez de búsqueda de información, que debe ser reducida al máximo; por tal razón se recomienda el uso de una Base de Datos NoSQL; puesto que al no existir relaciones, el sistema no consume tiempo al realizar joins y validaciones de restricciones.

En otro orden de ideas, el motor de búsqueda guardará documentos con estructuras variables; en este sentido, existen dos almacenes NoSQL principales orientadas en este ámbito, MongoDB y CouchDB; por ello, tomando en consideración la tesis de Seeger (2010) donde recomienda MongoDB de acuerdo a su investigación, se elegirá esta Base de Datos como almacén de datos. Además, que MongoDB usa como estructura un árbol B-Tree+; cuya eficiencia se analizó anteriormente; y finalmente usa una implementación de JSON llamado BSON para manipular los datos añadiendo serialización; esta característica será de mucha ayuda con la implementación del Bus de Integración.

Recuperación de Información

Existen dos modelos básicos para la Recuperación de Información (IR), que son usados. El primero de ellos es el Modelo Booleano; el cual considera los documentos descritos por un conjunto de términos de indexación, y una consulta puede ser cualquier expresión del álgebra booleana sobre los términos de indexación. El conjunto de documentos recuperados por el sistema serán aquellos documentos cuyos términos de indexación cumplen la consulta.

El segundo es el modelo vectorial; el cual propone un marco en el que es posible el emparejamiento parcial a diferencia del modelo de recuperación booleano, asignando pesos no binarios a los términos índice de las preguntas y de los documentos. Estos pesos de los términos se usan para computar el grado de similitud entre cada documento guardado en el sistema y la pregunta del usuario.

Ambos métodos tienen sus pros y contras; hoy en día los indexadores de contenido, usan algunos de los dos modelos pero con variantes bastantes eficientes e

incluso variantes híbridas de los dos modelos.

Klas (2010), en su tesis de grado, hizo una evaluación de los siguientes indexadores: Apache Lucene, Hibernate Search y Compass, en el cual concluyó que los indexadores basados en Lucene son bastante eficientes; asimismo Lucene usa una versión híbrida del modelo booleano y vectorial, utilizando el modelo booleano para filtrar documentos y el vectorial para priorizarlos.

Teniendo en cuenta que se usará MongoDB como Almacén de Datos, se decidirá el uso del indexador ElasticSearh, debido a que, está basado en Lucene y permite la integración con bases de datos NoSQL. Un Aspecto interesante de ElasticSearch es que usa una API REST, y un lenguaje de consulta JSON, el cual va acoplado con la estructura del repositorio controlado por MongoDB, utilizando el mismo método. Cabe resaltar que REST según Rodríguez (2008) “ha emergido en los últimos años como un modelo de diseño de Web Services predominante; ya que REST, ha tenido mucho más impacto en la Web y ha desplazado a SOAP, principalmente porque es considerablemente un estilo simple de usar” (p. 1); adicionalmente indica que la principal evidencia de su dominio es que ha sido adaptada por Yahoo, Google y Facebook para el desarrollo de los servicios de la Web 2.0.

Selección de un Proyecto de Motor de Búsqueda

Considerando el análisis anterior de tipos y métodos de los motores de búsqueda, se procede a elaborar una distinción de acuerdo a las características resaltables, de una serie de implementaciones en software libre que se encuentran en la comunidad. Su código se tomará como inicio a la construcción del sistema del presente estudio.

Esta evaluación se realizó tomando en cuenta solo las características, en ningún momento se realizaron pruebas de rendimiento y corridas de los mismos. En la *Cuadro No. 10* se pueden ver la comparaciones.

Cuadro No. 10.
Comparación de Motores de Búsqueda Software Libre

Crawler	Incremental/ Periódico	Secuencial/ Paralelo	Global/ Dirigido	Almacén de Datos	Indexación	Índice	Lenguaje	Fecha Actualización
crawler4j	Incremental	Paralelo	Mixto	Berkely DB / SQL	NO	NO	Java	Marzo 2011
Heritrix	Incremental	Paralelo	Mixto	Berkely DB / SQL	NO	NO	Java	Mayo 2010
Nutch	Incremental	Paralelo	Mixto	Cassandra / NoSQL	Solr	Invertido	Java	Septiembre 2011
WebSPHINX	Periódico	Paralelo	Dirigido	Archivo	NO	NO	Java	Julio 2002
Aspseek	Incremental	Paralelo	Mixto	Aspseek-sql / SQL	Propio	Invertido	C++	Octubre 2002

Fuente. El Investigador.

En aras de seleccionar una de las implementaciones; de acuerdo a la tabla anterior se puede destacar que casi todas, tomando en cuenta las técnicas de los motores de búsqueda, cumplen con las características deseadas; sin embargo, Nutch aventaja, pues hace uso de NoSQL, lo que constituye una ventaja a la hora hacer la integración y reajuste con MongoDB; además, contrario a los demás, usa indexación basada en Lucene, a través del framework Solr; lo que también proporciona utilidad para la integración con ElasticSearch; y en última instancia el proyecto está bien soportado por una comunidad de desarrollo en Apache.

Por otra parte, son bien conocidas las ventajas del lenguaje Java, como: Portabilidad, Seguridad, Rendimiento, Robustez, etc.; sin embargo en este caso, es importante señalar la ventaja de Java con respecto al manejo de múltiples hilos; como lo indica Rose India Technologies (2006, Advantages of multithreading over multitasking) “las ventajas de Java en procesos multitarea-multihilos son: reduce el tiempo computacional, mejora el rendimiento de la aplicación, los hilos comparten el

mismo espacio de dirección lo que ahorra memoria, el cambio de contexto entre los hilos es generalmente menos costoso que entre los procesos y el costo de comunicación entre hilos es relativamente bajo"; por otra parte Java facilita la implementación de Web Services.

Caracterización de Nutch

A continuación se describe su arquitectura, funcionamiento y procesos; lo que coadyuvará en la descripción del sistema que se desea y la determinación de los requisitos. La arquitectura se puede ver en la *Figura No. 28*:

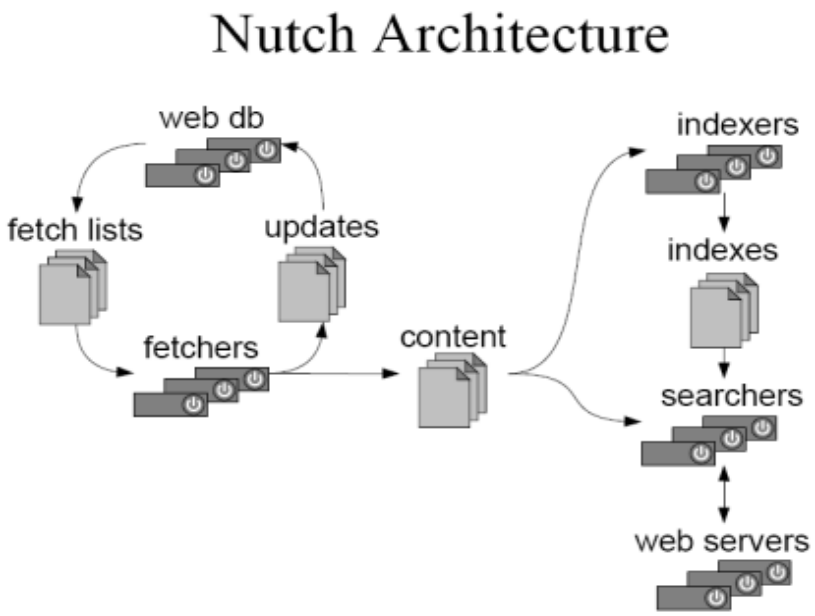

*Figura No. 28.***Arquitectura Nutch**. Tomado de Cutting (2004, p. 4).

El estilo arquitectural de los procesos centrales de Nutch, es de llamada y retorno, donde existe un programa principal y varios subprocesos que cooperan entre sí; en este caso, a través de un objeto configuración. El patrón usado para el desarrollo es IoC (Inversión de Control) que reduce el acoplamiento; característica que ayudará al desacoplamiento de los servicios.

Ahora bien, tomando en cuenta que Nutch delega el proceso de indexación al framework Solr, en otro nivel de abstracción la arquitectura es de tipo cliente servidor, como se observa en la *Figura No. 29.*

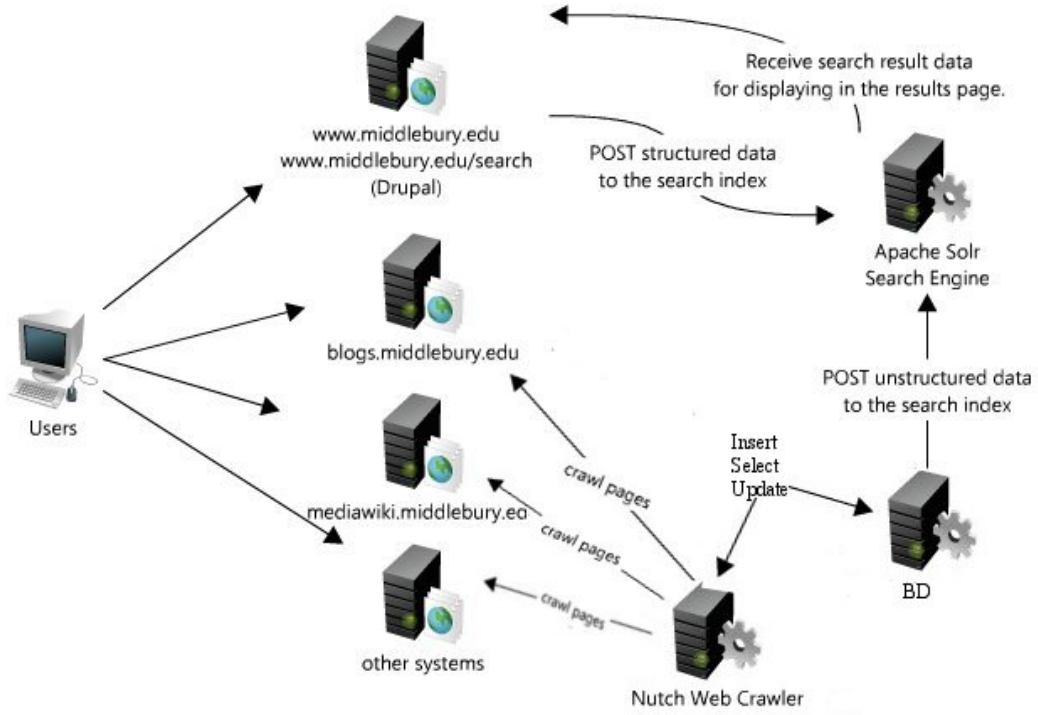

Figura No. 29. **Arquitectura Cliente Servidor Nutch**. Tomado de McBride (2010).

En la *Figura No. 29*, también se puede detallar como en un servidor (Nutch Web Crawler), están los procesos centrales de Nutch, realizando los procesos de extracción de URLs en la Web; estos interactúan con la Base de Datos que está en otro servidor. Luego se encuentra Solr, accesando a la Base de Datos para indexar el contenido y en último lugar, cualquier usuario a través de la interfaz de búsqueda accesa al servidor Solr para recibir los resultados según el criterio o palabras claves suministradas.

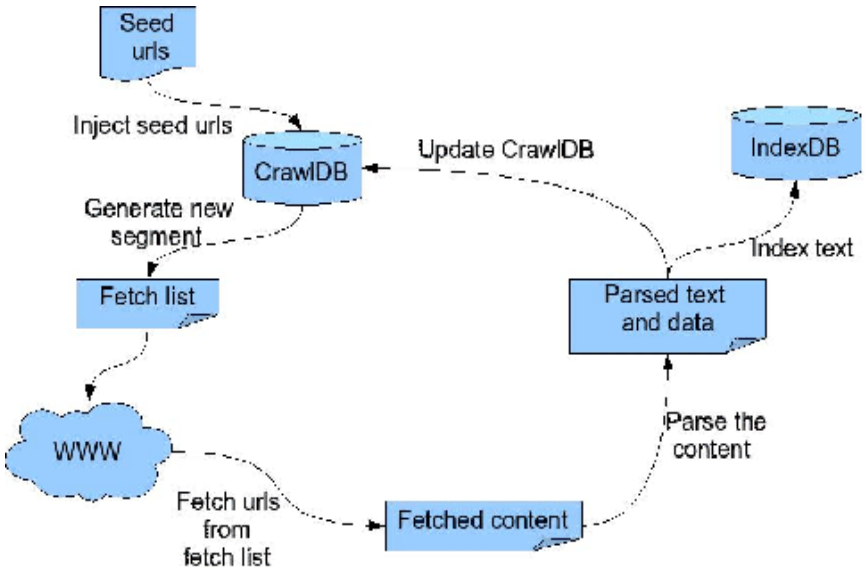

Figura No. 30. **Flujo de Trabajo de Nutch**. Tomado de Barbaria (2007, How Nutch Works).

En la *Figura No. 30*, se puede observar el flujo de trabajo de Nutch y también se puede percibir los tres procesos principales de los motores de búsqueda descritos en la investigación, que comprende: rastreo (crawling), indexación y búsqueda; y además parte de un conjunto de semillas.

Adicionalmente a los procesos básicos, Nutch posee otros procesos; cuyo flujo de ejecución se describe a continuación: el primer proceso es la Inyección de la semillas a la base de datos, en este proceso se instancia por cada semilla un objeto en donde todos sus atributos están vacíos excepto la clave primaria que corresponde al URL. Luego de esto comienza un ciclo que va a depender de la profundidad, dada como parámetro de configuración.

Por cada iteración se crea un segmento, en donde se almacena el cálculo de PageRank de todos los URLs, determinando la prioridad en que se hará el proceso de extracción de contenido de los URLs. Una vez establecido el orden, comienzan a ejecutarse en paralelo la extracción del contenido en la Web por cada URL. Una vez el contenido extraído y guardado en la Base de Datos, comienza a actuar el analizador (parser), cuyo objetivo es extraer los enlaces a otros sitios, los sitios que enlazan el contenido analizado y la metadata.

Una vez analizado todos los documentos, se actualiza la Base de Datos con los nuevos enlaces encontrados y allí comienza la nueva iteración. Paralelamente el indexador toma el contenido analizado para construir el índice y estar preparado para las consultas realizadas por el usuario final, usando la interfaz definida para ello.

SOA

Según Borja (2004, citado por la revista Computing, 2004); "SOA es una manera distinta de percibir la manera en que se comunican los componentes de una aplicación y la comunicación con otras aplicaciones externas, sin importar el lenguaje de programación en que están hechos" (p. 4). Por tanto, es necesario usar una guía o método a seguir para implementar SOA, es por ello, que se toma para el estudio la metodología propuesta por Papazoglou y Van Den Heuvel (2006). Según los autores esta metodología refleja un intento de acoplar principios del diseño y desarrollo que son aplicados tanto en los Web Services como en los procesos de negocio.

Existen estrategias para realizar el análisis y diseño orientado a servicio, se encuentran la estrategia de arriba hacia abajo o top-down, de abajo hacia arriba o bottom-up y la ágil o meet-in-the-middle. Cada una ofrece un enfoque diferente para realizar el trabajo con sus respectivos pros y contras y que de acuerdo a la naturaleza del proyecto que se va emprender, se debe seleccionar una como guía para la labor de análisis y diseño. Estas estrategias son integradas a la metodología de Papazoglou y Van Den Heuvel.

Para tener elementos que ayuden a la decisión de escoger una de las estrategias, se observa la *Cuadro No. 11*, comparativo de las tres opciones, publicado por Earl (2005).

Cuadro No. 11.
Comparación de Estrategias SOA

Estrategia	Dimensiones en el proceso de análisis y diseño				
	Calidad en la arquitectura	**Profundidad del Análisis**	**Utilización de Recursos (Tiempo y Dinero)**	**Tiempo de Entrega**	**Reusabilidad y Composición**
Top-Down	Alto	Alto	Alto	Alto	Alto
Bottom-Up	Bajo	Bajo	Bajo	Bajo	Bajo
Ágil	Alto	Alto	Medio	Medio	Alto

Nota. Tomado de Earl (2005, p. 73).

En este estudio lo deseable es obtener el mayor grado de calidad posible, por lo que la opción Bottom-up se descarta. Así también en el presente escenario, se dispone de una cantidad limitada de recursos para el estudio y se requiere un tiempo de entrega lo más reducido posible. La estrategia ágil surge como la opción a seguir, viendo que ofrece un alto grado en las características de calidad en la arquitectura, profundidad en el análisis, reusabilidad y composición, con una inversión de recursos menor con respecto a las demás estrategias. Aunado que concuerda con la metodología de desarrollo ágil XP.

Las fases propuestas por la metodología de Papazoglou y Van Den Heuvel son:

Fase de Planificación. En donde se determina la factibilidad, la naturaleza y alcance. En el estudio no se determinarán factibilidades por ser un proyecto especial, mas si el alcance y naturaleza del sistema.

Fase de Análisis. Durante el cual se investigan los requerimientos de la aplicación. Esta se cumplirá junto con la Fase I del estudio.

Fase de Diseño. Durante la misma los procesos y servicios conceptuales son transformados a un conjunto de relaciones e interfaces. Esta se cumplirá junto con la Fase II de la aplicación.

Fase de Construcción. Donde se implementan los Servicios Web. Esta se abarcara durante la Fase III de la Investigación.

Fase de Prueba. Donde se prueban el funcionamiento de los servicios y se determina si cumplen con los requerimientos. Esta se cumplirá con la última fase de la investigación.

Fase de Prestación de Servicio. En la cual se establecen los lineamientos para que otros entes hagan uso de los servicios.

Fase de Despliegue. Donde los servicios se despliegan en la arquitectura para ser accesados.

Fase de Ejecución de Servicios. En la misma, los servicios ya se encuentran en total operación y se debe verificar que la entrega de los servicios a los otros entes estén siendo accesados como debe ser.

Fase de Monitoreo. Consiste en monitorear y hacer mediciones de eficiencia y efectividad de los servicios provistos. Como la investigación no cubre la puesta en marcha del sistema; las últimas cuatro fases no serán cubiertas.

WebServices

Existen varios tipos de métodos por los cuales se puede implementar una SOA, como pueden ser: HttpServices, Mensajeria, Ajax (DWR), Jini, JavaSpaces y WebServices. Para que cualquiera de ellos puedan considerar servicios SOA, deben cumplir con estándares definidos como: corresponder a funcionalidades del negocio bien definidas (self contained), independiente de la implementación tecnológica e independiente de los sistemas operacionales que soportan (loose coupling), y deben ser compartidos y reutilizados por otros procesos o sistemas (reuse).

Tomando en cuenta que una SOA está basada en la implementación de servicios de negocio, lo que conlleva a un desarrollo de bloques funcionales que se pueden integrar y compartir en distintas aplicaciones; se decidió hacer uso de WebServices como herramienta de implementación de SOA; debido a que es la principal

tecnología usada, la que se recomienda y se promueve, en concordancia con Mahmoud (2005) que establece “Los Web Services es el estándar preferido para realizar una SOA” y Genkin (2007) afirma que “Web Services son un complemento natural para la construcción de soluciones SOA”.

Adicional, los WebServices es un estándar de facto que opera en la mayoría de los lenguajes de desarrollo, y es contemplado en una gran cantidad de soluciones de integración como bases de datos, ESB, ETL (Extract, Transform and Load), etc.; y Gartner (2010), sitúa a los WebServices en su investigación, como una tecnología en plena productividad.

Estilos de WebServices

Tal como se determinó en las bases teóricas, existen dos tipos de tecnologías de WebServices, aquellas basadas en SOAP y REST; ambas cumplen con las características básicas de los WebServices, pero sin embargo, existen algunas diferencias que son tomadas en cuenta para la elección de alguna de ellas. REST es la tecnología que en los últimos años ha tenido más auge; aunado a ello, se tomará en cuenta las diferencia entre ambos estilos, vista en la *Cuadro No. 6* y *Cuadro No. 7*. En tal sentido, el presente estudio se adoptará REST como estilo de WebServices debido a las siguientes consideraciones:

- La implementación es sencilla y fácil de usar.
- Los servicios no necesitan de un estado previo para su ejecución
- El productor y consumidor conocen el contexto y el contenido.
- REST se centra en la escalabilidad y rendimiento; necesario para que los procesos bases del sistema de búsqueda puedan ser desplegados en varios servidores.
- Tomando en cuenta que el proceso de crawling consume suficiente ancho de banda, REST consume poco ancho de banda e incluso puede ser usado en dispositivos móviles.
- Los datos que serán compartidos a través de los servicios es fácilmente

identificable como recurso.

Integración de Aplicaciones

Siguiendo los patrones de integración expuestos en las bases teóricas; los basados en mensajería son los más usados y más destacables según Hohpe y Woolf (2003); por ello será el patrón a tomar en cuenta en el desarrollo de la solución. Por otro lado, existen varios métodos de integración de aplicaciones, entre ellos ESB (Enterprise Service Bus), basado en patrones de mensajería; en este sentido los Web Services a pesar de ofrecer una forma estándar de comunicación e interoperabilidad, y de permitir la reutilización de activos de información al ofrecer métodos de acceso e integración; estos no permiten orquestación de flujos de tareas, entre otras características necesarias en una SOA.

Por ello, Papazoglou y Van Den Heuvel (2005) proponen un middleware ESB de integración como componente necesario en una SOA; siguiendo esta recomendación se considerará en la implementación de la arquitectura del buscador; además de otra ventajas que ofrece un ESB como: flexibilidad en la arquitectura para adaptarse a los cambios y crecimiento de acuerdo a las necesidades; escalabilidad sin necesidad de afectar el código; separación de la lógica del negocio, protocolos y formatos de mensajes para un rápido desarrollo y es orientado a la configuración en vez de la codificación de la integración.

Existen varias herramientas que implementan un ESB, entre ellas WebSphere ESB, Oracle ESB, ServiMix, Jboss ESB, Mule, Open ESB, entre otras.

Ahora bien, basado en una investigación realizada por Ciurana (2007), donde evaluó varios productos ESB como: Matrix, Mule, Open ESB, Sonic ESB y WebSphere ESB; tomó en consideración nueve criterios de evaluación y dedujo que, Mule y WebSphere ESB cumplieron con todos los criterios, superando a los demás, con la diferencia que WebSPhere ESB tiene un costo de licencia muy alto

Adicionalmente, realizó una serie de pruebas “en caliente” de cada uno de los

productos y concluyó que Mule es el mejor ESB de código abierto existente en el mercado debido a que posee una comunidad de desarrollo a su alrededor activa; esta en producción en muchas empresas grandes alrededor del mundo; es fácil de instalar, implementar, mantener y extender; es de fácil comprensión y supera a otros sistemas por el gran número de adaptadores y extensiones.

En este mismo sentido, Ceijas (2008) expuso un caso de éxito en la empresa CANTV, en la ponencia presentada en el foro latinoamericano de sistemas, tecnología y comunicaciones; donde explicó los beneficios que se adquirieron al implementar un ESB usando Mule e incluso hizo muestra de algunos números estadísticos en cuanto al rendimiento de la plataforma.

Tomando en cuenta estas dos experiencias, en el presente desarrollo se decidió el uso de la herramienta Mule para la implementación de un ESB como middleware de la SOA a implementar.

Descripción del Sistema

El sistema, el cual se denominó “ARANEA”, es un motor de búsqueda dirigido, basado en Crawlers; que permitirá hacer búsqueda, extracción e indexación de contenidos en sitios Web, partiendo de un conjunto de semillas o páginas iniciales suministradas por un administrador. El sistema comienza a recorrer las páginas siguiendo los hiperenlaces de salida para continuar con nuevas páginas, siempre y cuando estas estén dentro del dominio de las páginas iniciales. Este recorrido se debe realizar en base a dos parámetros de configuración como lo son la profundidad y la amplitud que deben ser suministrados por el administrador.

La información de la metadata de cada página debe ser obtenida a parte, para identificar las mismas, además el contenido tendrá que ser indexado para ayudar en el proceso de búsqueda.

El Crawler, debe permitir la extracción de varios contenidos al mismo tiempo, cumpliendo con las políticas de exclusión de robots y de no sobrecargar los distintos

servidores; por esta razón debe ordenar el tiempo de acceso a cada servidor y página.

Debe ser construido con tecnología que permita su portabilidad a distintas plataformas de sistemas operativos, con el objeto de que no se limite su despliegue. También debe permitir distribución de carga, para que sus componentes sean instalados en varios servidores. Por tal motivo, el sistema se desplegará bajo enfoque SOA, y sus componentes se comunicarán usando estrategias de mensajes para facilitar la integración; permitiendo la interacción de objetos externos al sistema con algunos de sus componentes como el de búsqueda, indexación y acceso a base de datos.

El acceso al sistema será a través de dos interfaces, una donde un administrador podrá agregar tantos sitios iniciales o semillas, y el establecimiento de parámetros de configuración; y la otra donde el usuario podrá hacer consultas. Por último, podrá dar servicio a distintos almacenes de datos, que podrán estar el lugares remotos, por ellos se debe especificar entre los parámetros la dirección IP de la base de datos.

Mapa Conceptual

A partir de la temática que aborda el dominio de la aplicación, en un mapa conceptual se puede representar la relación e interacción de los principales conceptos de un motor de búsqueda. Dicho mapa de la *Figura No. 31*, permite conceptualizar un modelo funcional de los procesos, lo cual ayuda a establecer relaciones de dependencia entre los conceptos y es útil para modelar el sistema.

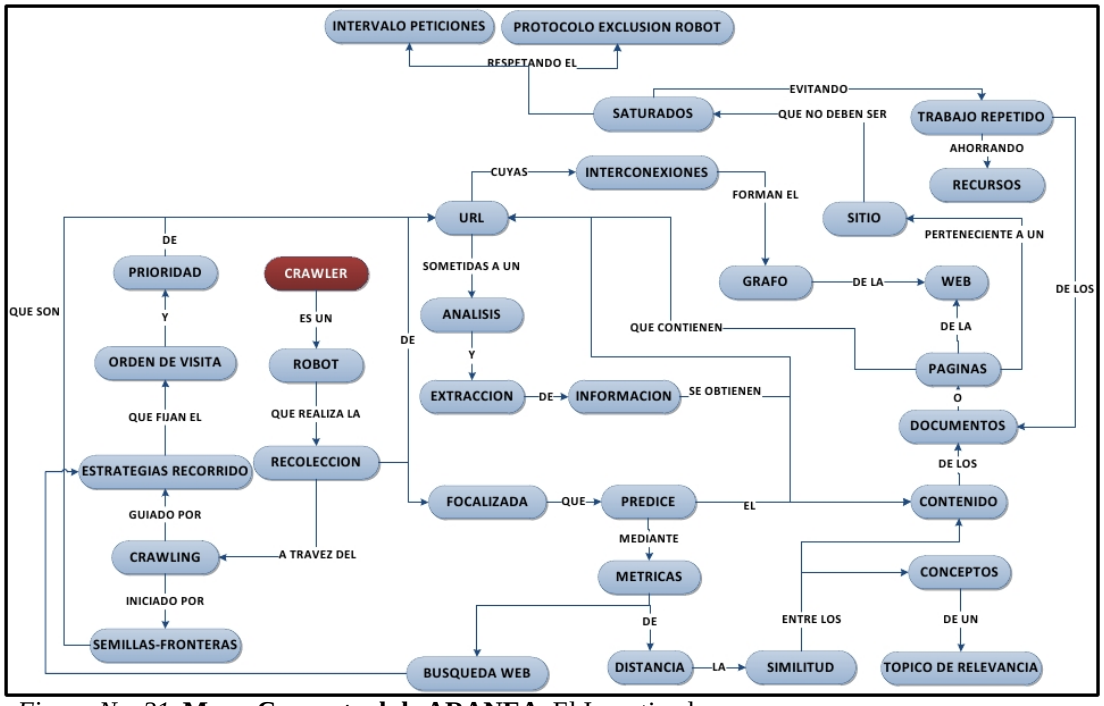

Figura No. 31. **Mapa Conceptual de ARANEA.** El Investigador.

Historias de Usuario

En aras de elicitar los requisitos, se hará uso de las historias de usuario. En el presente estudio se realizarán de acuerdo a los procesos principales y funciones descubiertos en las bases teóricas, análisis de los motores de búsqueda y la descripción del sistema; las mismas contendrán una descripción general del proceso en lenguaje formal. En los cuadros No. 12 al 15, se encuentran las descripciones de las historias de usuario.

Cuadro No. 12
Historia de Usuario: Introducir Semillas y Parámetros de Configuración

Historia de Usuario
Numero: 1
Nombre: Introducir Semillas y Parámetros de Configuración
Descripción: El usuario administrador, entra en la interfaz de administración y se muestran las opciones para agregar las semillas y parámetros como: profundidad, amplitud y dirección IP de la Base de Datos. Los parámetros son mandatarios, pero las semillas puede ser dejada en blanco; luego el usuario administrador puede dar click en enviar.

Fuentes. El Investigador.

Cuadro No. 13
Historia de Usuario: Extraer Contenido

Historia de Usuario
Numero: 2
Nombre: Extraer Contenido
Descripción: El sistema toma las direcciones de los sitios que fueron establecidos como semillas. Si las semillas no son suministradas, el motor no hará este proceso de extracción; si las semillas son insertadas y la base de datos ya contiene datos adicionará a las semillas iniciales, aquellos sitios que se encuentran almacenados en la base de datos para colocarlas como parte del conjunto de partida. El sistema inspecciona las entradas del conjunto de partida, extrae su contenido y la metadata; para luego tomar los enlaces a otras páginas y se comience iterativamente a extraer los datos de las nuevas páginas encontradas. Este proceso iterativo se realizara dependiendo de la profundidad y amplitud establecidas como parámetro. La extracción del contenido de las páginas tanto las iniciales como las nuevas encontradas, es realizado de acuerdo a un ranking de ordenamiento. El contenido extraído y la metadata es guardada en la base de datos especificada.

Fuente. El Investigador.

Cuadro No. 14
Historia de Usuario: Indexar Contenido

Historia de Usuario
Numero: 3 **Nombre: Indexar Contenido**
Descripción: El sistema toma el contenido guardado por cada una de las páginas almacenadas en la base de datos y elabora un índice invertido para ordenar la información; ya sea en formato html, pdf, word, odt, xls, etc. permitiendo cualquier tipo de extensión, exceptuando contenido multimedia.

Fuente. El Investigador.

Cuadro No. 15
Historia de Usuario: Consultar Contenido

Historia de Usuario
Numero: 4
Nombre: Consultar Contenido
Descripción: El usuario ingresa a la interfaz de búsqueda o consulta, en donde puede ingresar la o las palabras claves referentes a su búsqueda y da click en el botón buscar; luego el sistema hace uso del índice inverso, y a través de un algoritmo de relevancia devuelve como resultado ordenado a la interfaz, un conjunto de entradas de la base de datos que contienen la o las palabras introducidas.

Fuente. El Investigador.

Diagrama Preliminar de Casos de Uso

Tomando en cuenta las historias de usuario, se desprende el siguiente diagrama de casos de uso en primera instancia del motor de búsqueda, en la *Figura No. 32*

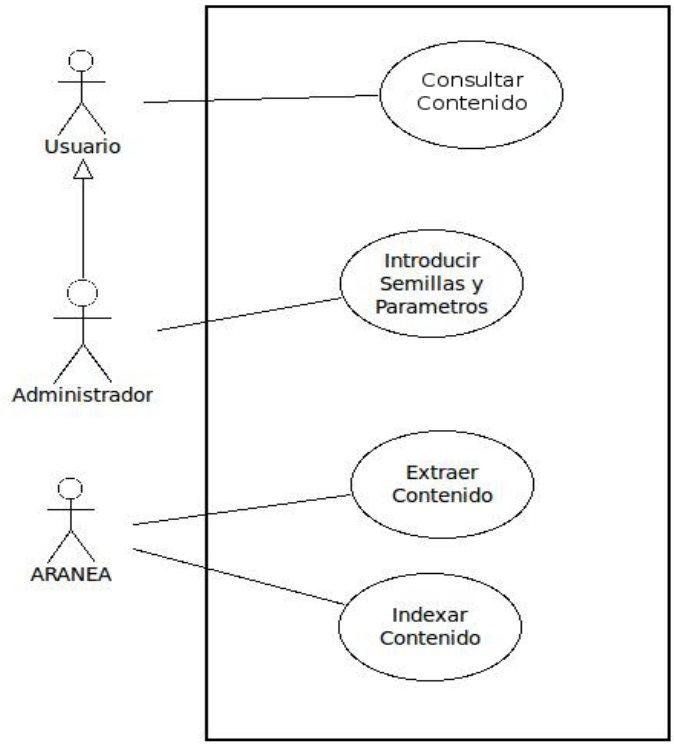

Figura No. 32. **Diagrama de Casos de Uso Preliminar de ARANEA.** El Investigador.

Análisis Orientado a Servicio

En el desarrollo de una SOA el análisis constituye la fase de primordial importancia, por lo cual se debe garantizar que se lleve a cabo de manera ordenada, detallada y que garantice la documentación. Los siguientes apartados se presentan cada uno de las etapas del proceso de análisis.

Las preguntas principales hechas durante esta fase son:

• ¿Qué servicios necesitan ser construidos?

• ¿Qué lógica necesita ser encapsulada por cada servicio?

El grado al cual se respondan estas preguntas está directamente relacionado con la magnitud del esfuerzo invertido en el análisis. La determinación de las capas de servicio construir y cómo enfocar su entrega, son puntos de decisiones críticos que

terminarán formando la estructura de todo el ambiente orientado a servicio.

Especificación de Requisitos

El primer paso a seguir para el análisis orientado a servicios es proveer de una especificación de requisitos; el cual se ha estructurado tomando como base algunas de las directrices dadas por el estándar "IEEE Recommended Practice for Software Requirements Specification ANSI/IEEE 830 1998". Debido a que no se trata de un documento individual dedicado únicamente a la especificación de requisitos, se ha incluido solamente aquellas partes consideradas útiles y necesarias para la correcta interpretación de los requisitos.

Propósito

El objeto de la especificación es definir de manera clara y precisa todas las funcionalidades y restricciones de la aplicación que se desea construir.

Conceptos y Términos

En este apartado se describirá brevemente algunos de los conceptos que puedan utilizarse a lo largo del documento y cuyo significado pueda resultar difuso o desconocido para el lector.

Conceptos utilizados como sinónimos:

El término **enlace** se usará a veces como sinónimo de **hiperenlace**, es decir, cuando hablemos de dos páginas enlazadas.

Los términos **aplicación, sistema y programa** se usarán como sinónimos con el objetivo de evitar posibles redundancias en su uso. Por el mismo motivo se hará uso de lo términos **motor de búsqueda**, **buscador** y **sistema de búsqueda.**

El término de **interfaz de consulta** se utilizará como sinónimo para representar la interfaz del usuario final y que también podrá llamarse **interfaz de búsqueda.**

Acrónimos:

URL :Uniform Resource Locator, es decir, localizador uniforme de recurso.

HTML: Acrónimo inglés de Hypertext Markup Language (lenguaje de formato de documentos de hipertexto).

ERS Especificación de Requisitos Software.

API: Del inglés Application Programming Interface - Interfaz de Programación de Aplicaciones.

HTTP: Protocolo de transferencia de hipertexto (HyperText Transfer Protocol)

Diccionario de Actores

Dentro del flujo de trabajo que ocurre en ARANEA, existen 3 actores fundamentales, estos son:

- **Usuario.** Cualquier persona que usa la interfaz de búsqueda, con la intención de consultar la Web de acuerdo a criterios o palabras claves introducidos por el mismo.
- **Administrador.** Es un tipo de usuario especializado que debe tomar en cuenta, las directrices y condiciones para evaluar la confiabilidad de un repositorio y añadirlos al conjunto de semillas; también debe cargar los parámetros de configuración.
- **ARANEA.** El sistema el cual invoca los procesos dentro del mismo.

Requisitos Funcionales

En este apartado se presentan los requisitos funcionales que deberán ser

satisfechos por el sistema. Todos los requisitos aquí expuestos son esenciales, es decir, no sería aceptable un sistema que no satisfaga alguno de los requisitos aquí presentados.

RF-1: El motor de búsqueda debe extraer información de las páginas web y almacenarla en soporte físico permanente en una Base de Datos.

RF-2: Sólo soportará el protocolo HTTP.

RF-3: Debe Soportar cualquier tipo de documento (PDF, WORD, HTML, etc.), excepto multimedias

RF-4: Por cada URL, es necesario almacenar la fecha en la que se creó el registro, la fecha en la que se modificó por última vez y un identificador único que será el URL.

RF-5: A partir de un URL inicial debe ser capaz de seguir sus hiperenlaces analizándolos progresivamente hasta que no queden más dependiendo de la profundidad.

RF-6: Permitirá establecer ciertos criterios para las búsquedas; que se refieren a la profundidad de las búsquedas, límite máximo de páginas que el sistema analizará.

RF-7: Debe permitir establecer límites a los dominios.

RF-8: Deben seguirse las redirecciones, en particular para los códigos HTTP 301 y 302, siempre que no se sobrepasen los límites especificados.

RF-9: Debe tenerse en cuenta que una conexión puede fallar y reaccionar de forma adecuada, reintentando o descartando la conexión según se considere oportuno dado el caso concreto.

RF-10: Debe tener una interfaz de agregación de sitios semillas al sistema de búsqueda por parte del administrador, así como los datos para la configuración.

RF-11: El sistema debe ser capaz de bajar y tratar varios cientos de páginas por segundo.

RF-12: La aplicación debe interactuar con varios servidores al mismo tiempo

RF-13: Debe ser tolerante a comportamientos extraños, y código HTML con errores.

RF-13: El sistema debe seguir el convenio de exclusión de páginas (robot

exclusión).

RF-14: Debe evitarse poner demasiada carga de trabajo en un servidor. Por lo general se recomienda acceder a un servidor cada 30 sec.

RF-15: El sistema debe abarcar los procesos de búsqueda de páginas, extracción de contenido, extracción de datos y texto estructurado, actualización de los datos, indexación y búsqueda.

RF-16: Debe poseer una interfaz de consulta, donde se tipeen las palabras claves a buscar y a partir de allí obtenga los resultados, de acuerdo al criterio insertado.

RF-17: Debe existir un proceso de ordenación de prioridad de extracción.

RF-18: Se debe establecer un cálculo de relevancia para el orden del resultado de las consultas.

RF-19: Se deben extraer los contenidos de la metadata de las página y guardarlos; así como sus enlaces.

Requisitos No Funcionales

Los requisitos no funcionales son primordiales en el desarrollo de esta herramienta, los cuales se nombran a continuación:

RN-1: Se desea ofrecer un sistema distribuido, sin restricciones tecnológicas notables.

RN-2: Debe permitir el acceso simultáneo a los componentes. Y a una mayor cantidad de páginas Web al mismo tiempo.

RN-3*:* Reducir al máximo los error del sistema

RN-4: El sistema deberá ser tolerante a fallos de subsistemas externos.

RN-5: Será deseable que la implementación del sistema ofrezca suficiente modularidad para poder reemplazar partes del mismo con facilidad.

RN-6: Debe proveer fácil acceso a los mecanismos de configuración de la Herramienta.

RN-7: La herramienta debe adaptarse para acceder a documentos de diferentes tipos.

RN-8: Los procesos encargados de las conexiones y el análisis del contenido de las páginas deben permitir su modificación, adición de nuevos clientes y/o analizadores o su reemplazo por elementos similares de la manera más simple posible.

RN-9. La aplicación debe funcionar en cualquier plataforma.

RN-10: Aunque el tiempo total que el crawler necesite para completar un análisis no tiene restricciones, sí deberá garantizarse que la interfaz gráfica responda a las acciones del usuario en un tiempo razonable.

RN-11: Los datos almacenados en la base de datos han de ser siempre consistentes.

RN-12: Debe existir un bajo acoplamiento entre las diferentes estructuras arquitectónicas.

Modelo de Calidad de ARANEA

En todo proceso de desarrollo, lo deseable es que el resultado del mismo sea un producto de calidad que pueda satisfacer las necesidades de los usuarios. Muchos investigadores consideran que los requisitos no funcionales, expresados en términos de calidad, son cruciales para el diseño arquitectural, específicamente cuando las aplicaciones deben responder a situaciones críticas y a cambios en el ambiente ; por tal motivo se procederá a enmarcar los requisitos no-funcionales, adoptando un modelo de calidad. Para tal fin, en esta investigación se empleará la Norma ISO-25010, por ser una de las más usadas y recomendadas. *Ver Cuadro No. 16.*

Cuadro No. 16
Atributos de Calidad Asociados a los Requisitos No Funcionales

Requisitos No Funcionales	Propiedades de Calidad Asociada (Características de Calidad) ISO-25010
• **RN-1:** Se desea ofrecer un sistema distribuido, sin restricciones tecnológicas notables.	**Portabilidad:** Adaptable
• **RN-2:** Debe permitir el acceso simultáneo a los componentes. Y a una mayor cantidad de páginas Web al mismo tiempo.	**Fiabilidad:** Disponibilidad
• **RN-3*:*** Reducir al máximo los errores del sistema	**Funcionalidad Apropiada:** Correcta.
• **RN-4:** El sistema deberá ser tolerante a fallos de subsistemas externos.	**Fiabilidad:** Tolerancia a Fallos.
• **RN-5:** Será deseable que la implementación del sistema ofrezca suficiente modularidad para poder reemplazar partes del mismo con facilidad.	**Mantenibilidad:** Modularidad
• **RN-6:** Debe proveer fácil acceso a los mecanismos de configuración de la Herramienta	**Usabilidad:** Accesibilidad, Operable
• **RN-7:** La herramienta debe adaptarse para acceder a documentos de diferentes tipos.	**Fiabilidad:** Madurez
• **RN-8:** Los procesos encargados de las conexiones y el análisis del contenido de las páginas deben permitir su	**Mantenibilidad:** Modificable

Requisitos No Funcionales	Propiedades de Calidad Asociada (Características de Calidad) ISO-25010
modificación, adición de nuevos clientes y/o analizadores o su reemplazo por elementos similares de la manera más simple posible.	
• **RN-9.** La aplicación debe funcionar en cualquier plataforma	**Portabilidad:** Adaptable, Instalable
• **RN-10:** Aunque el tiempo total que el crawler necesite para completar un análisis no tiene restricciones, sí deberá garantizarse que la interfaz gráfica responda a las acciones del usuario en un tiempo razonable.	**Funcionalidad Apropiada:** Completa. **Eficiencia:** Tiempo de Respuesta
• **RN-11:** Los datos almacenados en la base de datos han de ser siempre consistentes.	**Seguridad:** Integridad
• **RN-12:** Debe existir un bajo acoplamiento entre las diferentes estructuras arquitectónicas.	**Compatibilidad:** Interoperabilidad. **Portabilidad:** Reemplazable.

Fuente. El Investigador.

Luego, en la Figura No. 33, se puede observar gráficamente el Modelo de Calidad, resultante del sistema ARANEA.

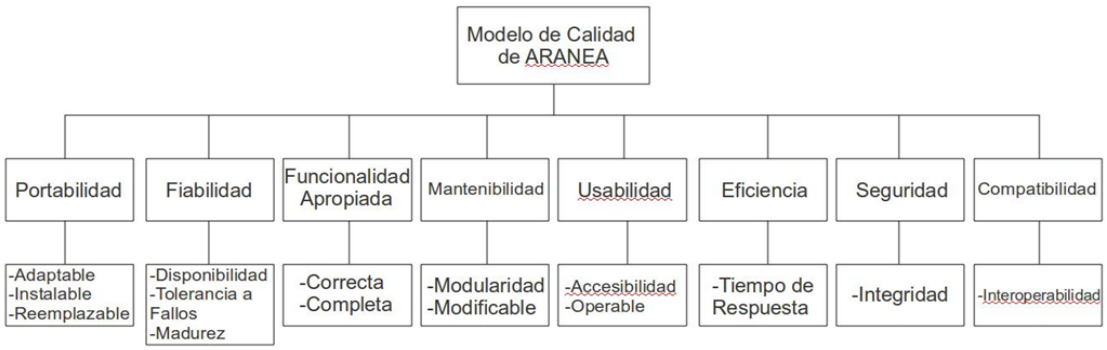

Figura No. 33. **Modelo de Calidad de ARANEA.** El Investigador.

Diagrama Final de Casos de Uso

A partir de la información recopilada, elicitación de los requisitos, y las funcionalidades y flujo de trabajo de Nutch; surgieron nuevos procesos que se tomarán en cuenta para el desarrollo de la aplicación .por lo tanto el diagrama de casos de uso final queda como en la *Figura No. 34*:

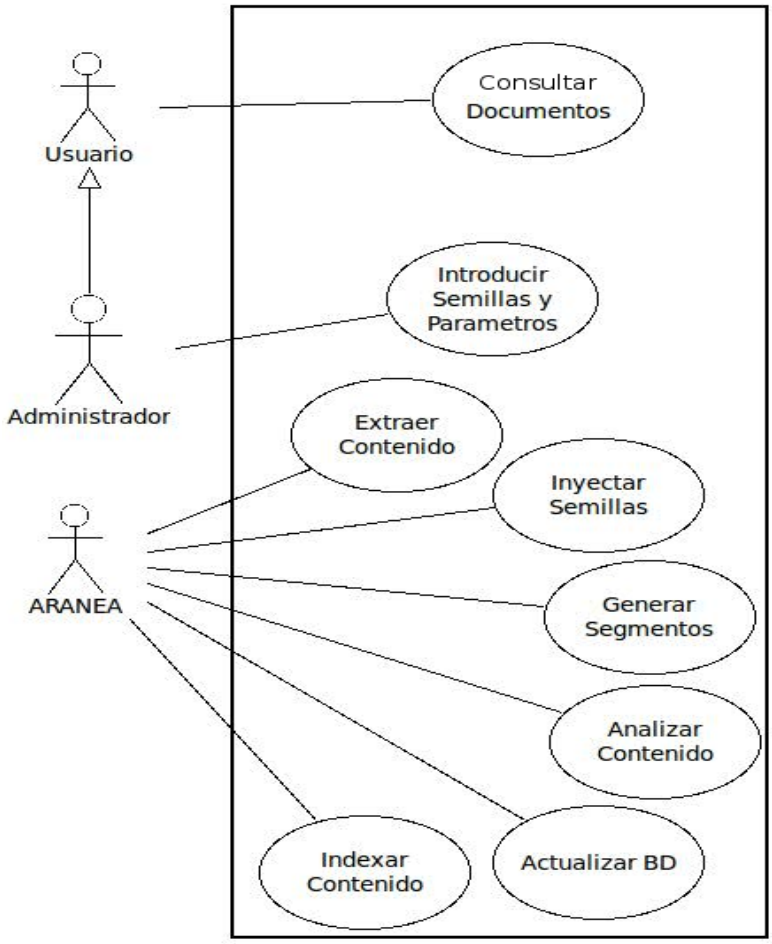

Figura No. 34. **Diagrama Final de Casos de Uso de ARANEA.** El Investigador.

En los *Cuadros No. 17 al 24*; se observa la descripción de cada uno de los casos de uso.

Cuadro No. 17
Caso de Uso: Introducir Semillas y Parámetros

Nombre del Caso de Uso: Introducir Semillas y Parámetros	**Ref. 1**
Descripción: Permite al Administrador agregar los sitios semillas y los parámetros de configuración.	
Pre-condición: Ninguna	

	Acción del Actor	***Acción del Sistema***

Nombre del Caso de Uso: Introducir Semillas y Parámetros		**Ref. 1**
Curso Normal	1. El usuario entra a la interfaz administrativa y rellena los datos de las semillas y los parámetros como: amplitud, profundidad y dirección ip de la base de datos	2. El sistema valida los datos y los envía a la cola asíncrona.
Curso Alterno		2. El sistema valida los datos, si faltan los parámetros de configuración, muestra un error a la interfaz, informando sobre la falta de datos.
Post-condición: Objeto guardado en la cola asíncrona con las semillas y los parámetros de configuración.		

Fuente. El Investigador.

Cuadro No. 18
Caso de Uso: Inyectar Semillas

Nombre del Caso de Uso: Inyectar Semillas	**Ref. 2**
Descripción: El sistema inicializa el conjunto de datos iniciales.	
Pre-condición: Debe Existir datos en la cola asíncrona	
	Acción del Sistema
Curso Normal	1. Verifica que existan datos en la cola asíncrona y los toma de la misma. 2. Determina si hay semillas o no. 3. Si hay semillas, se normalizan las mismas para evitar malformaciones en la URL. 4. Se inicializa el documento por cada semillas ingresada. 5. Guarda en la base de datos las semillas inicializadas.

Nombre del Caso de Uso: Inyectar Semillas	**Ref. 2**
Curso Alterno	3. Si no hay semillas, el proceso no continua.
Post-Condición: Documentos inicializados y guardados en la base de datos.	

Fuente. El Investigador.

Cuadro No. 19
Caso de Uso: Generar Segmentos

Nombre del Caso de Uso: Generar Segmentos	**Ref. 3**
Descripción: El sistema determina un orden para la extracción del contenido.	
Pre-condición: Debe Existir documentos inicializados en la base de datos	
	Acción del Sistema
Curso Normal	1. Busca en la Base de Datos los documentos. 2. Verifica el Score que contiene. 3. Establece el nuevo Score. 5. Actualiza en la base de datos
Post-Condición: Documentos actualizados con el nuevo Score	

Fuente. El Investigador.

Cuadro No. 20
Caso de Uso: Extraer Contenido

Nombre del Caso de Uso: Extraer Contenido.	**Ref. 4**
Descripción: El sistema busca en la Web el contenido de cada una de las entradas de los documentos.	
Pre-condición: Debe Existir documentos en la base de datos con Score definido	
	Acción del Sistema
Curso Normal	1. Busca en la Base de Datos los documentos. 2. Configura los agentes de búsqueda 3. Realiza colas de acuerdo a los dominios

Nombre del Caso de Uso: Extraer Contenido.	**Ref. 4**
	5. Determina el orden de las colas de acuerdo al Score de cada entrada. 6. Se crean los hilos por cada entrada. 7. Define el tipo de contenido a descargar. 8. Verifica la exclusión de robots del sitio URL. 9. Accesa al URL especificado. 10. Extrae el contenido de acuerdo al tipo. 11. Verifica las colas si continua con el proceso o hace una parada de la iteración. 12. Guarda los documentos con el contenido extraído
Curso Alterno 1	8. Si hay el dominio no permite accesos de robots, el sistema pasa por alto el URL y continúa con el siguiente.
Curso Alterno 2	9. Si existe algún problema con el acceso al URL, se paso por alto y se agrega el URL al último de la cola, dando oportunidad hasta tres veces
Curso Alterno 3	9. Si hay redirecciones de tipo 301 y 302; la URL se pasa por alto y se sigue la redirección.
Post-Condición: Documentos actualizados con el contenido extraído	

Fuente. El Investigador.

Cuadro No. 21
Caso de Uso: Analizar Contenido

Nombre del Caso de Uso: Analizar Contenido	**Ref. 5**
Descripción: El sistema analiza el contenido de los documentos y extrae la metadata, enlaces de entrada y de salida	
Pre-condición: Debe Existir documentos con contenido extraído.	
	Acción del Sistema
Curso Normal	1. Busca en la Base de Datos los documentos. 2. Verifica que el contenido este bien formado

Nombre del Caso de Uso: Analizar Contenido	**Ref. 5**
	3. Determina el tipo del contenido. 4. Extrae la metadata del contenido. 5. Determina los enlaces de salida y entrada del contenido. 6. Verifica que aun exista entradas en la lista. 7. Guarda los documentos con la metadata y enlaces.
Curso Alterno	4. Si hay un problema extrayendo la metadata, paso por alto la entrada y sigue con la siguiente.
Post-Condición: Documentos actualizados con la metadata y enlaces	

Fuente. El Investigador.

Cuadro No. 22
Caso de Uso: Actualizar BD

Nombre del Caso de Uso: Actualizar BD	**Ref. 6**
Descripción: El sistema toma los enlaces de salida e inicializa los nuevos enlaces encontrados a nuevos documentos.	
Pre-condición: Debe Existir documentos en la base de datos con nuevos enlaces de salida.	
	Acción del Sistema
Curso Normal	1. Busca en la Base de Datos los documentos. 2. Determina los enlaces de salida encontrados de cada ítem. 3. Verifica que ya no esté guardado cada enlace nuevo 5. Instancia un documento por cada enlace nuevo. 6. Verifica el Score de ordenamiento de cada documento, para determinar la fecha de revisita y actualización de contenido. 7. Guarda los documentos nuevos y actualizados con Score
Post-Condición: Nuevos documentos ingresados a la Base de Datos y actualizados con el nuevo Score	

Fuente. El Investigador.

Cuadro No. 23
Caso de Uso: Indexar Contenido

Nombre del Caso de Uso: Indexar Contenido.	**Ref. 7**
Descripción: El sistema busca las palabras del contenido y las asocia en un índice inverso	
Pre-condición: Debe Existir documentos en la base de datos	
	Acción del Sistema
Curso Normal	1. Busca en la Base de Datos los documentos. 2. Extrae todas las palabras de cada contenido. 3. Construye el índice inverso a partir de las palabras extraídas, relacionando con la entrada del documento en la base de datos. 4. Guarda el índice inverso en un almacén de datos aparte.
Post-Condición: Índice inverso creado y guardado.	

Fuente. El Investigador.

Cuadro No. 24
Caso de Uso: Consultar Documentos

Nombre del Caso de Uso: Consultar Documentos.		**Ref. 8**
Descripción: El usuario usa la interfaz de consulta o búsqueda, donde puede introducir palabras claves para su búsqueda y el sistema devuelve a la interfaz un resultado.		
Pre-condición: La Base de Datos debe estar indexada.		
	Acción del Actor	***Acción del Sistema***
Curso Normal	1. El usuario entra a la interfaz de búsqueda y coloca la o las palabras claves por las que desea buscar el contenido y da click a la opción de buscar.	2. El sistema recibe los datos y busca

Nombre del Caso de Uso: Consultar Documentos.		**Ref. 8**
		las palabras en el índice inverso de acuerdo a la correspondencia de las palabras insertadas. 3. Ordena de acuerdo a criterios de relevancia, el grupo de resultados de los contenidos. 4. Envía la lista de resultados a la interfaz y los muestra.
	5. El usuario recibe los resultados y puede dar click en el titulo de aquellos que desea ver y se redirecciona el navegador al URL en el WEB. 6. El usuario puede seguir insertando palabras para la búsqueda.	
Post-condición: Lista de resultados de búsqueda.		

Fuente. El Investigador.

Selección de los Servicios Candidatos

De acuerdo Earl (2005), los procesos principales de un sistema servirán a priori como servicios candidatos de la arquitectura. Por otra parte, en la caracterización de Nutch se palpó, que este contiene los procesos importantes de un motor de búsqueda, además de ejecutar otros procesos que ayudan al proceso. Como Nutch es el proyecto que se seleccionó como base al sistema ARANEA, ergo, dichos procesos se tomarán como servicios candidatos.

En este orden de ideas, atendiendo a los requisitos elicitados y los casos de uso

definidos, se determinarán los subprocesos, que también podrán ser servicios candidatos dentro de la arquitectura; para ello se debe determinar si estos subprocesos son compartidos por otro proceso o bien será necesario su publicación para ser usado por otro proceso o ente externo.

Estos servicios candidatos puede estar basados en tareas o entidades y cada opción ofrece sus respectivas ventajas y desventajas. Observando esta situación y previendo que el presente estudio no persigue dentro de sus objetivos realizar una comparación entre estas dos modalidades, se optó por un modelado centrado en tareas. He aquí en la *Cuadro No. 25*, donde se observan los procesos principales y sus subprocesos o tareas.

Cuadro No. 25
Procesos y Subprocesos del Sistema de Búsqueda

Procesos	Subprocesos
Introducir Semillas y Parámetros.	
Inyectar Semillas	- Extraer Semillas - Normalizar Semillas - Guardar en Base de Datos
Generar Segmentos	- Buscar Todos los Datos en la Base de Datos - Verificar Generación de Score - Establecer Score - Guardar en Base de Datos
Extraer Contenido	- Buscar Todos los Datos en la Base de Datos - Crear Configuración de Agentes - Verificar Intervalo de Extracción - Delimitar Filtros - Establecer Cola por Dominio

Procesos	Subprocesos
	- Crear Hilos Por Sitio - Crear Cola de Hilos - Determinar Protocolo - Extraer Contenido. - Guardar en Base de Datos
Analizar Contenido	- Buscar Todos los Datos en la Base de Datos - Verificar Análisis - Extraer Palabras - Actualizar Atributos - Extraer Enlaces y Metadata - Verificar Redirecciones - Guardar en Base de Datos
Actualizar BD	- Buscar Todos los Datos en la Base de Datos - Crear Nuevas Páginas Encontradas - Redefinir Score - Actualizar Información - Guardar en Base de Datos
Indexar Contenido	- Extraer Palabras. - Construir Índice Invertido. - Guardar Índice.
Consultar Documentos	- Determinar Relevancia - Ordenar Búsqueda - Mostrar Resultados

Fuente. El Investigador.

El mayor interés en este paso es asegurar que cada operación de servicio candidata sea potencialmente reusable y tan autónoma como sea posible.

De esta manera se procede a revisar las agrupaciones de las operaciones que son las que conforman cada servicio candidato respectivamente y proceder a aplicar los criterios de orientación a servicio importantes en esta etapa de análisis.

Así pues, atendiendo a que solo los subprocesos de comunicación con la Base de Datos son reusables por los demás procesos y que cada proceso es autónomo en su algoritmo; añadiendo que el insumo de un proceso es el resultado del anterior proceso, se decidió:

- Eliminar el subproceso de extracción de semillas y agregarlo como parte del proceso de introducir semillas y parámetros. Ya que el proceso de extracción de semilla se realizaba por medio de un archivo de texto; por ello se dará esta funcionalidad al administrador y pueda establecer la configuración y las semillas iniciales a través de un servicio.
- Agregar un nuevo proceso de manipulación de los datos de la Base de Datos. Esto debido a que son subprocesos comunes de los procesos principales; por tal motivo se desacoplará para que sean servicios.
- Por lo anterior, se eliminarán los subprocesos de insertar y seleccionar de la Base de Datos de cada proceso y estos ahora accesarán a la Base de Datos a través de servicios.

En resumen los servicios estarían conformado por: Introducir Semillas y Parámetros, Inyectar Semillas, Generar Segmentos, Extraer Contenido, Analizar Contenido, Actualizar BD, Indexar Contenido, Consultar Documentos y Conectar con BD.

Planificación de Entregas

Una de las etapas de la metodología XP, es determinar las prioridades sobre las

tareas que ejecutará el sistema, con el fin de organizarlas para construirlas en cada iteración del proceso de desarrollo.

Una vez que se han definido las prioridades de las tareas y además se han enlazado con las iteraciones, los desarrolladores deben medir el esfuerzo necesario para completarlas, tomando en cuenta no solo el tiempo para su desarrollo, sino también el invertido en la planificación y la validación del producto. La *Cuadro No. 26*, muestra la planificación de las iteraciones, mostrando el esfuerzo necesario en semanas para completar cada tarea del sistema.

Cuadro No. 26
Planificación de Entregas

Iter.	Tareas a Desarrollar	Esfuerzo (semanas)
0	- Describir la Arquitectura de Software. - Definir Herramientas de Desarrollo.	2
1	- Desarrollar Interfaz de Agregación de Semillas y Parámetros de Configuración	3
2	- Adaptar el Componente de Inyección, Orientado a Servicio	3
3	-Adaptar el Componente de Generación de Segmentos, Orientado a Servicio	3
4	- Adaptar el Componente de Extracción de Contenido, Orientado a Servicio	3
5	- Adaptar el Componente de Análisis de Contenido, Orientado a Servicio	3
6	-Adaptar el Componente de Actualización de Base de Datos. Orientado a Servicio	3
7	- Adaptar el Componente de Indexación, Orientado a Servicio	3

Iter.	Tareas a Desarrollar	Esfuerzo (semanas)
8	- Configurar el Bus de Servicio - Realizar la orquestación de los componentes	3
9	- Desarrollar la Interfaz de Búsqueda y Resultado del Usuario	2

Fuente. El Investigador.

Fase II: Fase de Diseño de Arquitectura

Diagrama Conceptual

Como parte importante del diseño orientado a servicio, cuyo propósito es modelar el dominio del problema; una vez que se conocen los conceptos involucrados y sus relaciones se puede hacer uso de artefactos tales como el diagrama conceptual, que ayudara establecer las asociaciones de los recursos u objetos a guardar; en este caso el recurso a almacenar serán las direcciones de los repositorios a los cuales el motor de búsqueda accesa y extrae su contenido; en este sentido, se revisaran los conceptos y propiedades que la caracterizan, y que podrían ser necesario almacenar; lo cual ayudara a diseñar la estructura del documento a guardar en el almacén de datos. El diagrama se describe en la *Figura No. 35.*

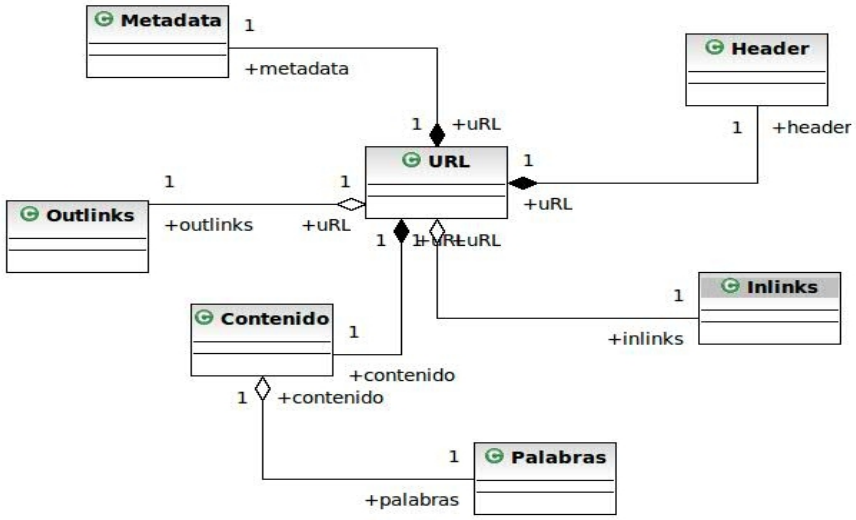

Figura No. 35. **Diagrama Conceptual.** El Investigador.

Estructura del Documento del Repositorio

Según, Jiménez (2008) “la archivística acepta el valor de describir un documento o, lo que es lo mismo, una unidad documental simple, de forma individualizada, mediante la definición de sus componentes suficientes y necesarios que permiten que se pueda reconocer, capturar y gestionar mediante un sistema adecuado” (p. 31). Como el objeto que se va a obtener de la extracción de las páginas Web será en definitiva un documento, se deben definir sus componentes o atributos; para ello se toma como base el análisis realizado en el diagrama conceptual y el mismo debe ser conducido por un modelo ontológico de dominio, el cual permite organizar y definir un conjunto de conceptos en un área de conocimiento particular; en este caso será la descripción de metadatos. Las ontologías en definitiva son necesarias para obtener vocabularios para representar conocimiento.

De acuerdo a Chaudhri (2012) “Dublin Core es la mejor iniciativa de descripción de metadata que facilita el descubrimiento de recursos electrónicos”; así la Dublin Core Metadata Initiative (DCMI) se ha convertido en el modelo ontológico de mayor envergadura, maduro, robusto y soportado; siendo una norma internacional ISO 15836 (2009). Por esta razón, se tomará dicha norma como base para la

especificación de los atributos del documento.

El conjunto de elementos de la DCMI se centra en 15 descriptores, resultado de un consenso y un esfuerzo interdisciplinar e internacional. La lista de atributos no es cerrada ni es preceptiva. Orienta en línea de mínimos, cuál tendría que ser la información a asociar a cualquier documento, por esta razón se añaden otros atributos que ayuden al proceso. A continuación la *Cuadro No. 27*, se observa el nombre de los atributos del modelo, su descripción y el descriptor que se asocia con el modelo ontológico de Dublin Core.

Cuadro No. 27
Atributos del Modelo

Nombre	Descripción	Tipo	Atributo Dublin Core
baseUrl	Atributo clave que guarda la dirección URL del documento. Es clave, ya que cada URL es único en la Web	String	Identificador
status	Describe el número del último proceso al que se sometió	Integer	
fetchTime	Fecha de su última extracción de la Web	Date	Date
prevFetchTime	Fecha Previa de su última extracción de la Web	Date	
fetchInterval	Fecha calculada para el intervalo entre extracciones	Long	
retriesSinceFetch	Número de veces que se ha intentado hacer la extracción del contenido.	Integer	

Nombre	Descripción	Tipo	Atributo Dublin Core
modifiedTime	Última fecha que se modificó el contenido	Date	
protocol	Tipo de Protocolo	String	
content	Contenido Extraído por el Crawler de la Web	byte	Description
contentType	Tipo del Contenido	String	Type/Format
prevSignature	Entidad Publicadora	String	Publisher
signature	Nombre Actual del Creador	String	Creator
title	Titulo	String	Title
text	Descripción del Contenido.	String	Subject
parseStatus	Atributo que identifica si el Contenido ha sido “parseado”	Integer	
score	Puntuación que identifica la prioridad del documento para la Extracción	Real	
reprUrl	Dirección URL, que se guarda si el mismo contenido se encuentra en la Web con dos URLs distintas	String	
headers	Guarda la Información Encabezada de la conexión	Array	
outlinks	Guarda los enlaces que posee el Contenido.	Array	Relation
inlinks	Guarda los enlaces externos que apuntan al Contenido.	Array	Source

Nombre	Descripción	Tipo	Atributo Dublin Core
markers	Marcador que identifica el proceso inyector que lo ejecutó	Array	
metadata	Guarda la metadata del contenido.	Array	Language

Fuente. El Investigador.

Iteración No. 0

El objetivo de esta iteración es definir la arquitectura que se usará para la construcción del motor de búsqueda ARANEA; además de hacer un resumen de las herramientas tecnológicas seleccionadas en la Fase de Análisis y por último establecer los patrones de diseño e integración a usar.

Diseño Orientado a Servicio

Para realizar el diseño orientado a servicio se recomienda un orden en el diseño de cada uno de los tipos de servicios. Los de aplicación son los primeros en abordarse para su diseño; estos representan la subcapa más baja de la capa de servicios, responsable de llevar a cabo cualquier paso del proceso demandado por las capas de negocio o de orquestación.

A diferencia de los servicios de capas centradas en negocios, el diseño de servicios de aplicación no requiere de experticia en análisis del negocio. La capa de servicios de aplicación es solamente una abstracción del ambiente técnico del dominio, mejor definida por aquellos que comprenden este ambiente de forma amplia.

Los servicios de negocio contendrán lógica de workflow usada para coordinar una composición subyacente de servicio. En este paso se pueden utilizar diferentes enfoques de modelado tradicionales, en este caso se utiliza diferentes diagramas de

actividad. El propósito de este ejercicio es documentar cada ruta de ejecución posible; el diagrama resultante será la entrada útil para los subsiguientes casos de prueba.

Identificar el conjunto de grupo comunes que pueden tener lugar dentro de las fronteras de los servicios es un proceso importante dentro del diseño. Anteriormente se identificaron una serie de servicios candidatos que conforman de manera preliminar las capas de servicio de negocio y de aplicación.

Aquellos servicios que representan lógica genérica, reusable y neutral, pueden ser clasificados como servicios de aplicación; en conjunto estos establecen preliminarmente una capa de servicio de aplicación. Por contrario, el rol principal de los servicios de negocio es actuar como controladores, componiéndose de servicios de aplicaciones para llevar a cabo la lógica de negocio requerida.

Así, el servicio de Establecer Configuración y Semillas pertenece a la capa de negocio, debido a que su llamado desencadenara un flujo de trabajo entre los servicios de aplicación. El resto de los servicios compondrán a la capa de aplicación, ya que cada servicio será accesado solo entre los servicios de la aplicación y cada uno de ellos corresponden a un proceso autónomo y desacoplado de la misma. Las capas de los servicios queda como se observa en la *Figura No. 36*

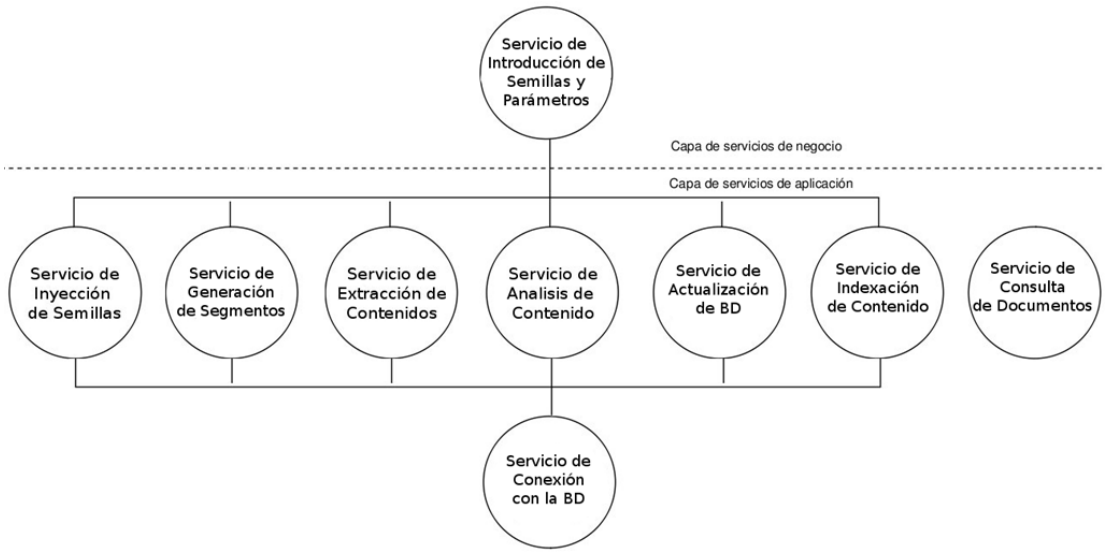

Figura No. 36. **Capas de Servicio de ARANEA.** El Investigador.

Otro proceso importante en el diseño orientado a servicio, es el delimitar la arquitectura a usar, el cual se llevará a cabo en el siguiente apartado.

Propuesta de la Arquitectura de Software

Uno de los marcos de trabajo para representar arquitectura, construido sobre la base de UML y siguiendo el estándar IEEE 1471 (2000); a su vez confeccionado para usarse en el desarrollo de software de gran escala, es el propuesto por Garland (2003); en donde señala que su metodología desde sus comienzos fue concebida como una forma ágil de representar arquitectura, exponiendo que el arquitecto puede hacer uso de aquellos artefactos que verdaderamente presenten información útil al equipo de diseñadores.

Para definir la arquitectura de software correspondiente a ARANEA, se emplearán algunas de las vistas sugeridas por Garland (2003), en especifico aquellas que para este caso de estudio resulten de valor y relevancia para alcanzar los objetivos propuestos; como también debe darse prioridad a la incorporación de patrones de diseño que direccionen un desarrollo orientado a servicios y garantizar su interoperabilidad y acceso a la información.

Es indispensable dar un panorama de todos los componentes que conformarán el sistema, como ellos se conectan y se despliegan en la infraestructura de hardware, con el fin de:

- Establecer la división de responsabilidades y roles dentro del equipo.
- Marcar puntos de discusión acerca de la pertinencia desde el punto de vista arquitectónico de la ubicación física de los componentes de software.
- Establecer los equipos y la infraestructura de red necesaria para el desarrollo e implementación de la solución de software.

La *Figura No. 37* muestra un diagrama de contexto para entender que actores y sistemas externos, interactúan con el sistema ARANEA.

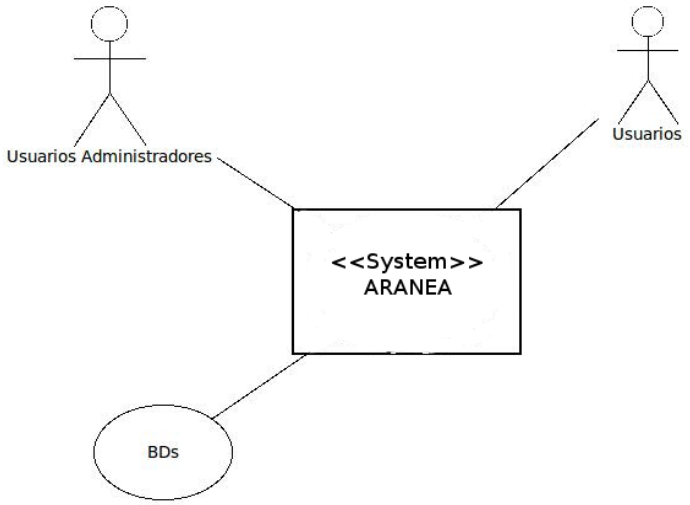

Figura No. 37. **Diagrama de Contexto de ARANEA.** El Investigador.

Tal como se ha explicado, una de las metas de esta investigación es encajar el sistema de búsqueda ARANEA en una arquitectura orientada a servicios. Este tipo de arquitectura proporciona un desacoplamiento de sus componentes, por lo que en cualquier instancia de tiempo se puede intercambiar alguna implementación de un componente sin afectar el flujo de trabajo del sistema; además, dará pie a interactuar con otros sistemas, sin importar la tecnología con que estén desarrollados.

El desarrollo basado en componentes ha llegado a tener mucha importancia en los últimos años en la construcción de sistemas de software; ya que según Garland (2003) “El desarrollo basado en componentes ayuda en la reducción de los tiempos de desarrollo y a incrementar la funcionalidad del sistema. Los componentes también permiten a los desarrolladores hacer uso desacoplado de los componentes fuera del sistema” (p. 111-112). Por lo que entonces este enfoque permite a los equipos de desarrollo construir y poner a prueba las partes del sistema de forma independiente.

Garland, establece que los componentes preliminares de la arquitectura pueden ser aquellos cuyas funcionalidades encapsula los casos de uso principales de la aplicación. Adicionalmente dice que “el componente es una unidad física de reemplazo para el sistema. La sustitución de los componentes es importante para la evolución del sistema. Si un componente es compatible con el reemplazo, este limita

el impacto de las actualizaciones de software sólo a los componentes modificados" (p. 114).

Por ello, haciendo un análisis de los casos de uso, cada uno de ellos servirán como componente de la aplicación; los mismos en aras de reducir el espacio del nombre y colocarle un sujeto, fueron renombrados como se observa en la *Cuadro No. 28*. Ahora bien, existen entes externos que hacen uso de dichos componentes, los cuales pueden ofrecer una visión de la relación que existen entre ellos, por tal motivo es importante su inclusión en la arquitectura; es así como se tienen: bd y web browser como componentes externos adicionales del modelo. En la *Figura No. 38* se observa el diagrama de componentes. Bajo este nivel de abstracción, en los diagramas de componente, interacción y actividad no se presentará el bus de integración como componente; ya que el objetivo que persiguen es el de mostrar las relaciones que poseen dichos componentes pero no a través de que medio.

Cuadro No. 28
Relación de Nombre Caso de Uso y Componentes de ARANEA

Nombre Caso de Uso	Nombre de Componente
Introducir Semillas y Parámetros	Configurador
Inyectar Semillas	Inyector
Generar Segmentos	Generador
Extraer Contenido	Fetcher
Analizar Contenido	Parser
Actualizar BD	Actualizador
Indexar Contenido	Indexador
Consultar Documentos	Buscador

Fuente. El Investigador

Otro componente que se desprende de los casos de uso y que se observa en las

capas de servicio es el de Conexión con la BD el cual se llamará Conector BD.

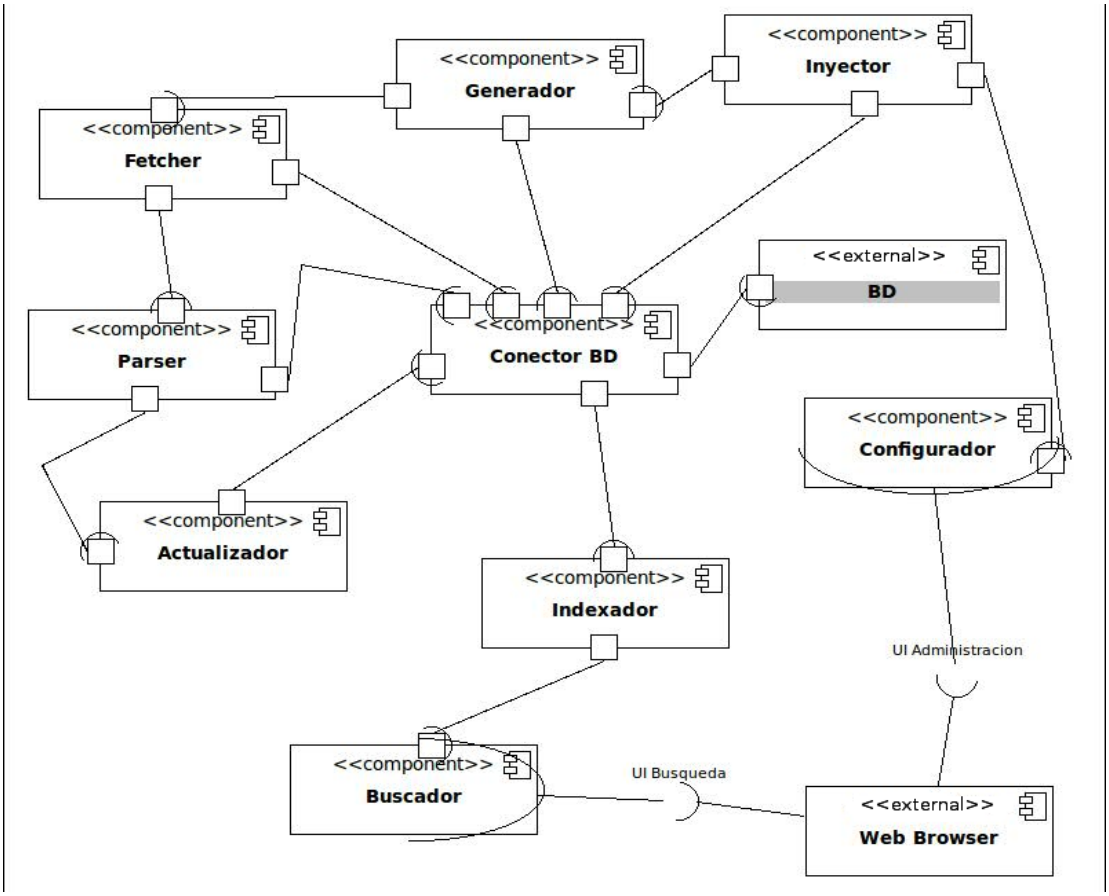

Figura No. 38. **Diagrama de Componentes de ARANEA.** El Investigador.

Otro de los diagramas recomendados por Garland, es el de secuencia, orientado a componentes, el mismo se puede observar en la *Figura No. 39*, de cómo se interrelacionan los componentes de ARANEA. Ya que según Garland (2003) "Los diagramas de interacción orientado a componentes son más escalares que los orientados a objetos" (p. 132). En dicho diagrama se agregan dos nuevas entidades: Cola Asíncrona e Índice Inverso, por ser almacenes de datos nos da una idea de cómo se substraen datos de los mismos.

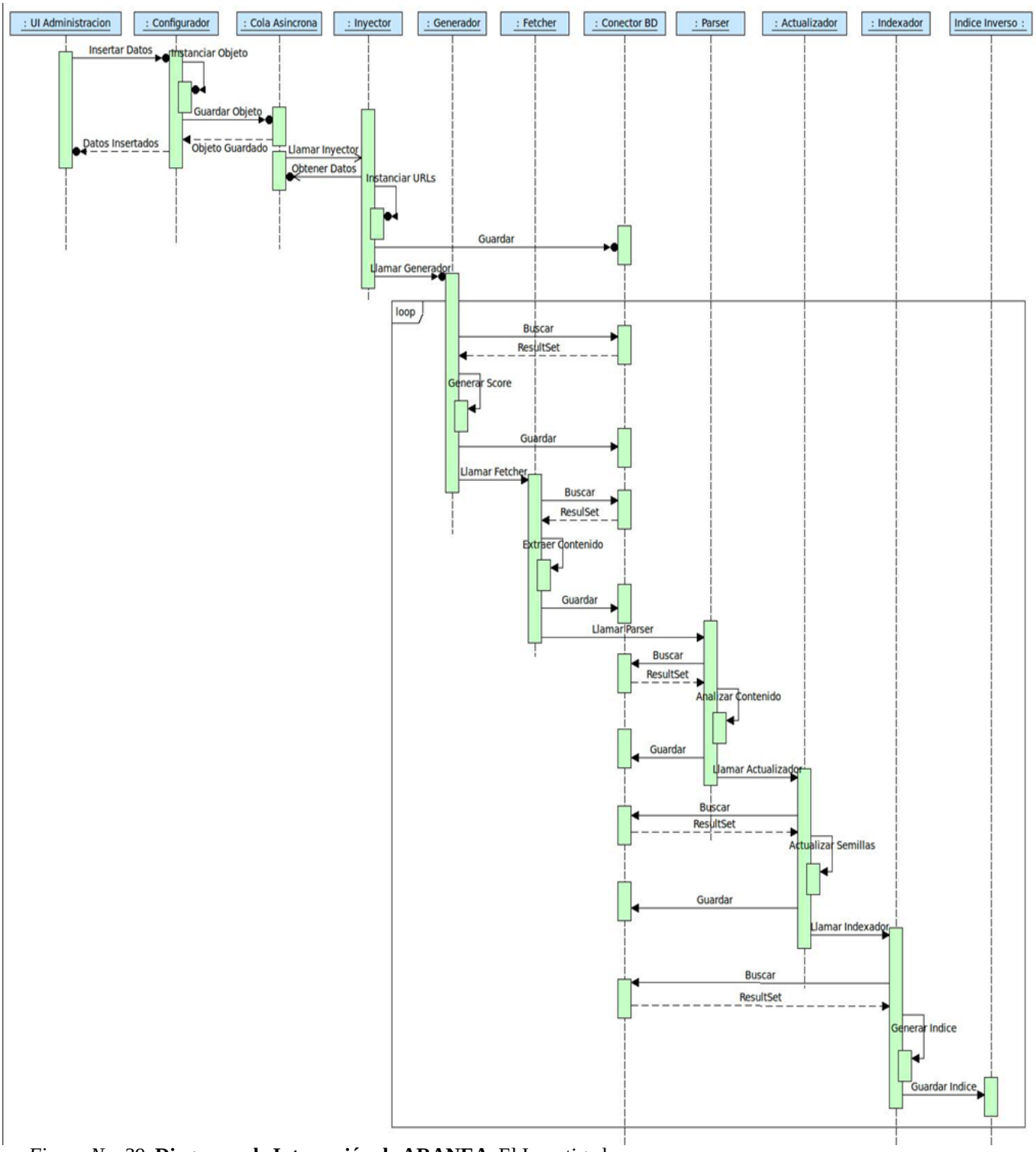

Figura No. 39. **Diagrama de Interacción de ARANEA.** El Investigador.

En la *Figura No. 39*, todos los detalles de las instancias de objetos individuales exportados por los diferentes componentes se eliden para una mejor comprensión global de la dinámica de los componentes explorados. Los estereotipos se utilizan para indicar que las instancias en el diagrama corresponden a instancias de componentes. También, el componente de la Base de Datos se obvia, teniendo en cuenta que el componente Conector BD, hace la interacción con la Base de Datos física del sistema.

En la *Figura No. 40*, se observa el diagrama de secuencia de los componentes que interactúan en el procesos de consulta por parte del usuario final, a través de la interfaz de búsqueda y resultado; que se desencadena cuando el usuario final, ingresa palabras claves para que el motor de búsqueda genere los resultados que correspondan.

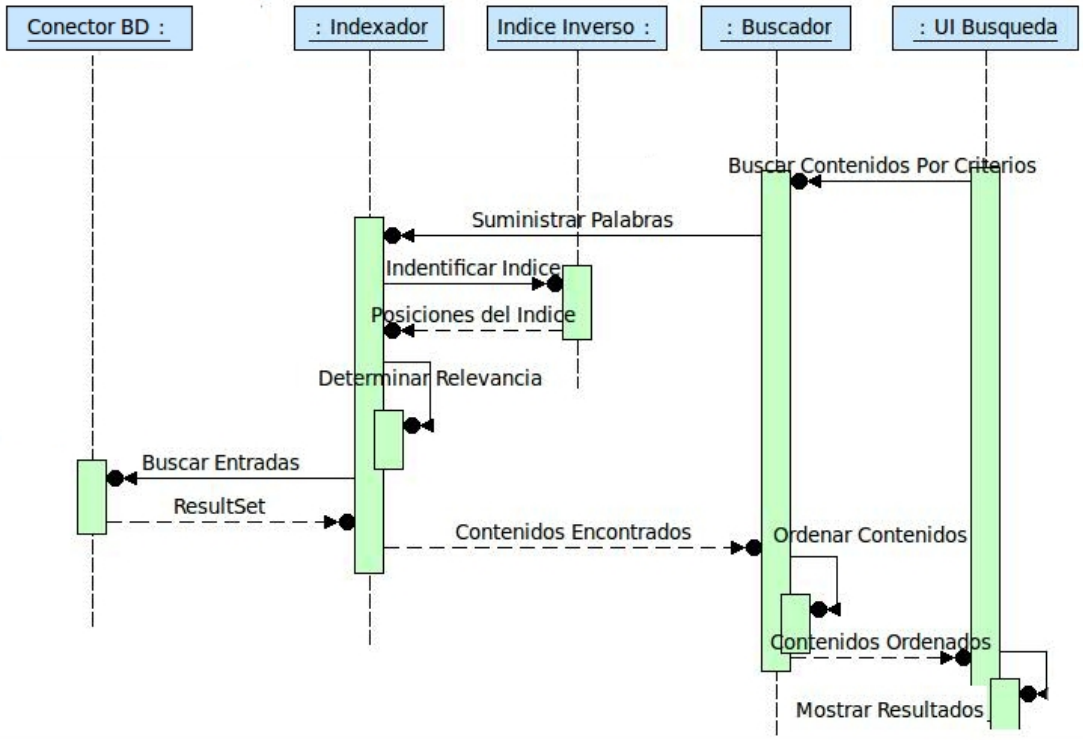

Figura No. 40. **Diagrama de Interacción del Proceso de Búsqueda o Consulta.** El Investigador.

Uno de los diagramas bastante usados, orientados a componentes, es el de actividad; en donde se puede observar las actividades que realizan cada componente en el flujo de ejecución del sistema de los procesos centrales de ARANEA, obviando el proceso de consulta. Este se puede ver en la *Figura No. 41;* donde se han obviado los componentes de la Cola Asíncrona y la Base de Datos ya que son componentes de apoyo en la gestión de almacenar datos pero no influyen en la ejecución del flujo de trabajo del sistema.

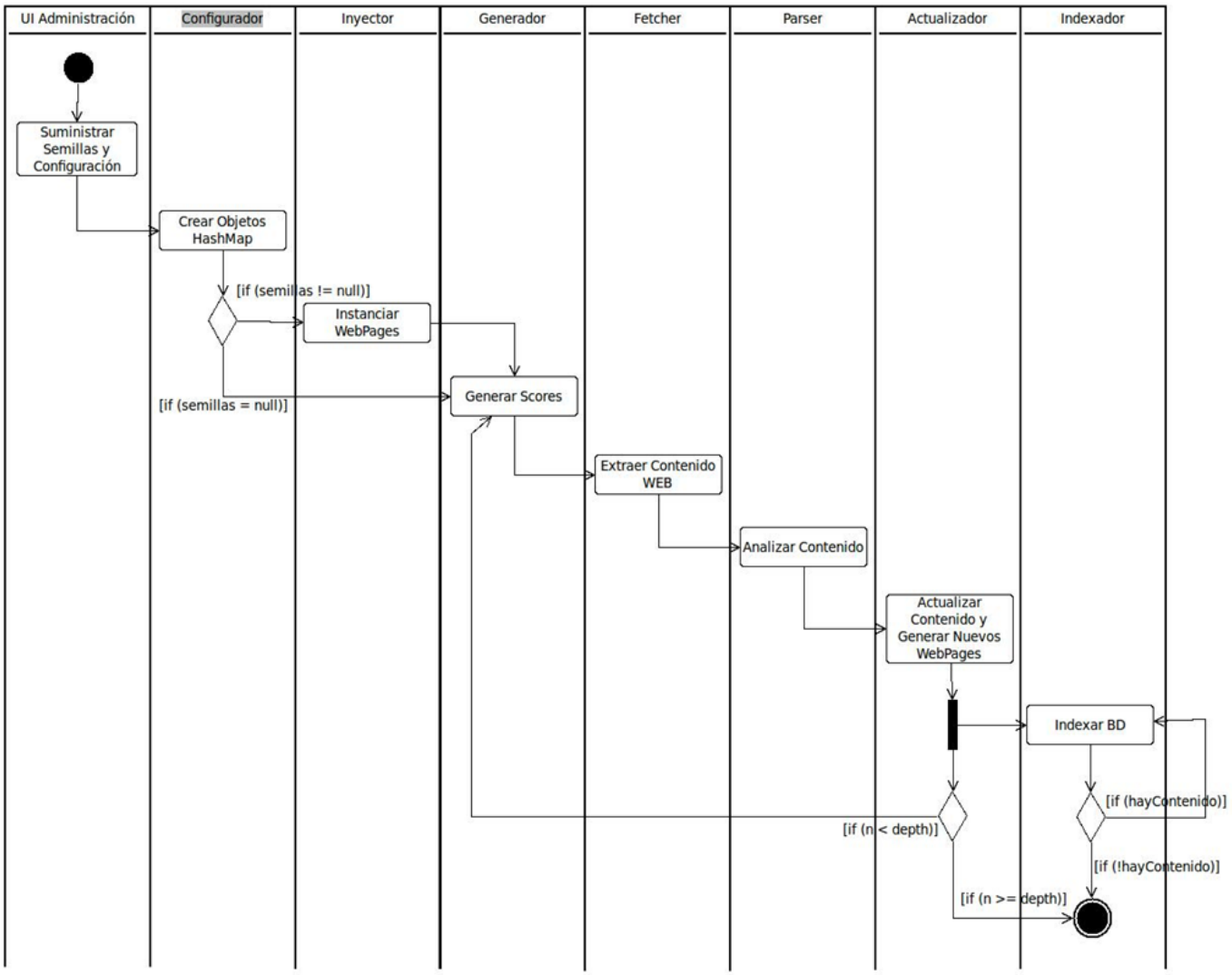

Figura No. 41. **Diagrama de Actividad de ARANEA.** El Investigador.

Ahora bien, se mostrará a continuación en la *Figura No. 42* los almacenes de datos que usará ARANEA, para ello se empleará el Diagrama Físico de Datos.

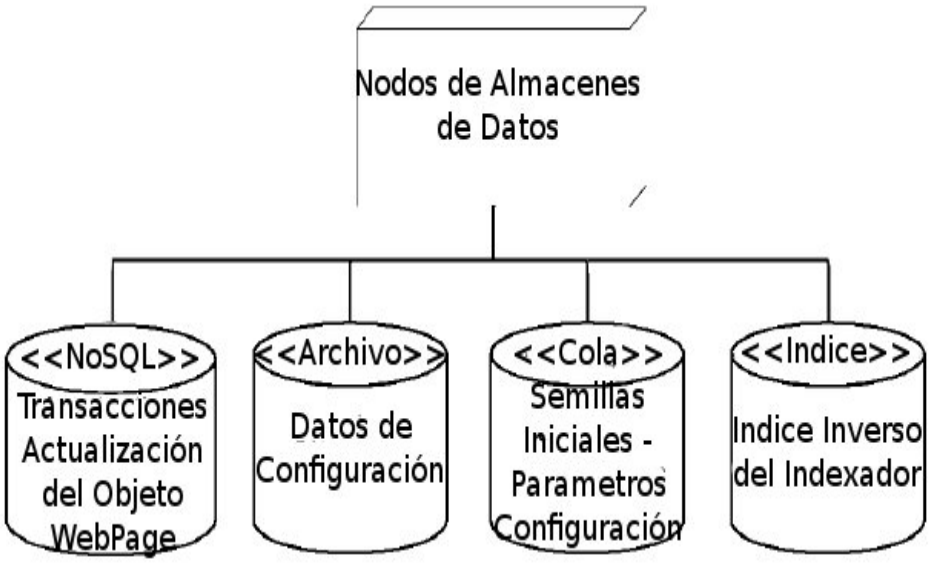

Figura No. 42. **Diagrama Físico de Datos de ARANEA.** El Investigador.

Esta metodología arquitectural orientada a componentes, permite desacoplar los procesos importantes del negocio en servicios independientes y autónomos, lo que ayuda a ubicar los servicios de la arquitectura SOA propuesta dentro del modelo, junto con un ESB como medio de integración basado en mensajes; permitiendo el acceso a través de cualquier plataforma.

Ahora bien, en favor de hacer el modelo más escalable se propondrá dos ESB dentro de la arquitectura, uno que orquestará los componentes centrales de ARANEA como son: inyector, generador, fetcher, parser, actualizador y cola asíncrona, que se llamará Bus Aranea; y otro bus de integración que establecerá la comunicación entre el Bus Aranea, las peticiones externas y la base de datos, que se llamará Bus Principal, tendrá los componentes de Configurador, Buscador, Conector BD e Indexador. De esta manera los elementos centrales de ARANEA con su Bus Aranea podrán estar desplegados en varios servidores, y en el momento que el Bus Principal reciba una

petición de nuevas semillas este determinará la relación de carga de los servidores y aquel que posea menos carga se le asignará el trabajo que sucede luego de agregar las semillas. En la *Figura No. 43* se puede percibir dicha arquitectura.

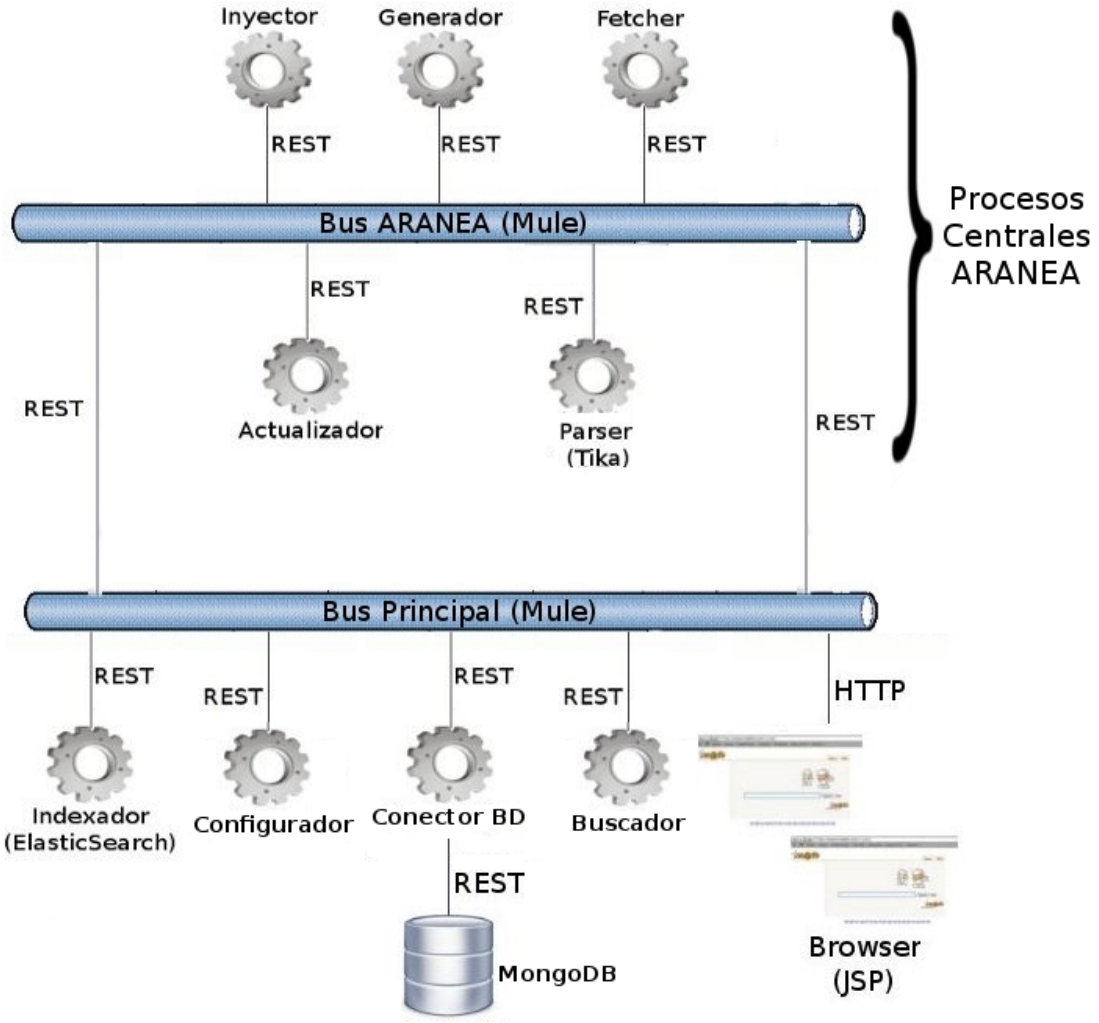

Figura No. 43. **Arquitectura SOA de ARANEA.** El Investigador.

De esta manera, la arquitectura permite desarrollar una aplicación distribuida. Usando este esquema de trabajo para el desarrollo y despliegue de la aplicación, se logra facilitar la mantenibilidad; ya que al dividir la aplicación en componentes funcionales bien diferenciados, es más fácil identificar el fragmento de código que es necesario cambiar.

Por último se mostrarán dos diagramas de despliegue, donde se observa la disposición física de los distintos nodos que componen a ARANEA. Aunque no es objetivo del estudio el despliegue o puesta en marcha de ARANEA, se dará una propuesta. En esta ocasión se dispuso reflejar tanto los buses de integración, como los componentes, servidores y los almacenes de datos para dar una visión amplia de cómo serán los vínculos de todos los elementos. En el diagrama de la *Figura No. 44*, se denota el despliegue de los elementos principales de ARANEA llamado ARANEA Core, en un solo servidor como un entorno de ejecución, para luego en la *Figura No. 45*, observar como dicho entorno puede ser instalado en varios servidores y su conexión con el Bus Principal.

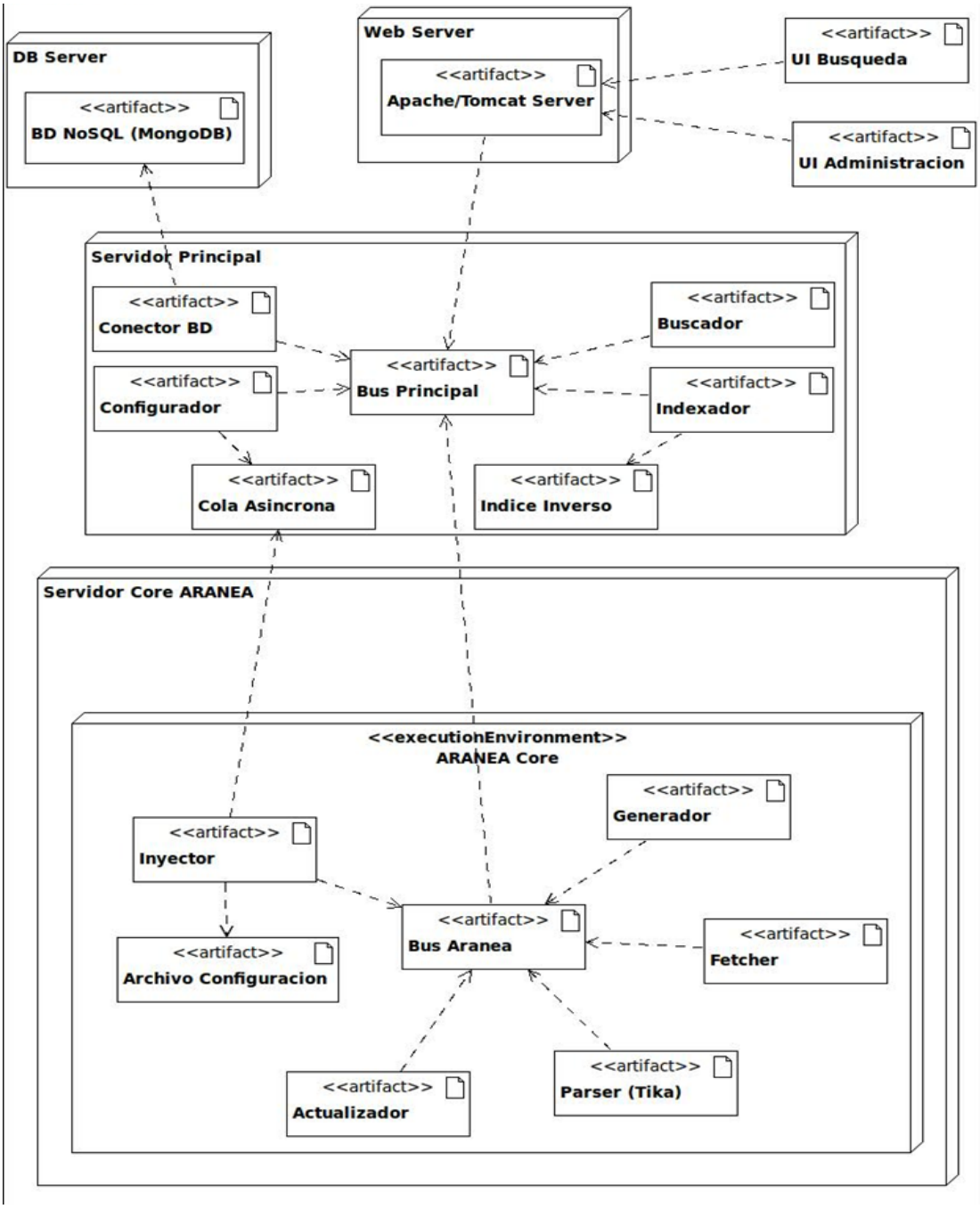

Figura No. 44. **Diagrama de Despliegue con ARENA Core Detallado.** El Investigador.

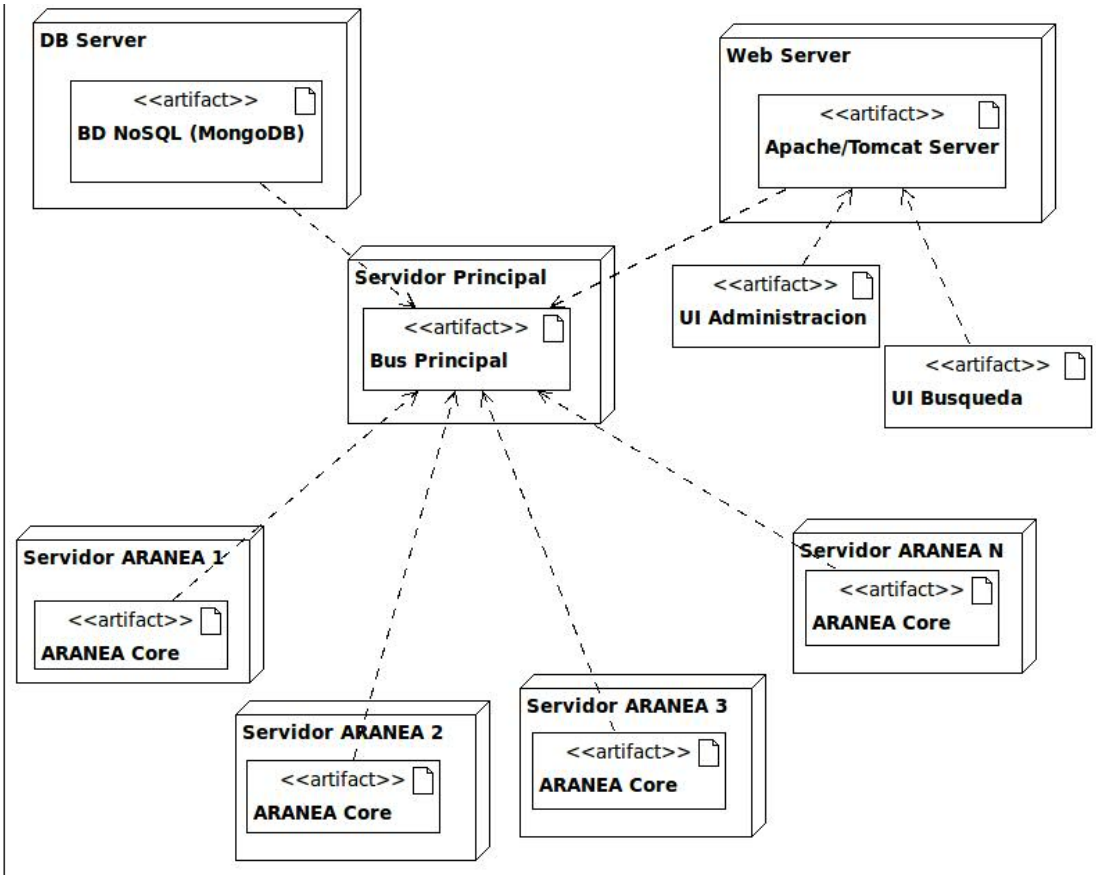

Figura No. 45. **Diagrama de Despliegue con ARENA Core en Varios Servidores.** El Investigador.

En última instancia, la arquitectura orientada a componentes proporciona un mecanismo para dividir un gran conjunto de funciones en un conjunto coherente de componentes de ejecución e ilustra las interacciones entre ellos. Por tal motivo, en la Fase III de implementación, donde se desarrollará cada componente se hará uso de los diagramas de interacción y de actividad para una mayor comprensión.

Infraestructura y Herramientas Tecnológicas

Uno de los factores que se consideraron para la selección de Nutch como software base, es su fundamento en Java; debido a los beneficios ya mencionados, recalcando la necesidad de que el sistema funcione en distintas plataformas de

sistemas operativos; además cuenta con una gran cantidad de Frameworks que pueden facilitar el desarrollo.

Un framework es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, en base a la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Para el desarrollo de ARANEA se usaron una serie de frameworks, los cuales son descritos a continuación y para qué serán usados.

Cuadro No. 29
Frameworks Usados en ARANEA

Framework	Uso dentro de ARANEA
Tika	Es framework de código abierto que ayuda al análisis de contenido, extrayendo enlaces y metadata; por lo cual se usará en el componente “Parser”.
ElasticSearch	ElasticSearch es un indexador de datos; basado en Lucene, lo que permitirá búsquedas eficientes. Edemas provee una interfaz REST para la comunicación.
Mule	Es el más robusto y maduro de los sistemas de bus de integración de código abierto.
ActiveMQ	Framework para el manejo de colas síncronas y asíncronas. Fácil de usar
MongoDB	Almacén de Datos NoSQL, seleccionado de acuerdo a los beneficios antes mencionados
JSP	Por su sencillez y estar basado en JAVA, para la creación de las interfaces de usuario, como la de administración y la de consulta

Fuente. El Investigador.

Entre los componentes se hará uso de REST como protocolo de comunicación. La interacción de las dos interfaces a desarrollar con el sistema se realizará a través de un navegador, usando HTTP como protocolo de comunicación y JSP como lenguaje de desarrollo.

Una de las características de XP, es que en cada iteración de desarrollo, el componente debe ser probado, es por ello, que se contará con JUNIT como herramienta de prueba y verificación de los componentes desarrollados.

Diseño de Criterios de Confiabilidad Para Repositorios

La Web ofrece una gran riqueza de información, así como la oportunidad para la gente de expresarse e intercambiar ideas; de modo que los documentos pueden exponerse fácilmente, de manera gratuita, no regulada o no controlada. Por lo que según la Universidad de British Columbia afirma que "Cuando se usa una fuente Web para realizar un trabajo de investigación, es necesario desarrollar habilidades para evaluar la credibilidad y la pertinencia del contenido" (2011, párr. 1); allí radica la importancia de establecer algunos criterios de evaluación, con el fin de ayudar a determinar en lo posible la calidad del contenido de un repositorio.

En este sentido, existen numerosos estudios realizados, en gran parte por Universidades, que han producido diversas guías y listas de verificación disponibles como apoyo en la evaluación de los recursos de Internet. El presente diseño de una guía que ayude al administrador de ARANEA a diagnosticar la calidad y confiabilidad de un repositorio; no pretende ser exhaustivo ni evaluativo, sino una recopilación documental de tres estudios realizados por Universidades, cuyo criterio de selección fue la similitud entre estudios, soporte actual del proyecto e importancia de uso del criterio de evaluación; lo que en conjunto proporcionará una visión general de los principales aspectos que deben considerarse al evaluar un recurso Web.

El siguiente *Cuadro No. 30*, representa el resultado del análisis y recopilación de los trabajos realizados por la Universidad de Berkeley, Universidad de British

Columbia y la librería de Virginia Tech que a su vez están basados en los criterios publicados por Research Library Group. En el mismo se presentan varios ítems o características que poseen criterios de evaluación, junto con la razón de ser del mismo y como se puede evaluar el criterio; lo que según la Universidad de Berkeley (2011) “al evaluar estos criterios, podrá ayudar a determinar si una página Web es un recurso adecuado para un trabajo de investigación, o no” (párr. 1).

Cuadro No. 30
Propiedades y Criterios de Calidad Para Evaluar Confiabilidad de Repositorios

Propiedad	Criterios a Evaluar	Razón de Ser	Como Evaluar
Autoría	✓ ¿Está el autor claramente identificado? ✓ ¿El repositorio pertenece a un grupo, organización, institución, corporación o gobierno? ✓ ¿Está claro quién es el responsable de la creación y mantenimiento del repositorio? ✓ ¿El publicador tiene una reputación de fiabilidad? ✓ ¿Está disponible la información de contacto del repositorio?	• A menudo es difícil determinar la autoría de una página web. • A diferencia de los recursos tradicionales; los recursos web rara vez tienen editores o chequeadores. • No hay estándares para la información en la web que garantice que toda la información es precisa y útil. • Si no se puede encontrar un autor o una organización responsable de un sitio web es muy sospechosa su confiabilidad.	• Buscar en la parte superior e inferior de la página web. • Utilizar el servicio Whois para determinar el propietario de la página. • ¿Existe un vínculo a un sitio web principal que respalde el repositorio? • Buscar en la primera parte de la URL de la página web. Si es Org? . Edu? . Gob? . Red? . Com? • Buscar dentro del repositorio un enlace que determine quiénes son y cuál es su misión o la filosofía • Indagar en otros enlaces información sobre el autor.

Propiedad	Criterios a Evaluar	Razón de Ser	Como Evaluar
Cobertura	✓ ¿Es la información relevante para el alcance? ✓ ¿El repositorio contiene información que no se encuentra en otro sitio? ✓ ¿Qué tan reflexivo es el material? ✓ ¿Existe indicaciones de que el repositorio está completo y no en construcción? ✓ ¿Está claro que el material del repositorio es un trabajo completo o es una porción de algún trabajo impreso?	• La cobertura de Internet a menudo difiere de la cobertura de impresión. • Con frecuencia es difícil determinar el alcance. • A veces la información del repositorio es publicada sólo por diversión. • Si existe algún indicio de que la página está todavía en construcción, puede ser mejor no utilizarlo. • A menudo cuando un repositorio es solo una parte de un trabajo impreso, la información contenida está fuera de contexto o es menospreciada	• Leer el repositorio minuciosamente. • Verificar alguna información con referencias bibliográfica en otros lugares.

Propiedad	Criterios a Evaluar	Razón de Ser	Como Evaluar
Objetividad	✓ ¿La información muestra un mínimo de sesgo? ✓ ¿Es la página una presentación de los hechos y no está diseñado para influir en la opinión? ✓ ¿Se encuentra el repositorio libre de anuncios o enlaces patrocinados?. Si no, ¿Existe una clara separación del patrocinio con el contenido? ✓ ¿Es claro y explicito el objetivo de publicación del repositorio? ✓ ¿Utiliza lenguaje violento y provocador?	• Con frecuencia, los objetivos de los patrocinadores y los autores no están claramente establecidos. • El contenido de la página puede estar influenciada por el anunciante. • Examinar si los patrocinadores de los anuncios podría haber patrocinado la investigación publicada. • Sea cauteloso y escéptico de que el contenido de la página es sin prejuicios.	• Leer el repositorio minuciosamente. • Verificar algún enlace donde se encuentre establecido la misión y filosofía del repositorio. • Observar que otros sitios Web enlazan al repositorio. • Prestar atención como se presenta la información; si existen opiniones con claridad o son sesgadas para ayudar a algún anunciante en particular.
Exactitud	✓ ¿Es la información libre de errores? ✓ ¿Hay un editor o alguien que	• Cualquiera puede publicar cualquier cosa en la Web. • A diferencia de los recursos	• Leer el repositorio minuciosamente. • Verificar la Información

Propiedad	Criterios a Evaluar	Razón de Ser	Como Evaluar
	verifica y/o comprueba la información? ✓ ¿Es un repositorio de publicación de expertos? ✓ ¿El contenido posee fuentes bibliográficas?	tradicionales; los recursos web rara vez tienen editores o revisores. • En la actualidad, no existen estándares para la Web para asegurar la exactitud.	importante en otros sitios Web.
Vigencia	✓ ¿Se puede encontrar última fecha de actualización? ✓ ¿Los documentos poseen fecha de creación? ✓ ¿Existen indicios de que el material es actualizado frecuentemente? ✓ ¿Los enlaces están funcionando y señalan a páginas existentes?	• Las fechas de publicación o de revisión no siempre se proporcionan. • Las páginas con enlaces rotos no se puede actualizar periódicamente. • La información cambia constantemente; por lo que saber la fecha de actualización es relevante y la misma tendrá coherencia.	• Leer y analizar el repositorio para ver si el autor atribuye la información y/o datos a un año determinado. Por ejemplo "En 1997, 35 accidentes de tráfico fueron causado por los pollos que cruzan la carretera." • Analizar las citas bibliografías o una lista de referencias. Y ver la vigencia de las mismas. • Observar el pie de página, la

Propiedad	Criterios a Evaluar	Razón de Ser	Como Evaluar
		• Si los enlaces a otras páginas no funcionan, es un indicador de que el repositorio no es mantenido.	fecha de actualización del sitio.
Propósito	✓ ¿Cuál es el propósito principal de la página? ¿ Es para vender un producto? ¿Para hacer un punto de vista político? ¿Para divertirse? ✓ ¿Es el sitio de un recurso completo o se centran en una gama estrecha de la información? ✓ ¿Cuál es el énfasis de la presentación? Técnica, científica, clínica, popular, primaria, etc.	• El propósito puede determinar la calidad del contenido, dependiendo del tópico de búsqueda. Existirá un sesgo en la información si el propósito es la venta de un producto, o si el punto principal es divertirse. • También la información puede sesgarse si un sitio o página no es exhaustiva, y se concentra en una estrecha gama de información.	• Leer y analizar la información del repositorio. • Verificar si existe un enlace sobre el objetivo del repositorio. • Examinar la completitud del los contenidos.

Fuente. El Investigador.

Fase III: Fase de Implementación del Motor de Búsqueda

En esta fase se desarrollarán o se adaptarán los componentes del motor de búsqueda. Donde se ejecutarán las siguientes iteraciones de acuerdo a la planificación de entregas planteadas en la metodología XP y propuestas en la fase anterior. En cada iteración se resolverán los casos de uso planeados y se mostrarán los diagramas de secuencia, que dará una perspectiva amplia del proceso involucrado y un diagrama de actividad que proveerá una vista más detallada en forma de seudo-código.

Iteración No 1

Objetivo

Desarrollar la Interfaz y servicio para la Agregación de Semillas y Parámetros de Configuración. En esta iteración se resolverá el caso de uso: Introducir Semillas y Parámetros; como se ve en la *Figura No. 46.*

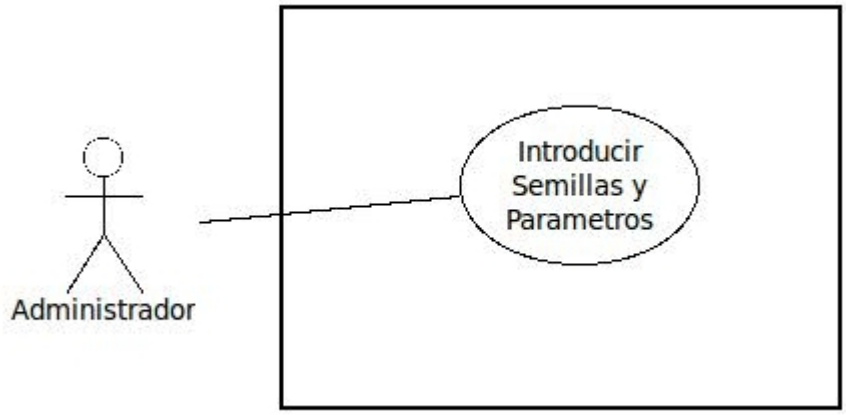

Figura No. 46. **Diagrama de Casos de Uso Introducir Semillas y Parámetros.**
El Investigador.

Desarrollo

Para esta iteración se realizó la interfaz de administración y el componente configurador que contendrá el servicio que recibe los datos de la interfaz y los guardará en una cola asíncrona; para que luego sean tomados por el inyector. La estructura a almacenar en la cola asíncrona es de tipo HashMap el cual contiene dos objetos también HashMap, uno para los parámetros de configuración y el otro las semillas junto con el Id auto-generado de la configuración. En la *Figura No. 47*, se observa la interfaz desarrollada.

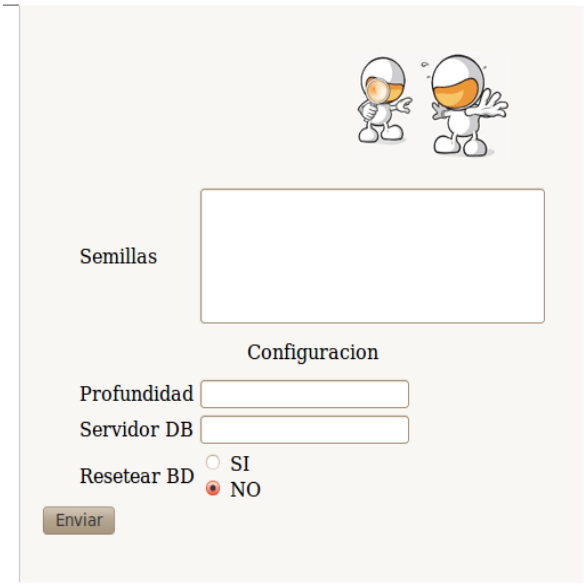

Figura No. 47. **Interfaz Administrativa de ARANEA**.
El Investigador.

Ahora, se muestra en la *Figura No. 48* y *Figura No. 49*, diagrama de interacción y actividad, respectivamente, donde se visualiza la lógica del proceso e intercambio que siguen los elementos involucrados.

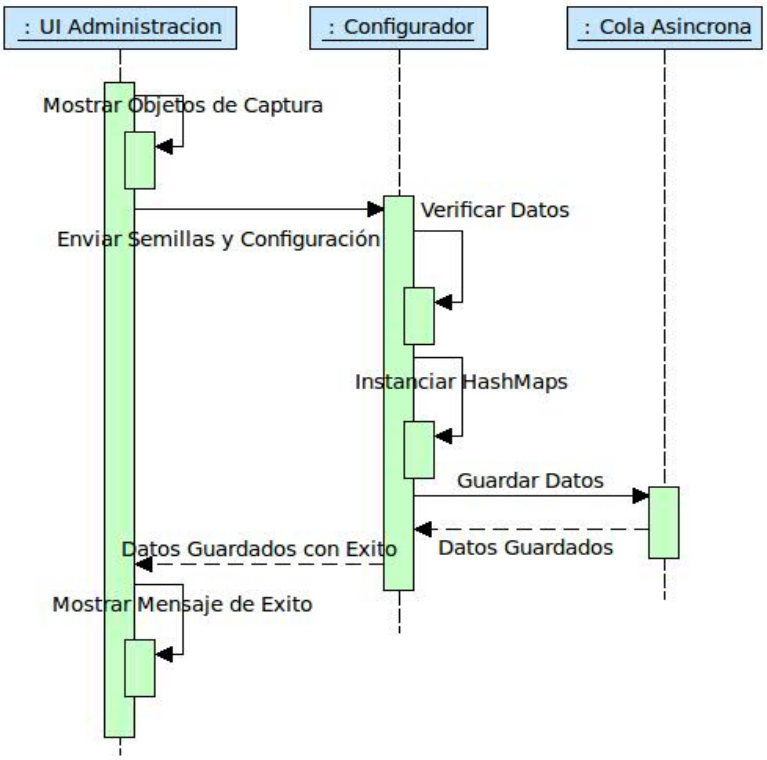

Figura No. 48. **Diagrama de Interacción de Introducir Semillas y Parámetros.** El Investigador.

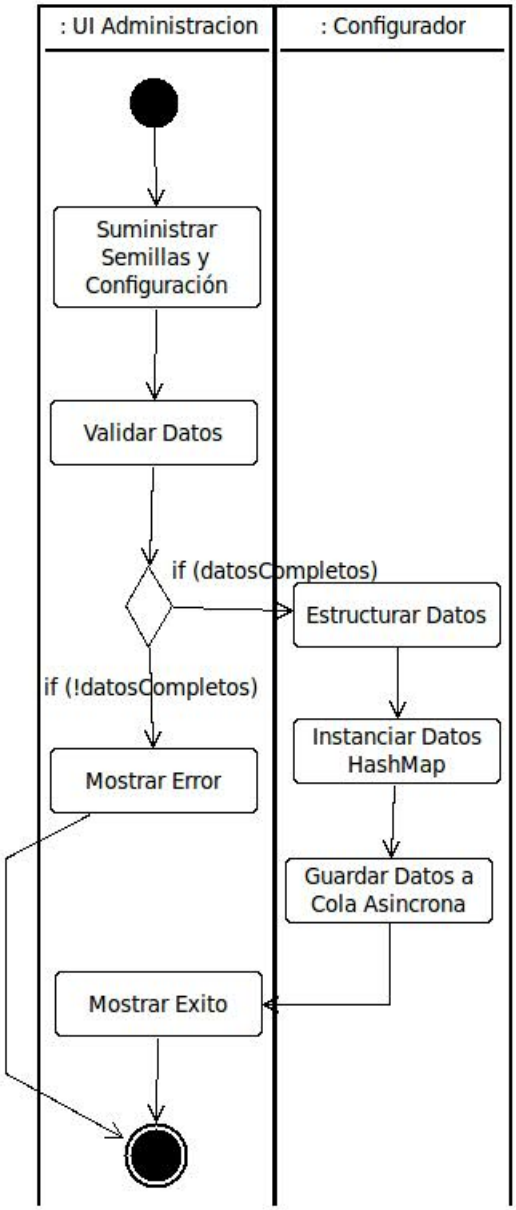

Figura No. 49. **Diagrama de Actividad de Introducir Semillas y Parámetros.** El Investigador.

Por último, no se mostrará el diagrama de clases ya que este caso de uso solo constituye una clase y los requisitos abarcados en totalidad o en parte de esta iteración son: **RF-10.**

Iteración No 2

Objetivo

Adaptar el Componente de Inyección, desde el punto de vista Orientado a Servicio. Donde se desarrollará el caso de uso: Inyectar Semillas, Ver *Figura No. 50.*

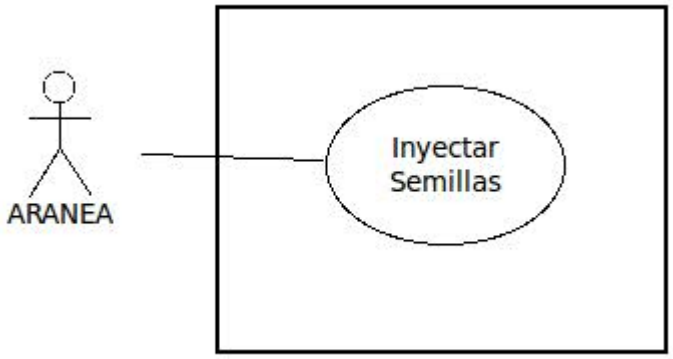

Figura No. 50. **Diagrama de Casos de Uso Inyectar Semillas.**
El Investigador.

Desarrollo

En esta iteración se desacopló el componente Inyector y se desarrolló su servicio REST; además de realizar el servicio de selección y actualización de la BD (Conector BD), que será útil para los demás componentes.

El inyector toma las semillas y la configuración de la cola asíncrona, para luego normalizar los URLs y proceder a crear los objetos inicializados de documento con los atributos establecidos; y por último usa el servicio de inserción en la Base de Datos; esta secuencia de eventos se puede observar en los diagramas de interacción (*Figura No. 51*) y actividad (*Figura No. 52*).

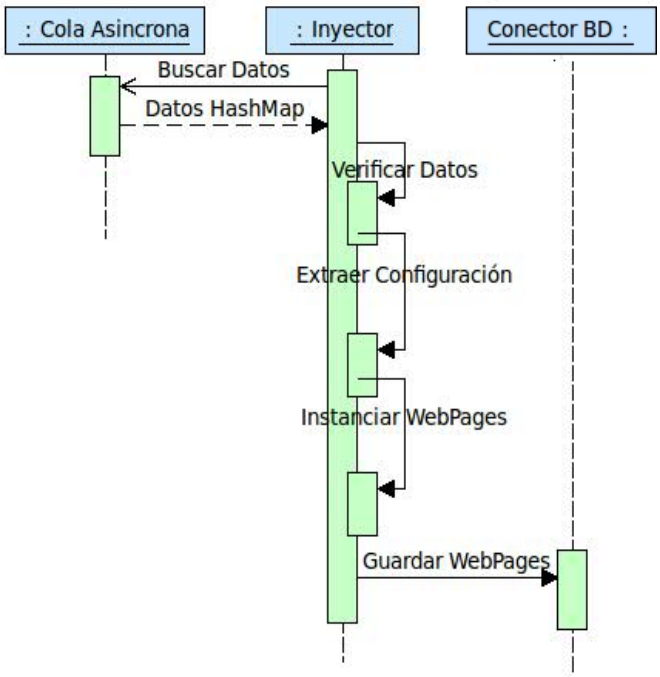

Figura No. 51. **Diagrama de Interacción de Inyectar Semillas.** El Investigador.

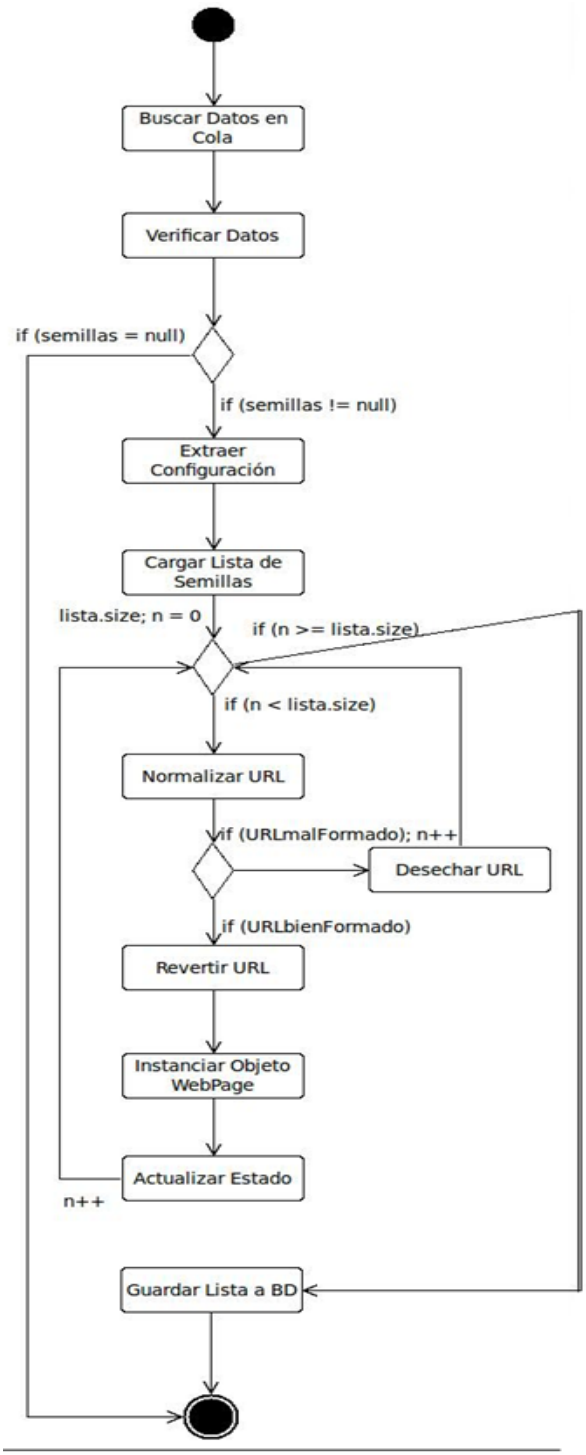

Figura No. 52. **Diagrama de Actividad de Inyectar Semillas.** El Investigador.

Para la adecuación, la clase original del inyector consistía en un Job Thread de Hadoop; lo que se procedió a eliminar esta característica, haciendo una clase sin herencia. También se suprimió la dependencia con MySQL y el orm AVRO. Para hacer uso del servicio de la base de datos y establecer la conexión con MongoDB, creando la clase necesaria con sus servicios para la inserción y selección de datos; así el inyector hace uso del servicio de inserción para almacenar el resultado de este proceso. Además, se creó el código necesario para el acceso a los datos de la cola asíncrona.

El diagrama de clases de aquellas que actúan en este proceso se observa en la *Figura No. 53*.

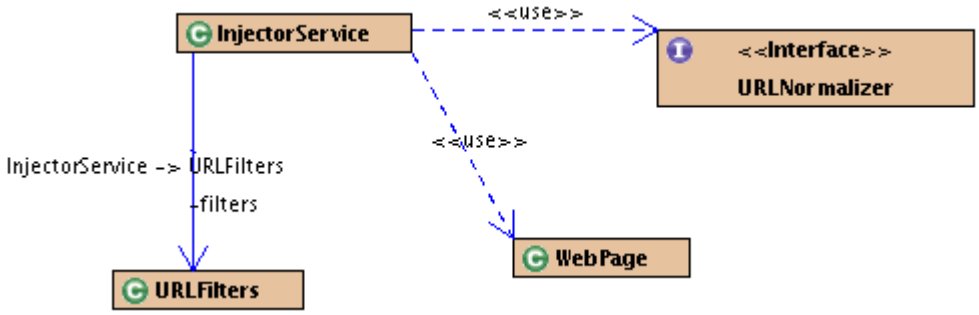

Figura No. 53. **Diagrama de Clases de Inyectar Semillas.** El Investigador.

Los Requisitos abarcados en totalidad o en parte de esta iteración son: **RF-04**.

Iteración No 3

Objetivo

Adaptar el Componente Generador, desde el punto de vista Orientado a Servicio. Para ello se desarrollará el caso de uso: Generar Segmentos. Ver *Figura No. 54.*

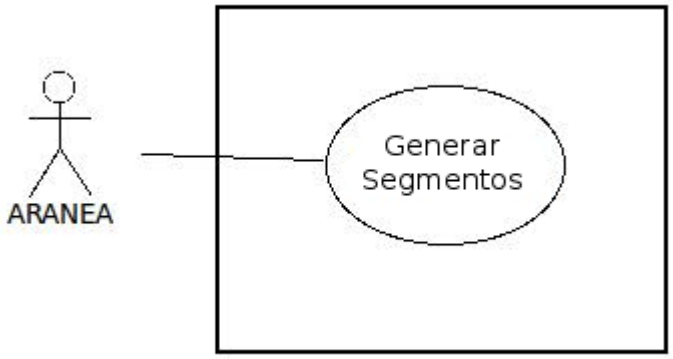

Figura No. 54. **Diagrama de Casos de Uso Generar Segmentos.**
El Investigador.

Desarrollo

En esta iteración se desacopló el componente llamado “Generador”, y elaborar sus servicios REST. En el mismo se hace uso del servicio de consulta de la BD, para tomar todos los datos almacenados y así proceder a estimar el ranking de prioridad (Score) para la extracción del contenido, luego de ello usa el servicio de inserción en la Base de Datos para actualizarlos. Este flujo se puede detallar en la *Figura No. 55*, correspondiente al diagrama de interacción y la *Figura No. 56*, correspondiente al diagrama de actividad.

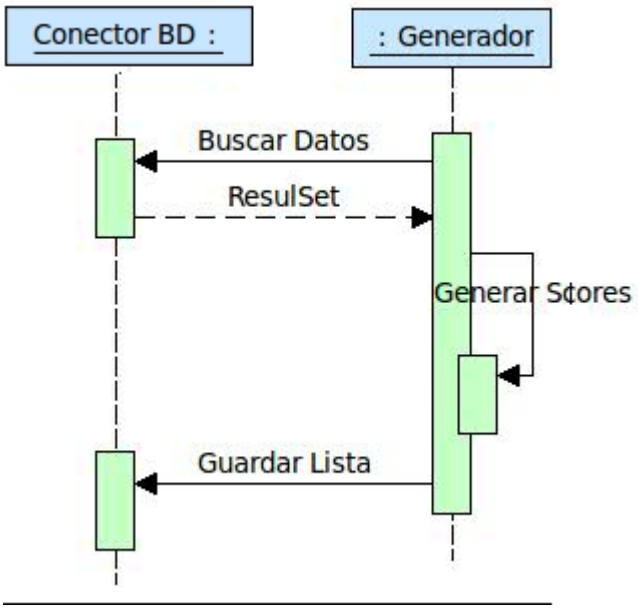

Figura No. 55. **Diagrama de Interacción de Generar Segmentos.** El Investigador.

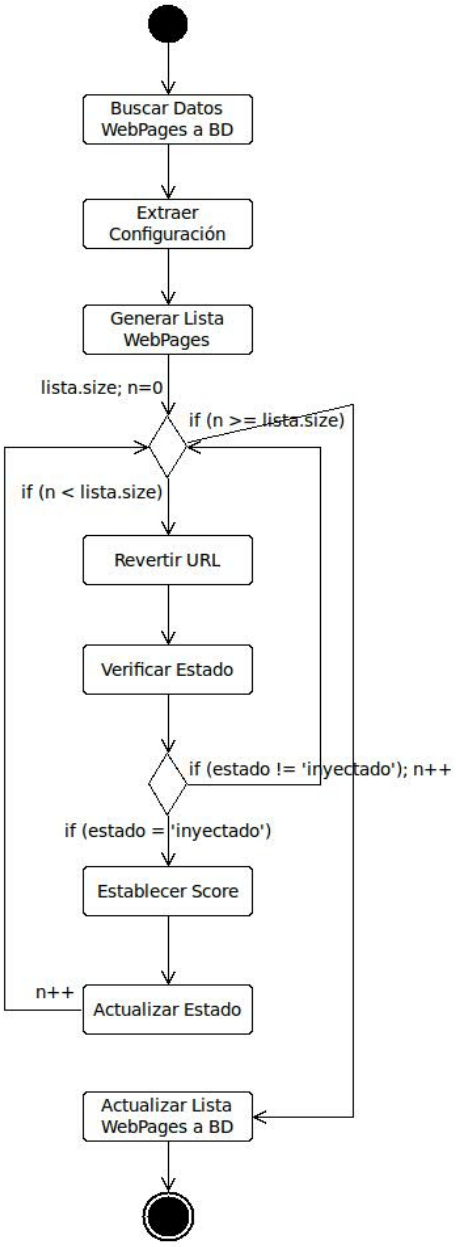

Figura No. 56. **Diagrama de Actividad de Generar Segmentos.** El Investigador.

Al igual que el inyector, el generador también consistía en un Job Thread de Hadoop; por lo que se eliminó esta característica, haciendo una clase sin herencia; y se convino eliminar la dependencia con MySQL y el orm AVRO; haciendo uso del servicio de inserción de MongoDB.

El diagrama de clases que actúan en este procesos se observa en la *Figura No. 57.*

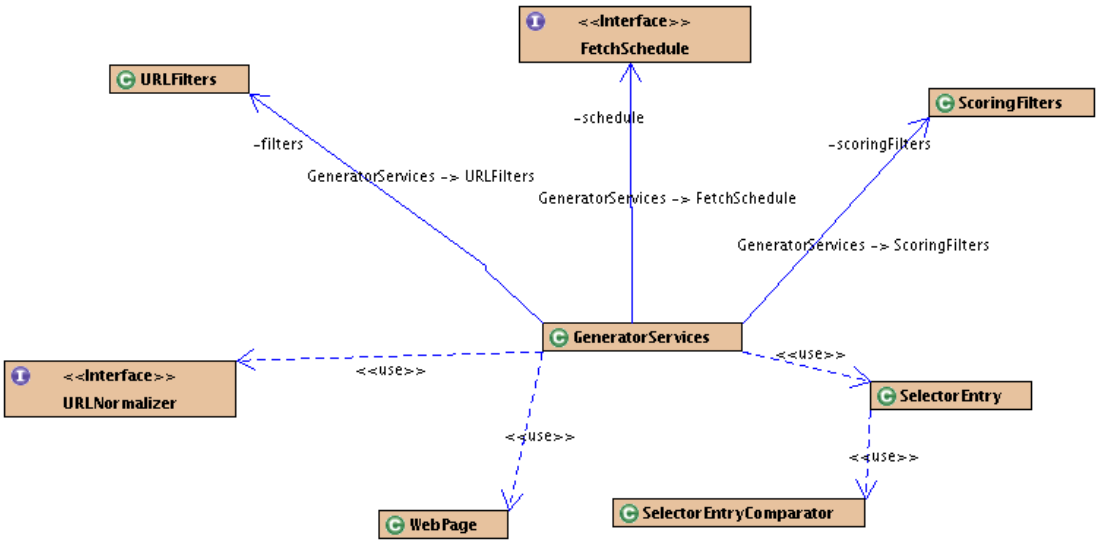

Figura No. 57. **Diagrama de Clases de Generar Segmentos.** El Investigador.

Los Requisitos abarcados en totalidad o en parte de esta iteración son: **RF-18.**

Iteración No 4

Objetivo

Adaptar el Componente de Extracción de Contenido, desde el punto de vista Orientado a Servicio. Para ello se desarrollara el caso de uso: Extraer Contenido. *Ver Figura No. 58.*

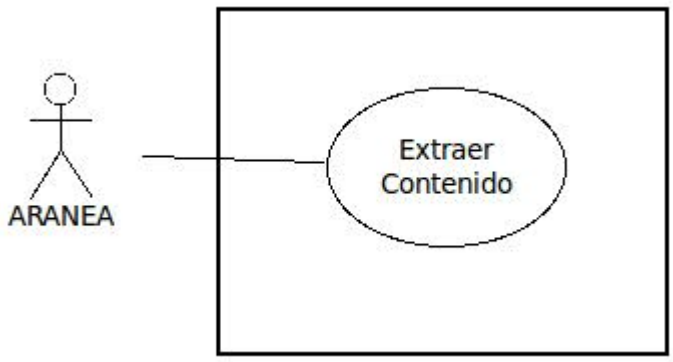

Figura No. 58. **Diagrama de Casos de Uso Extraer Contenido.**
El Investigador.

Desarrollo

En esta iteración se desacopló el componente denominado “Fetcher”, para realizar su servicio; el cual es el corazón central del sistema y el que por ende tiene más carga y se cubrirán una gran parte de los requisitos.

El proceso consume el servicio de consulta de la BD y luego se crean las configuraciones de los agentes. Con los datos extraídos de la Base de Datos, comienza un algoritmo donde por cada entrada se establece un objeto de tipo hilo que se ejecutan en paralelo. Para no sobrecargar los servidores de los sitios Web, el Fetcher establece dos tipos de colas una para cada dominio que se va a accesar y otra que tiene la cola de los dominios (una cola de colas); garantizando que accese solo una vez a cada servidor remoto y no sobrecargarlo.

Luego de establecidas las colas, el Fetcher comienza a extraer los contenidos de los servidores de origen; de acuerdo al orden establecido por el Generador en el proceso anterior. En este proceso se establecen varias restricciones:

- Verifica la exclusión de robots en el dominio.
- Verifica el protocolo.
- Establece un tiempo máximo de espera. De terminar el tiempo, lo descarta y continúa con el siguiente.
- Verifica errores de extracción del protocolo HTTP. Si ocurre, excluye el sitio

y continúa con el próximo.

Una vez terminado, hace uso del servicio de inserción para actualizar los datos en el almacén. Esta estructura del proceso se puede observar con claridad en el diagrama de secuencia de la *Figura No. 59* y el de actividad de la *Figura No. 60.*

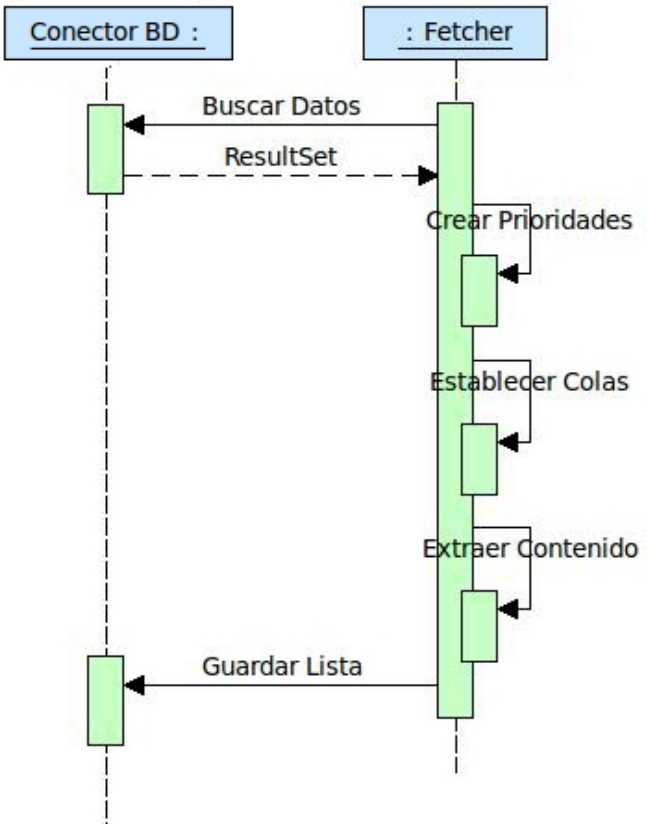

Figura No. 59. **Diagrama de Interacción de Extraer Contenido.** El Investigador.

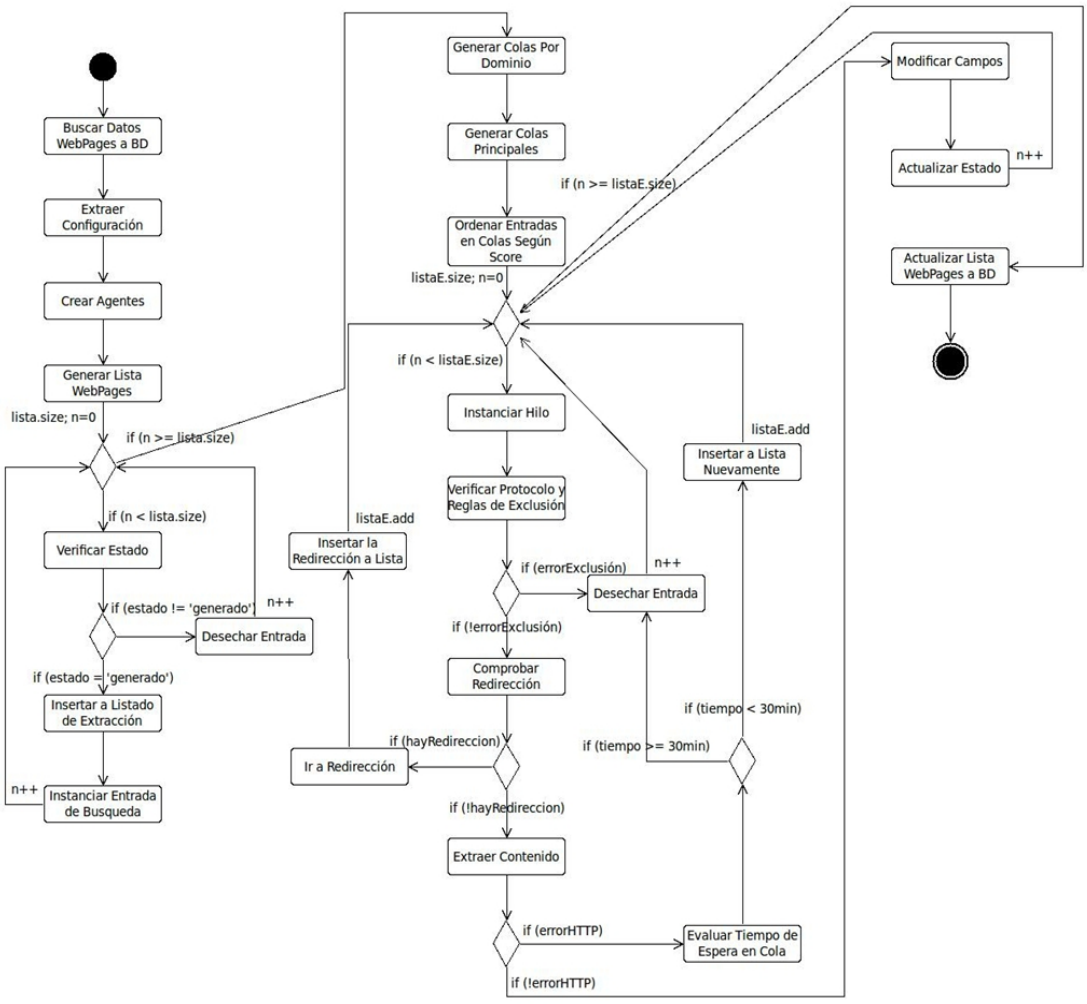

Figura No. 60. **Diagrama de Actividad de Extraer Contenido.** El Investigador.

Al igual que los otros procesos el fetcher consistía en un Job Thread de Hadoop; también se desechó esta característica, haciendo una clase sin herencia. Conviniendo eliminar la dependencia con MySQL y el orm AVRO; haciendo uso del servicio de inserción y selección de la BD.

El diagrama de clases de aquellas que actúan en este procesos se observa en la *Figura No. 61.*

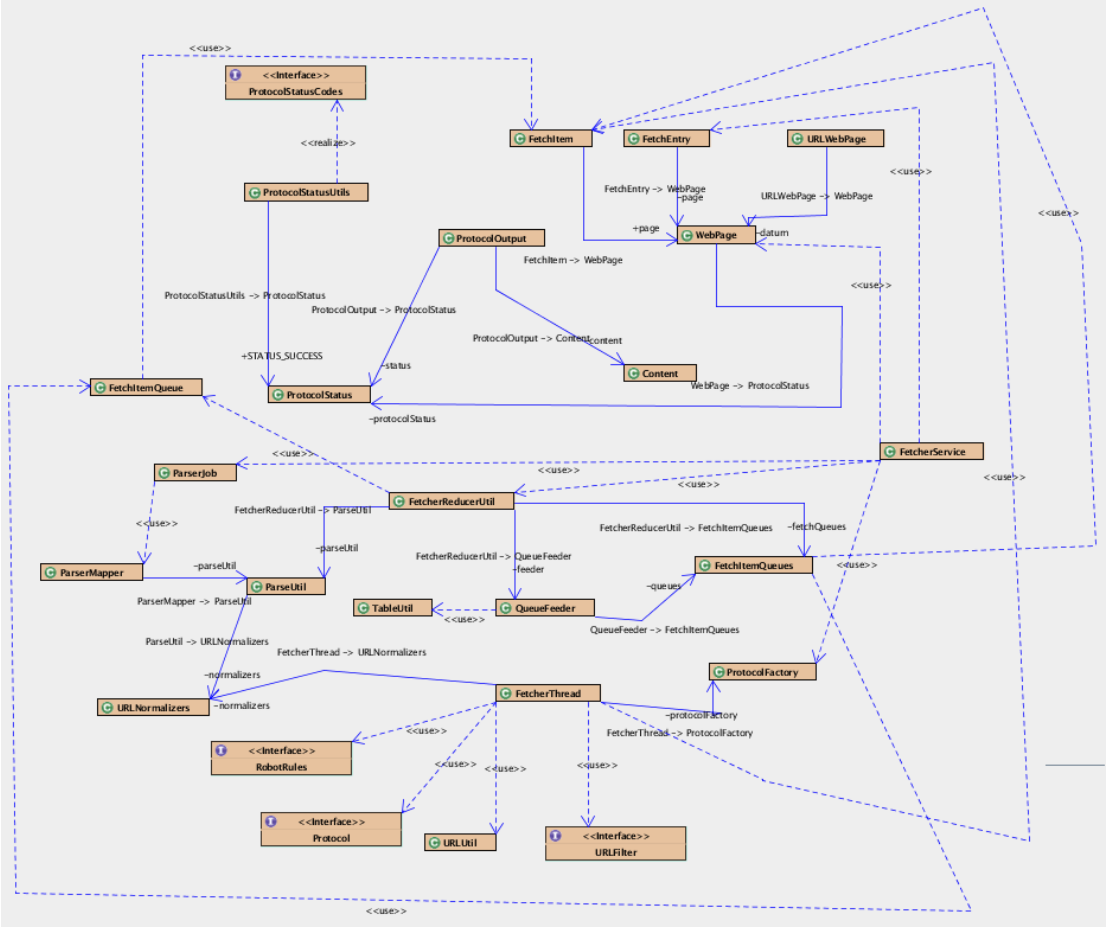

Figura No. 61. **Diagrama de Clases de Extraer Contenido.** El Investigador.

Los Requisitos abarcados en totalidad o en parte de esta iteración son: **RF-01**, **RF-02**, **RF-03**, **RF-06**, **RF-08**, **RF-09**, **RF-11**, **RF-12**, **RF-13**, **RF-14**, **RF-15**.

Iteración No 5

Objetivo

Adaptar el Componente de Análisis de Contenido, desde el punto de vista Orientado a Servicio. Para ello se desarrollará el caso de uso: Analizar Contenido. Ver *Figura No. 62*.

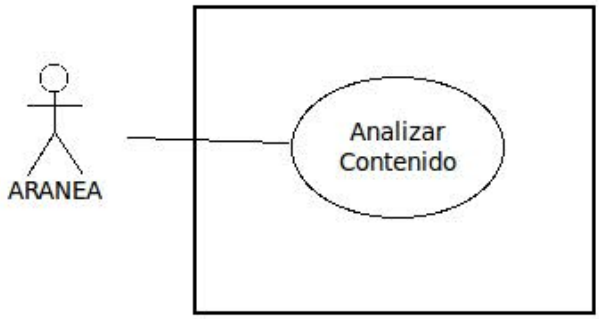

Figura No. 62. **Diagrama de Casos de Uso Analizar Contenido.** El Investigador.

Desarrollo

Para esta iteración se desacopló el componente llamado “Parser”. En este proceso se hace uso del servicio de consulta de MongoDB, para tomar todos los datos almacenados, para luego proceder a analizar el contenido y extraer los enlaces de entrada (Inlinks) y salida (Outlinks), metada, titulo y resumen. Por último se usa el servicio de inserción en la Base de Datos para actualizar los documentos. Este intercambio de mensajes se puede visualizar en los diagramas de la *Figura No. 63* y la *Figura No. 64.*

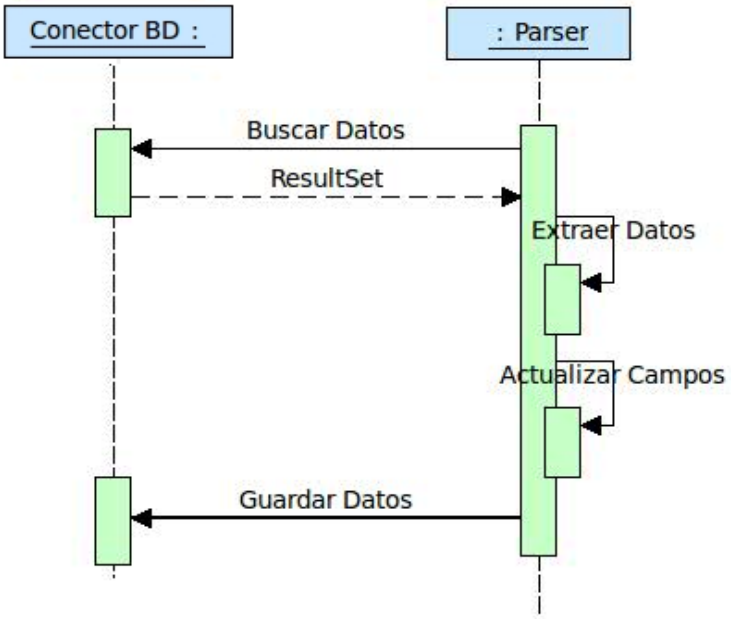

Figura No. 63. **Diagrama de Interacción de Analizar Contenido.** El Investigador.

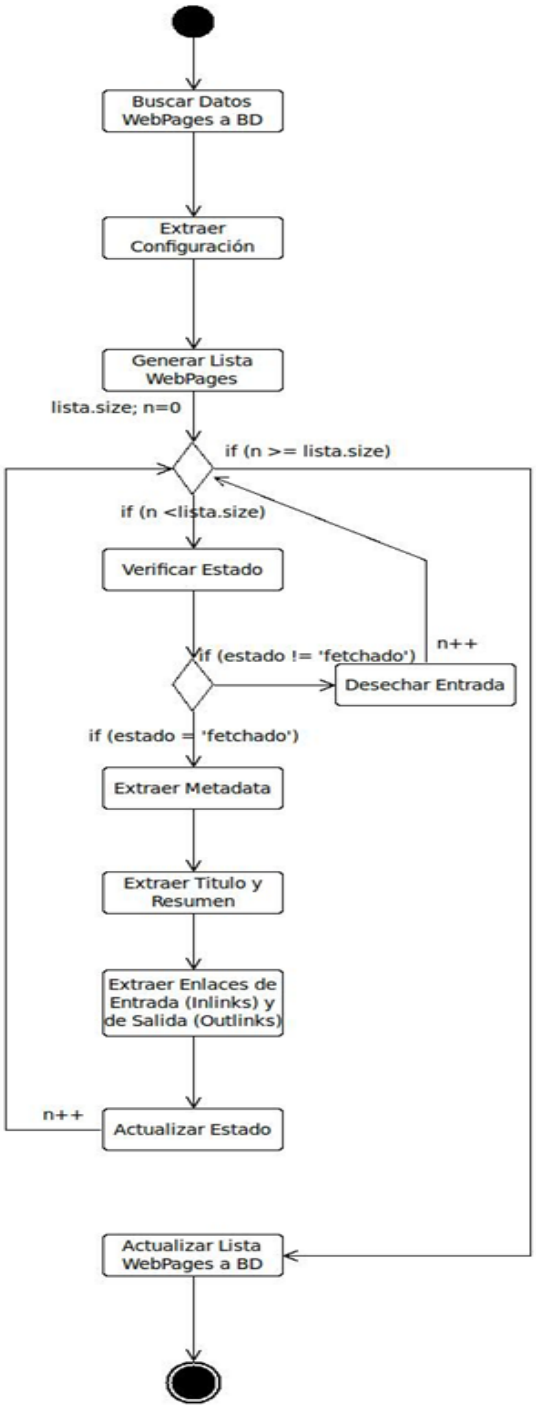

Figura No. 64. **Diagrama de Actividad de Analizar Contenido.** El Investigador.

Como la clase original del generador también consistía en un Job Thread de Hadoop; se procedió a eliminar esta característica, haciendo una clase sin herencia. Así mismo, se acordó eliminar la dependencia con MySQL y el orm AVRO; haciendo uso del servicio de inserción de la BD. Se delegó el trabajo de análisis al framework Tika; que se encargará de la extracción de la metadata, titulo, resumen y determinar los enlaces salientes y entrantes.

El diagrama de clases de aquellas que actúan en este procesos se observa en la *Figura No. 65.*

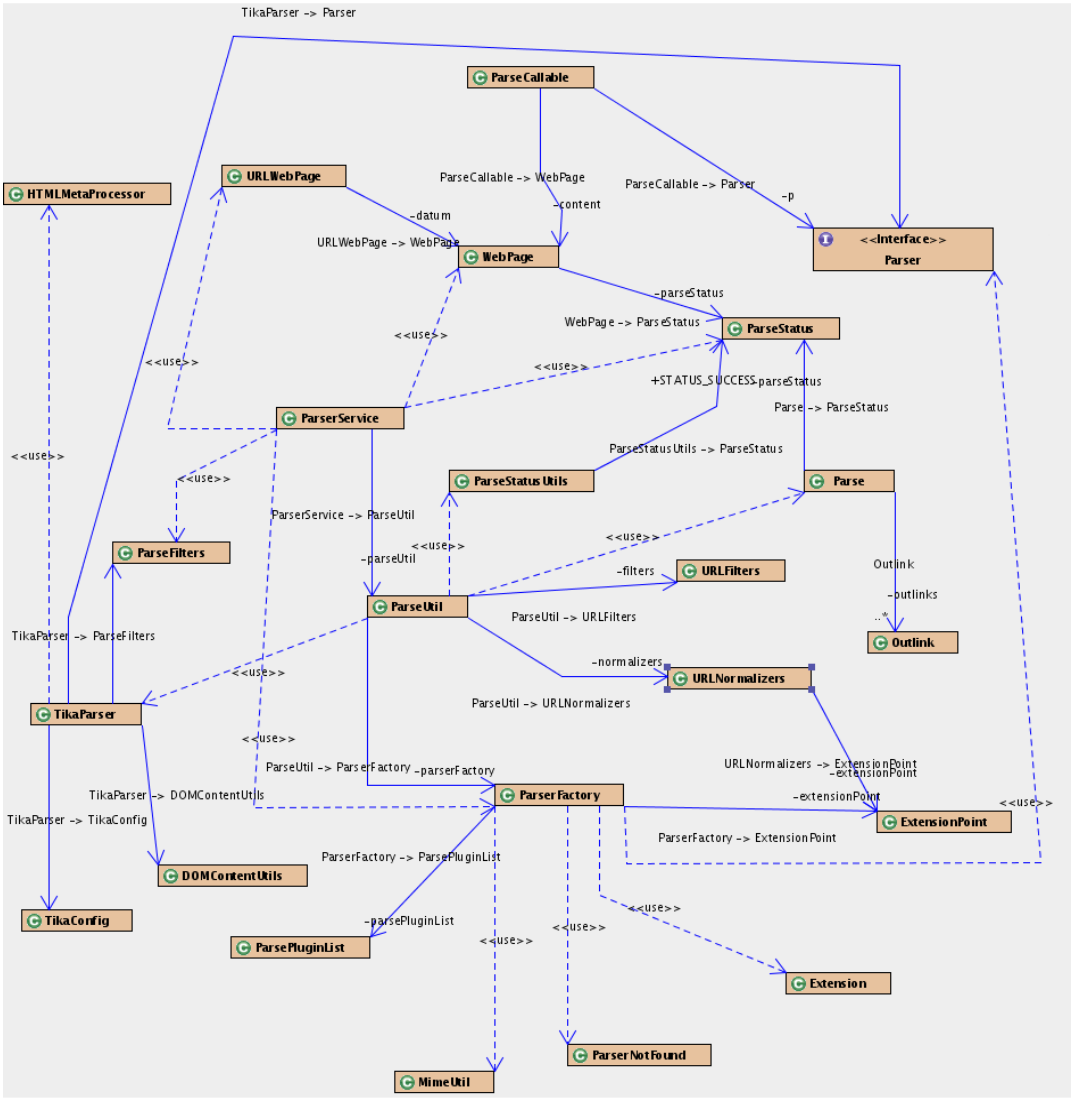

Figura No. 65. **Diagrama de Clases de Analizar Contenido.** El Investigador.

Los Requisitos abarcados en totalidad o en parte de esta iteración son: **RF-15**, **RF-19**.

Iteración No 6

Objetivo

Adaptar el Componente de Actualización, desde el punto de vista Orientado a Servicio. Para ello se desarrollará el caso de uso: Actualizar BD. Ver *Figura No. 66*.

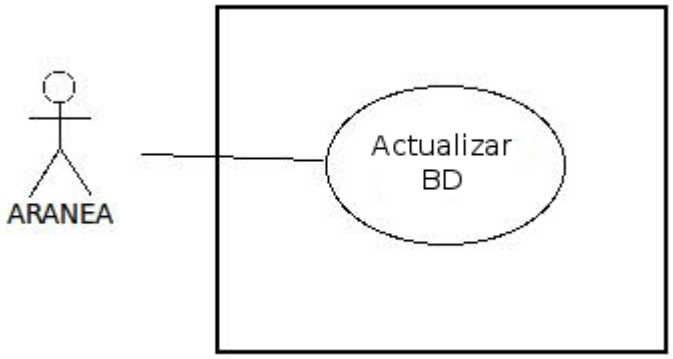

Figura No. 66. **Diagrama de Casos de Uso Actualizar BD.** El Investigador.

Desarrollo

En esta iteración se desacopló el componente llamado “Actualizador”, creando los servicios REST para su comunicación. Este proceso comienza haciendo una consulta con el servicio de la BD; luego por cada entrada de la Base de Datos; este analiza los enlaces nuevos encontrados por el parser, crea los nuevos documentos inicializados de los URLs encontrados y se recalcula el score, actualizando los datos de fecha e intervalo de búsqueda. Por último usa el servicio de inserción en la Base de

Datos para actualizar la data; posteriormente el bus de integración verifica el atributo "profundidad" de la configuración ya que de ello depende que se vuelva a invocar el generador y continuar con el ciclo de los procesos principales de ARANEA. Con los diagramas de interacción y actividades de las *Figuras No 67* y *68*, respectivamente, se puede ver el reflejo de dicho proceso.

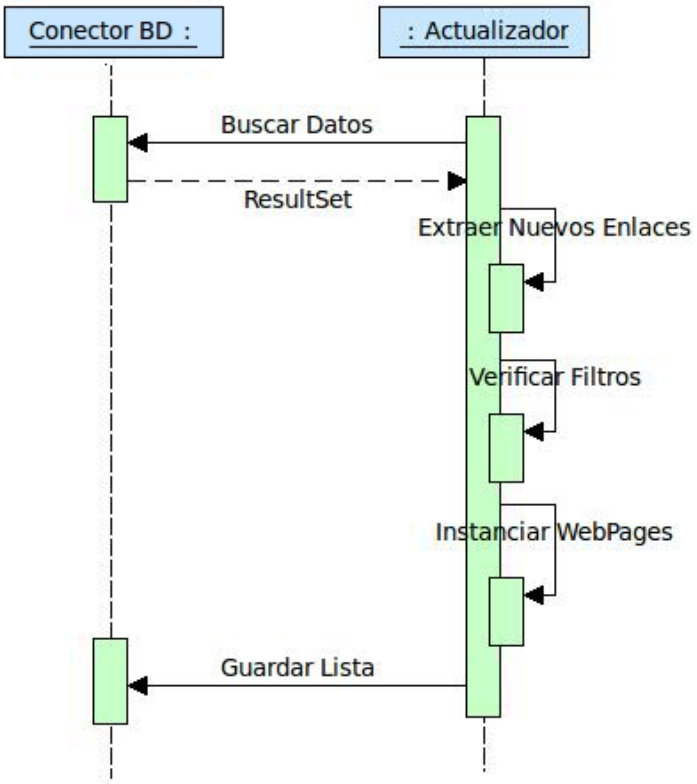

Figura No. 67. **Diagrama de Interacción de Actualizar BD.** El Investigador.

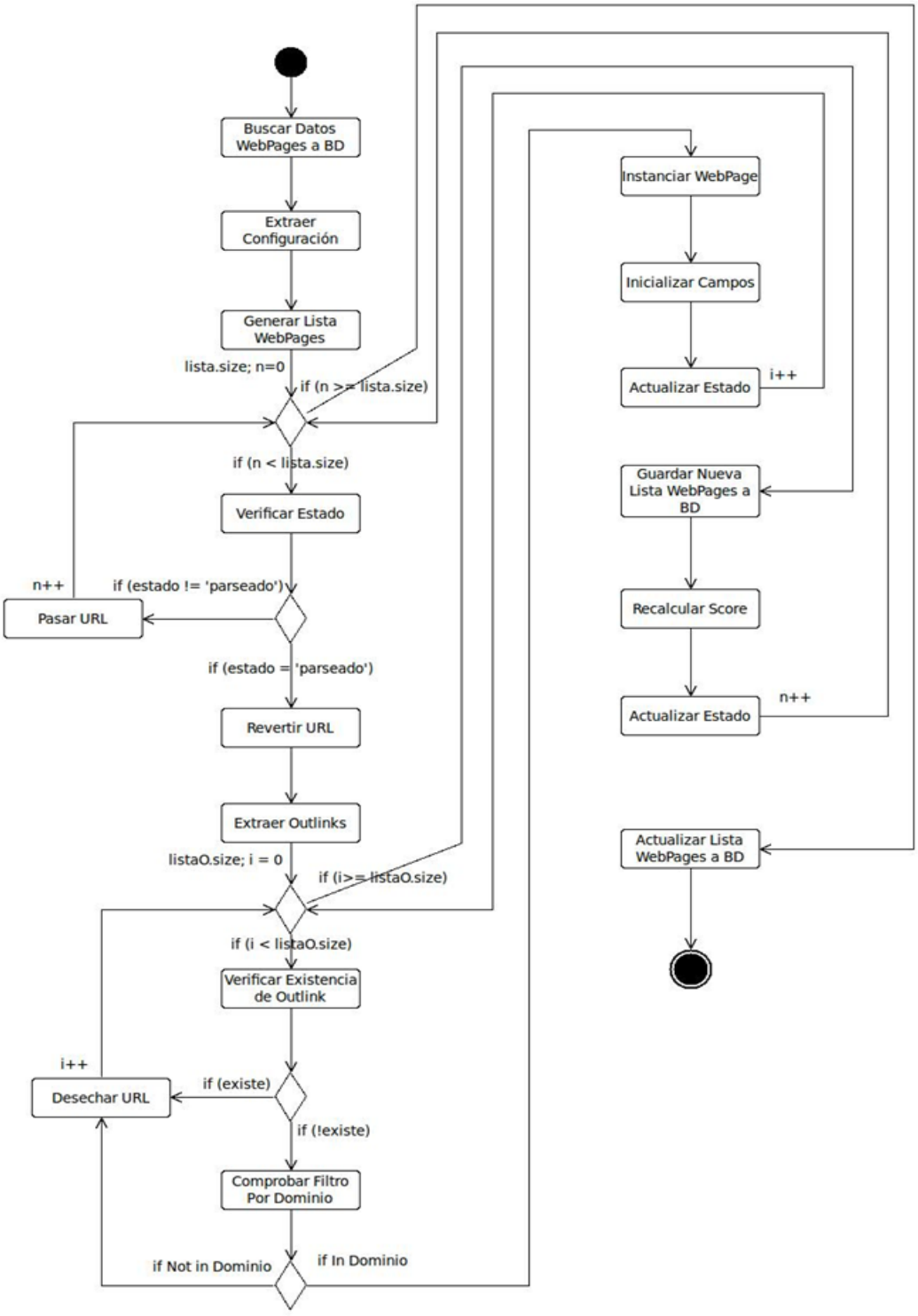

Figura No. 68. **Diagrama de Actividad de Actualizar BD.** El Investigador.

El actualizador se basaba en un Job Thread de Hadoop; y se eliminó esta característica, haciendo una clase sin herencia. Al igual se descartó la dependencia con MySQL y el orm AVRO; haciendo uso del servicio de inserción y selección de la BD.

Las clases principales que interactúan entre sí, se visualizan en el diagrama de clases de *Figura No. 69.*

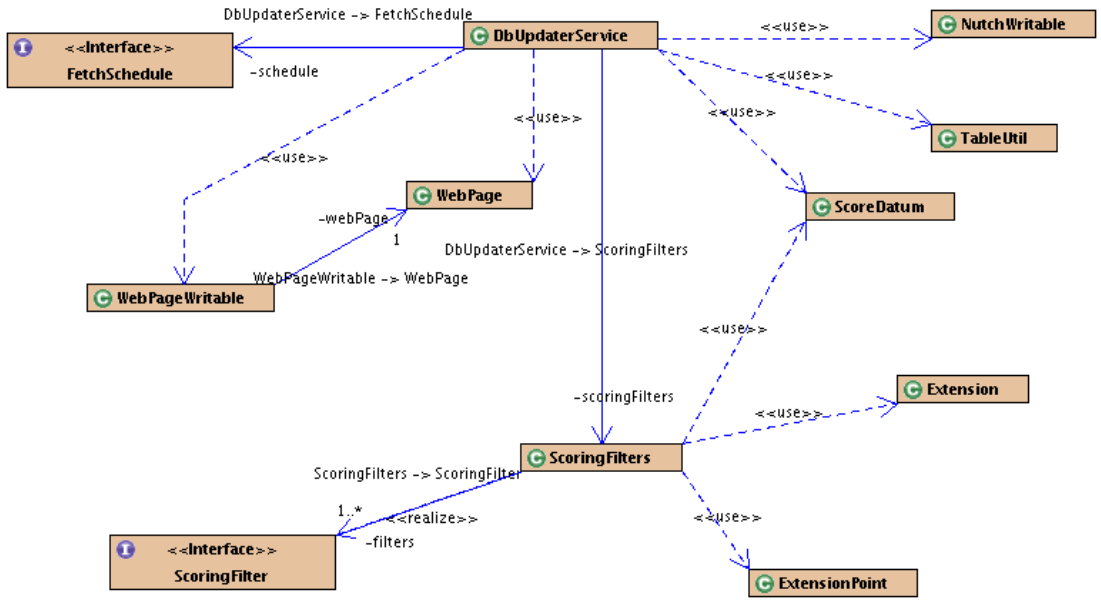

Figura No. 69. **Diagrama de Clases de Actualizar BD**. El Investigador.

Los Requisitos abarcados en totalidad o en parte de esta iteración son: **RF-04**, **RF-05**, **RF-15** y **RF-07**.

Iteración No 7

Objetivo

Adaptar el Componente de Indexación, desde el punto de vista Orientado a

Servicio. Donde se desarrollará el caso de uso: Indexar Contenido. Ver *Figura No. 70*.

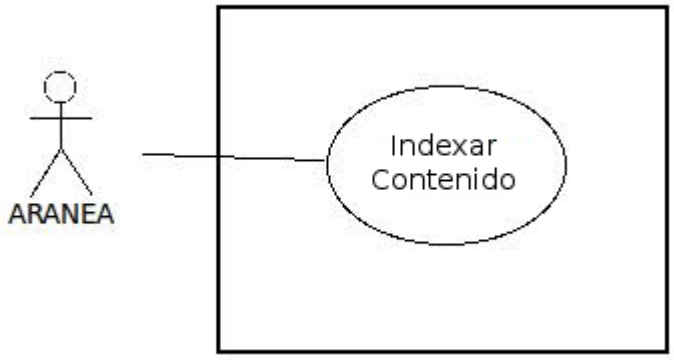

Figura No. 70. **Diagrama de Casos de Uso Indexar Contenido.** El Investigador.

Desarrollo

El componente indexador se ejecuta una vez es cumplido el componente Actualizador. Este proceso hace uso de un índice invertido para facilitar la consulta de datos. Originalmente Nutch, delega este proceso al framework Solr; pero en este estudio la indexación se facultó al framework ElasticSearch, por ser un indexador que trabaja con base de datos NoSQL. La sucesión de las tareas se pueden detallar en los diagramas de interacción y actividad. Ver *Figura No. 71* y *Figura No. 72*.

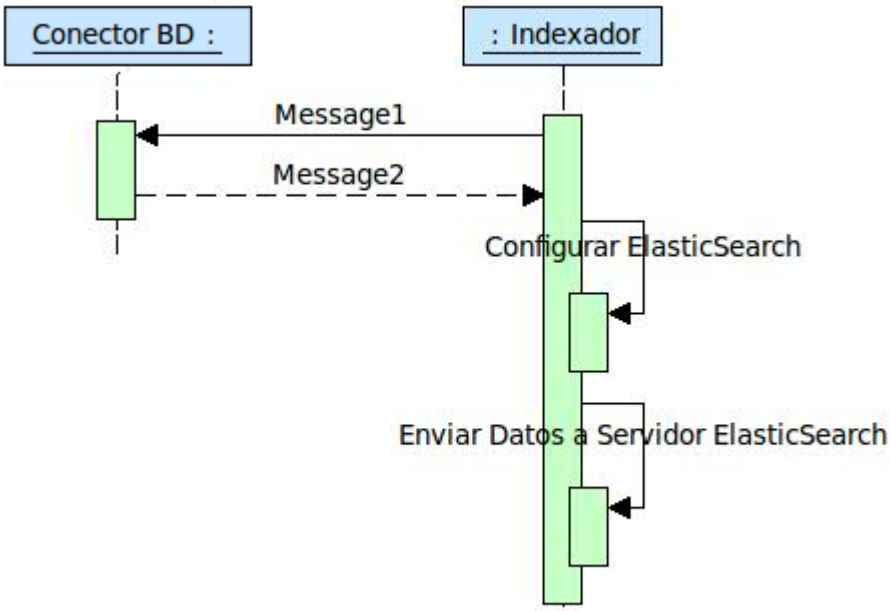

Figura No. 71. **Diagrama de Interacción de Indexar Contenido.** El Investigador.

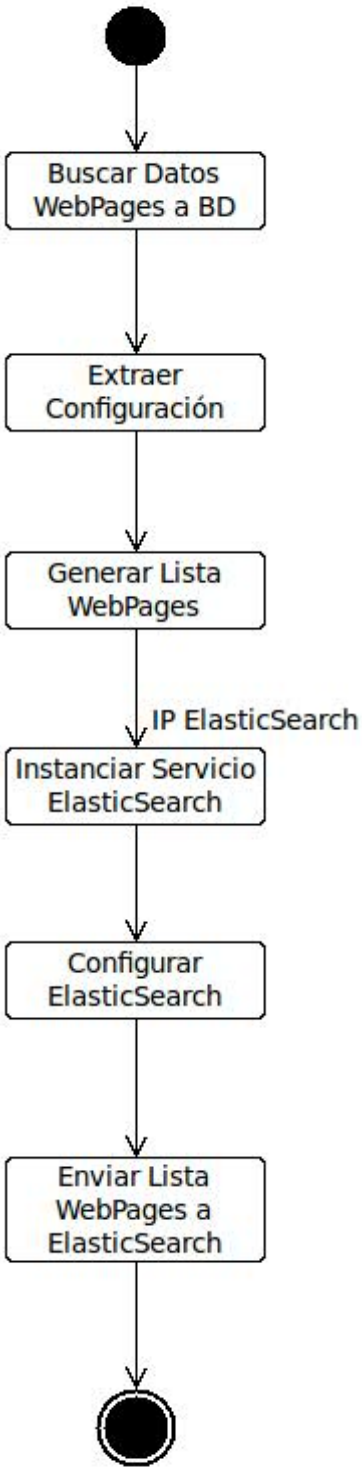

Figura No. 72. **Diagrama de Actividad de Indexar Contenido.** El Investigador.

La clase original del indexador, consistía en un Job Thread de Hadoop; por lo que se eliminó dicha característica, haciendo una clase sin herencia. Para cumplir el objetivo del indexador se eliminó la dependencia con Solr y se creó la comunicación y configuración con ElasticSearch para luego proveerle los datos y que este se encargue del proceso.

En la *Figura No. 73* se puede detallar el diagrama de las clases primordiales de este proceso.

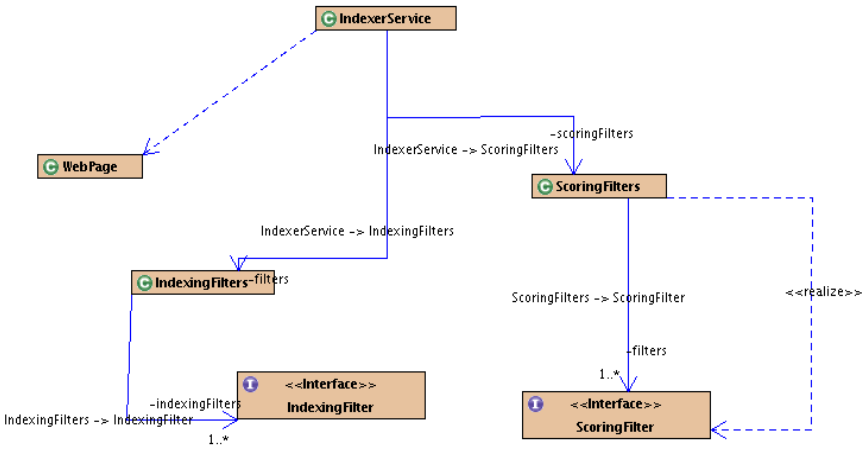

Figura No. 73. **Diagrama de Clases de Indexar Contenido**. El Investigador.

Los Requisitos abarcados en totalidad o en parte de esta iteración son: **RF-15**, **RF-03** y **RF-18.**

Iteración No 8

Objetivo

Configurar el Bus de Servicio ARANEA y Principal, y realizar la orquestación de los componentes.

Desarrollo

Como se propuso en la Fase de Diseño, se prosiguió con la configuración de los dos Bus de Integración para lo cual se hizo uso de Mule; por tal razón se realizaron los archivos de configuración de Mule y los archivos XML. El Bus AEANEA que permite la orquestación de los componentes principales de acuerdo a la secuencia del flujo de ejecución del motor de búsqueda y el Bus Principal, que permitirá la ejecución de las peticiones de administración, consulta de contenido y acceso a la Base de Datos.

En este procesos se cumple el requisito **RF-06**; ya que la orquestación depende de la profundidad del mismo, para saber hasta qué momento se cumple el ciclo de la orquestación de los procesos.

Iteración No 9

Objetivo

Desarrollar el proceso e interfaz de consulta o búsqueda, y resultado del usuario. Para ello se desarrollará el caso de uso: Consultar Documentos. Ver *Figura No. 74*.

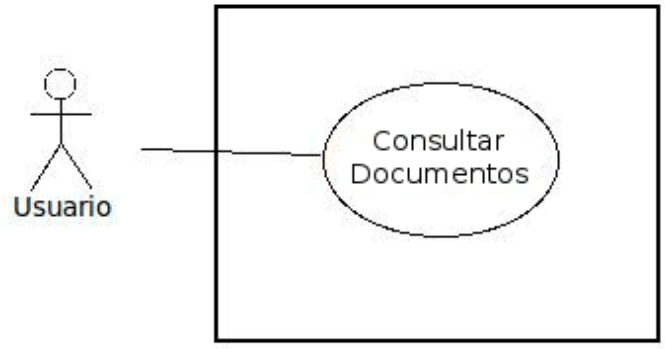

Figura No. 74. **Diagrama de Casos de Uso Consultar Documentos.** El Investigador.

Desarrollo

Para lograr el objetivo se desarrolló el componente Buscador, el cual contiene la lógica y el servicio que será llamado para iniciar la consulta en la Base de Datos, a través del índice provisto por ElasticSearch, según el criterio la palabra clave suministrada; además del servicio que retornará el resultado. También se desarrolla la interfaz realizada en JSP de la búsqueda, que contiene una caja de texto, donde el usuario escribirá la o las palabras claves a buscar y un botón que desencadene la consulta; esta misma interfaz recogerá los resultados para ser mostrados; la misma puede ser vista en la *Figura No. 75* y la *Figura No. 76* con los resultados.

Figura No. 75. **Interfaz de Consulta o Búsqueda.** El Investigador.

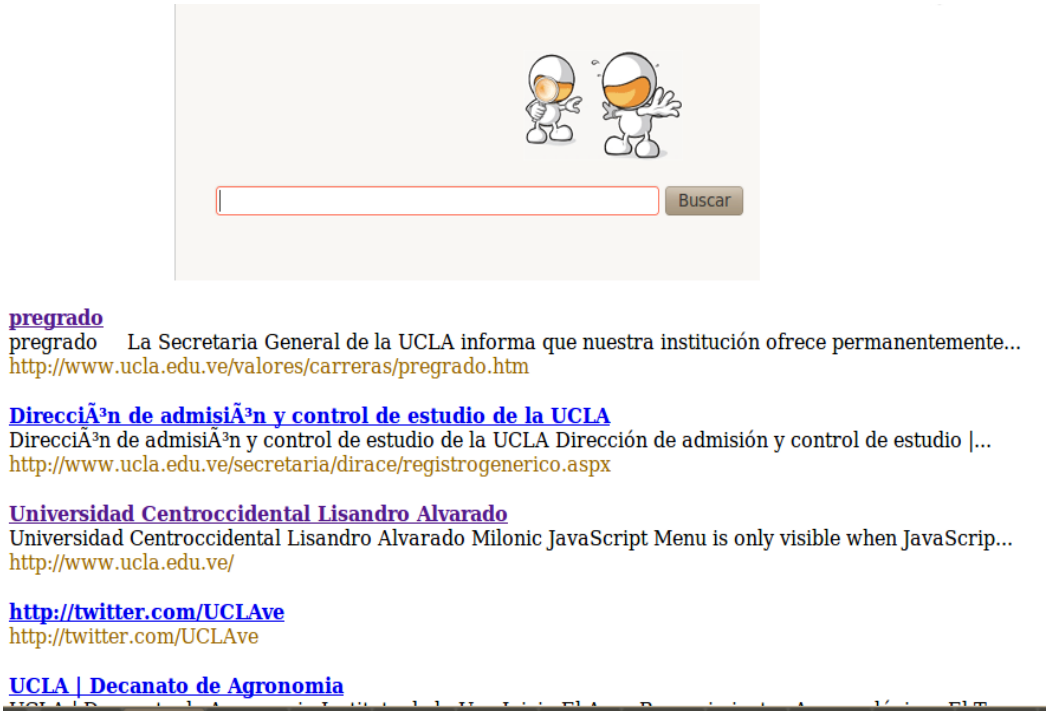

Figura No. 76. **Interfaz de Consulta con Resultados**. El Investigador.

El flujo de las tareas se especifica en los siguientes diagramas de interacción y actividad, representados por la *Figura No. 77* y *Figura No. 78*, respectivamente. En los mismos, un usuario puede acceder a la interfaz de consulta e ingresar en la caja de texto las palabras claves por las que desea buscar algún contenido. Esta interfaz hace uso del servicio de búsqueda, que instancia una comunicación con el indexador ElasticSearch, al cual se le suministra el criterio de búsqueda (palabras claves) y devuelve la entrada de los índices ordenados por relevancia; luego el buscador toma dichos índices y relaciona los URLs de los documentos almacenados; para finalmente retornar a la interfaz el resultado mostrado al usuario.

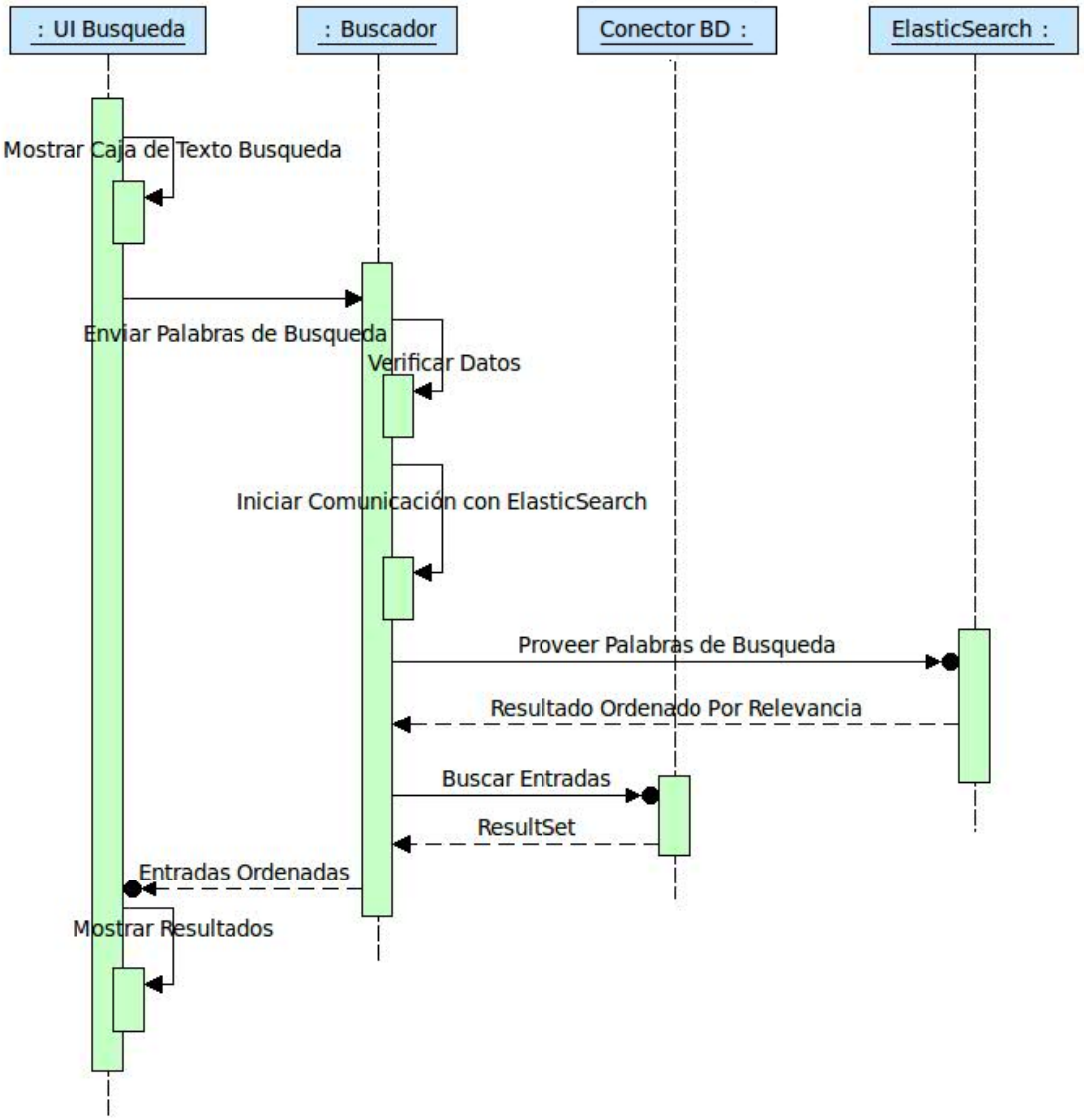

Figura No. 77. **Diagrama de Interacción de Consultar Documentos.** El Investigador.

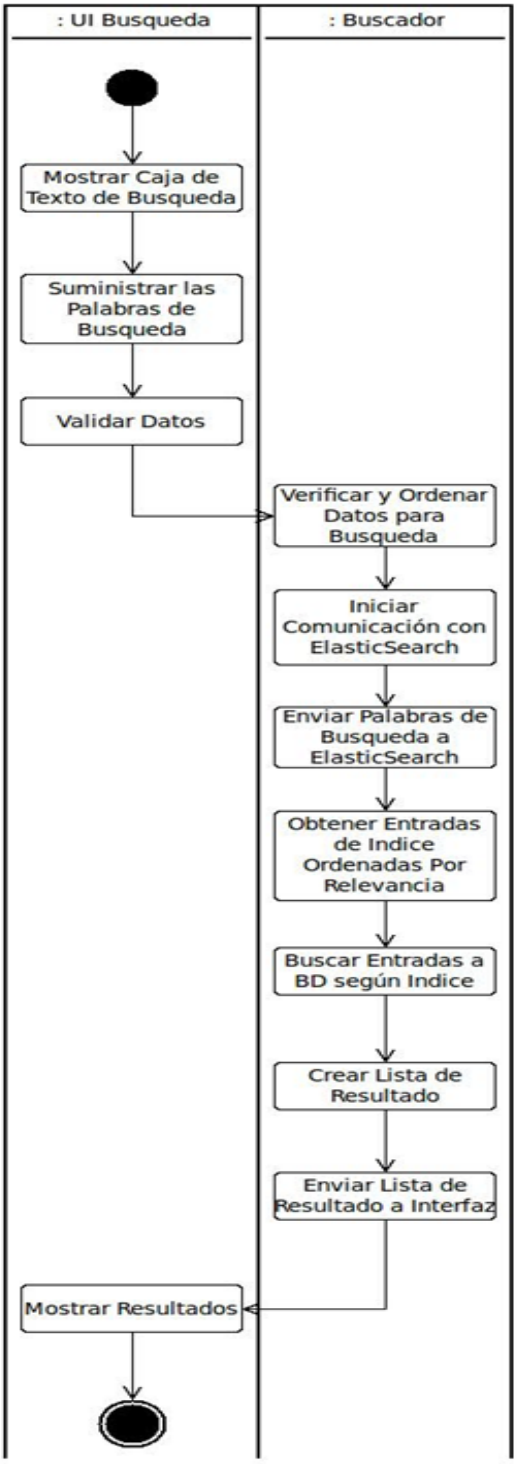

Figura No. 78. **Diagrama de Actividad de Consultar Documentos.** El Investigador.

El diagrama de clases no se mostrara debido a que el procesos solo contiene la clase del buscador.

Los Requisitos abarcados en totalidad o en parte de esta iteración son: **RF-15**, **RF-16** y **RF-18**.

Fase IV: Fase de Validación Tecnológica

En esta fase se realizará una planificación de pruebas de validación y verificación (V&V) de software para determinar errores en la aplicación, así como el cumplimiento de los requisitos funcionales y no funcionales. Según Montilva y Barrios (2007) establecen que este proceso forma parte de la gestión de la calidad, en sus metodología de desarrollo Watch ya que "se asegura la calidad de un software a través del uso de un proceso bien definido de Verificación y Validación (V&V)." (p 12.).

Es importante diferenciar los dos conceptos, ya que tienden a confundir, por ello Boehm (1979, citado por Drake y López, 2009), hace esa diferenciación conceptual.

> **Verificación:** ¿Estamos construyendo el producto correctamente? Se comprueba que el software cumple los requisitos funcionales y no funcionales de su especificación.
> **Validación:** ¿Estamos construyendo el producto correcto? Comprueba que el software cumple las expectativas que el cliente espera (p. 3).

Este proceso persigue dos objetivos: descubrir defectos del sistema y cotejar lo que hace el sistema con la especificación de requisitos. Por ello la verificación y la validación deberían establecer la confianza que el software es adecuado al propósito. Esto no significa que esté completamente libre de defectos, pero si lo suficientemente bueno para su uso y el tipo de uso determinarán el grado de confianza que se necesita; lo cual depende de la especificación de requisitos.

Existen dos tipo de V&V, las dinámicas y las estáticas; entre los métodos más empleados están las inspecciones de software (estática) y las pruebas de software (dinámica). Este estudio solo cubrirá las pruebas de software ya que las inspecciones

de software, según establece el método, deben ser empleadas por un conjunto de expertos que verifiquen el código estático; y como en la investigación lo más importante es comprobar el cumplimiento de los requisitos, se abordará con pruebas de software.

Existen varios tipos de pruebas de software; en la fase de implementación durante el desarrollo se hizo uso de las pruebas de unidad, a través de JUnit en cada iteración. En este apartado se utilizarán pruebas de validación de programa para determinar tanto los requisitos funcionales como no funcionales; en este orden de ideas, existen requisitos que se pueden verificar en tiempo de ejecución, por lo que se empleará la funcionalidad de "depuración" del entorno de desarrollo (IDE), estableciendo puntos de parada (break points) para visualizar los valores necesarios y proceder a validar el requisito.

Primero se realizarán las pruebas por cada caso de uso para examinar sus resultados y por último una prueba de integración. Para determinar los atributos que deben asignarse para cada caso de prueba se siguió las recomendaciones realizadas por Montilva (2009), como se especifican a continuación:

- ***Descripción:*** Una explicación resumida de la prueba.
- ***Requisitos a Comprobar:*** Los requisitos funcionales y no funcionales que son cubiertos por el caso de uso a probar.
- ***Pre-condiciones:*** Estado previo que debe tener el sistema antes de ejecutar la prueba.
- ***Procedimiento:*** Proceso o método a llevar a cabo en la prueba
- ***Casos de Prueba:*** Diferentes alternativa para probar el caso de uso.
- ***Datos de Entrada:*** Por cada caso de prueba, son los datos que se necesita para realizarla.
- ***Resultado Esperado:*** Por cada caso de prueba, de antemano se establece cual será la respuesta del sistema.
- ***Resultado Obtenido:*** Corresponde a una síntesis del resultado final luego de ejecutar la prueba; este puede ser redactado de manera general o por cada caso de prueba.

- ***Acciones:*** Son las actividades a realizar si el caso de prueba no arroja el resultado esperado.

El objetivo de las pruebas siempre será el de conseguir fallos en el producto deliberadamente, por lo que los datos de entrada en algunas ocasiones estarán orientado en ese sentido.

Pruebas del Caso de Uso: Introducir Semillas y Parámetros

Descripción. Esta prueba analizará la interfaz de administración, el componente configurador y la cola asíncrona; por ello se hará uso de dicha interfaz para los datos de entrada.

Requisitos a Comprobar. Se verificará el requisito **RF-10.**

Pre-condiciones. Los procesos de ARANEA, la cola asíncrona, los Buses de Integración, la Base de Datos y la interfaz administrativa deben estar en ejecución.

Procedimiento. El usuario administrador entrará en la interfaz, y colocará los datos sugeridos en cada caso de prueba y dará click en el botón enviar.

Casos de Prueba. Se ejecutarán 3 casos de prueba:

1. ***Datos de Entrada.*** Se introducirán los datos de las semillas pero no de la configuración. ***Resultado Esperado.*** La interfaz debe dar un mensaje que faltan los datos de la configuración.
2. ***Datos de Entrada.*** Se introducirán los datos de la configuración pero no de las semillas. ***Resultado Esperado.*** La interfaz debe arrojar un error de falta de datos.
3. ***Datos de Entrada.*** Se introducirán los datos de las semillas y de la configuración. ***Resultado Esperado.*** Que ambas estructuras se almacenen en la cola asíncrona.

Resultado Obtenido. Los resultados fueron los esperados, por lo que en general fue satisfactoria.

Acciones. Ninguna.

Pruebas del Caso de Uso: Inyectar Semillas

Descripción. Se examinará el componente Inyector; para determinar que extrae los datos de la cola asíncrona e instancia las semillas en la Base de Datos.

Requisitos a Comprobar. Se verificará el requisito **RF-04.**

Pre-condiciones. El sistema debe tener semillas y datos de configuración insertados en la cola asíncrona.

Procedimiento. Se invocará solo al componente inyector desde una unidad de prueba.

Casos de Prueba. Se establecerán tres casos de prueba:

1. ***Datos de Entrada.*** Se ejecutará la clase inyector pero con la cola de datos asíncrona sin datos o con el servicio de colas sin ejecutar. ***Resultado Esperado.*** Se espera que el inyector al no tener datos, no se ejecute.
2. ***Datos de Entrada.*** Se ejecutará el inyector pero con URLs de la semillas mal formadas. ***Resultado Esperado.*** Se espera que el sistema trate de normalizar las semillas, en caso contrario las deseche; para finalmente guardar en la base de datos las URLs que se normalizaron.
3. ***Datos de Entrada.*** Se ejecutará el proceso con las semillas bien formadas. ***Resultado Esperado.*** Se espera que el sistema guarde todos los objetos en la base de datos.

Resultado Obtenido. Los resultados fueron los esperados, excepto en el segundo caso; donde se generó un error a causa de las semillas que no tenían un URL bien formado, haciendo una parada inesperada en el proceso inyector.

Acciones. Surge un nuevo requisito. "**RF-20**: El sistema debe tratar con URL mal formados, tratando de normalizarlos o bien desecharlos". Luego se procedió a mejorar el algoritmo de normalización de URLs, para cumplir con los requisitos que cubre este caso de uso.

Pruebas del Caso de Uso: Generar Segmentos

Descripción. Se hará uso del componente Generador para determinar si este genera los scores de las entradas y actualiza dicho campo en la Base de Datos.

Requisitos a Comprobar. Se verificará el requisito **RF-18**

Pre-condiciones. Deben existir datos almacenados en la Base de Datos.

Procedimiento. Se deben insertar manualmente los datos o ejecutar el proceso de Inyección. En este caso no existen validaciones dentro del proceso, pero si es importante el algoritmo de ordenación de semillas; por lo que la prueba se enfocará en el resultado de este algoritmo.

Casos de Prueba. Para este caso se estableció solo un caso de prueba:

1. ***Datos de Entrada.*** Los documentos guardados en la Base de Datos. ***Resultado Esperado.*** Se correrá el proceso de generación, esperando que guarde en la base de datos los documentos, con el índice de ordenación (score) actualizado.

Resultado Obtenido. Como resultado se obtuvo lo esperado, por lo que se cumple con el requisito.

Acciones. Ninguna.

Pruebas del Caso de Uso: Extraer Contenido

Descripción. Se pondrá a prueba el componente Fetcher, el cual constituye el proceso central del ARANEA.

Requisitos a Comprobar. Por ser el proceso central, es el que más requisitos debe cumplir. En este se verifican los requisitos: **RF-01**, **RF-02**, **RF-03**, **RF-08**, **RF-09**, **RF-11**, **RF-12**, **RF-13**, **RF-14**, **RF-15**.

Pre-condiciones. Deben existir datos en el almacén que posean score. Así,

como datos de configuración.

Procedimiento. Se ejecutará el proceso Fetcher, a través de una unidad de prueba, donde se ejecute solo este proceso y los datos deben ser ingresados manualmente o bien ejecutar el proceso Generador para luego manipularlos de acuerdo a las condiciones de los datos de entrada de los casos de prueba a ejecutar.

Casos de Prueba. Para lograr el objetivo se establecerán 8 casos:

1. ***Datos de Entrada.*** Se ejecutará el fetcher con datos de sitios donde el tamaño sea mayor de 20mb. ***Resultado Esperado.*** Se espera que extraiga el contenido y lo guarde en la base de datos, además de actualizar la fecha del documento.
2. ***Datos de Entrada.*** Se colocaran datos con protocolos FTP y HTTPS. ***Resultado Esperado.*** Se espera el algoritmo deseche este tipo de datos y continúe la ejecución con el resto de los objetos, almacenándolos en la base de datos.
3. ***Datos de Entrada.*** Se establecerán datos con tipos de archivo PDF, HTML, WORD, EXCEL, ODT. ***Resultado Esperado.*** Se espera que el sistema descargue los contenidos sin generar error, guardándolos en la base de datos.
4. ***Datos de Entrada.*** Se insertarán datos de sitios que son redireccionados. ***Resultado Esperado.*** Se espera que el sistema siga la redirección e igual descargue su contenido.
5. ***Datos de Entrada.*** Se hará la corrida con los sitios almacenados, sin anormalidades. ***Procedimiento.*** En el proceso de extracción se debe cortar la conexión del servidor de internet por espacio de 10 minutos. ***Resultado Esperado.*** Se espera que el sistema continúe la ejecución y guarde los objetos pero sin haber actualizado el estatus de extracción.
6. ***Datos de Entrada.*** Se hará una corrida con todos los sitios almacenados sin anormalidades. ***Procedimiento.*** En el momento de creación de los hilos para la extracción de contenidos, verificar a través del depurador el tiempo de extracción de páginas HTML y cuantos hilos en paralelo pueden correr. ***Resultado Esperado.*** Se espera que el tiempo de extracción sea menor a 3 segundos y que se extraigan 10 contenidos al mismo tiempo

7. ***Datos de Entrada.*** Se establecerán datos con URL bien formados pero inexistentes en la WEB. ***Procedimiento.*** Se usará el depurador para observar si el sistema accesa a varios servidores al mismo tiempo o a un mismo servidor; también se determinará el tiempo de frecuencia en que accesa a un mismo sitio. ***Resultado Esperado.*** Se espera que para los sitios inexistente el sistema los deseche, que accese a varios servidores al mismo tiempo y que la frecuencia de acceso a un mismo sitio se mayor a 30 segundos

8. ***Datos de Entrada.*** Se establecen datos con sitios que no permiten el acceso de Crawlers. ***Resultado Esperado.*** Se espera que el buscador deseche los sitios que no permiten acceso, pero continúe con su ejecución hasta guardar los datos extraídos de los demás dominios que si lo permiten.

Resultado Obtenido. En la mayoría de los casos resultó lo esperado; solo en los siguientes casos se produjeron errores:

- 1: Error al descargar archivos mayores a 5mb
- 7: El Fetcher no respetó la restricción de esperar 30 segundos mínimos de intervalo entre extracción en un mismo sitio.

Acciones. Se tomaron las siguientes acciones:

- Se determinó un nuevo requisito: "**RF-21 -** El sistema debe extraer archivos de cualquier tamaño". Luego se quitó la restricción de los 5mb en el algoritmo.
- Se modificó el algoritmo para establecer la restricción de esperar 30 segundos como mínimo en los intervalos de acceso a los dominios.

Así, con estos cambios se cumplió con los requisitos establecidos para este proceso.

Pruebas del Caso de Uso: Analizar Contenido

Descripción. Se examinará el proceso del componente Parser, para verificar su funcionamiento y si extrae los datos que se necesitan de los documentos.

Requisitos a Comprobar. En esta prueba se verificarán los requisitos: **RF-15** y **RF-19**.

Pre-condiciones. Deben existir datos en la Base de Datos con el contenido extraído de la WEB en distintos formatos (DOC, XLS, PDF, HTML, etc.)

Procedimiento. Se establearan datos manipulados por el usuario probador en la Base de Datos o bien ejecutar los procesos anteriores hasta el Fetcher. Este proceso no contempla validaciones, lo importante será revisar los enlaces nuevos extraídos y la metadata. Por lo que la prueba se orienta en el resultado de este algoritmo.

Casos de Prueba. Para este caso se estableció solo un caso de prueba:

1. ***Datos de Entrada.*** Documentos guardados en la Base de Datos y que hayan pasado por el Fetcher. ***Resultado Esperado.*** Se espera que guarde en la base de datos los documentos actualizados con la metadata y enlaces de salida y entrada.

Resultado Obtenido. Como resultado, se obtuvo lo esperado, por lo que se cumple con los requisitos.

Acciones. Ninguna.

Pruebas del Caso de Uso: Actualizar BD

Descripción. Se ejecutará el componente Actualizador, para comprobar que este verifique los enlaces nuevos extraídos por el Parser y sean ingresados a la Base de Datos.

Requisitos a Comprobar. Para esta prueba se verificarán los requisitos: **RF-04**, **RF-05**, **RF-15** y **RF-07**.

Pre-condiciones. Deben existir datos en el almacén con datos extraídos y el Parser ejecutado, con los campos inlinks y outlinks completados.

Procedimiento. Se almacenarán los datos resultados del proceso Parser y se debe ejecutar solamente el componente Actualizador a través del entorno de desarrollo por una prueba de unidad

Casos de Prueba. Se establecerá un solo caso de prueba, donde se verificarán varios atributos de los objetos como se describe:

1. ***Datos de Entrada.*** Datos almacenados en la Base de Datos, que hayan pasado por el Parser. ***Resultado Esperado***. Se espera que el proceso inserte en la Base de Datos los documentos con los nuevos enlaces encontrados. En este punto se debe verificar que cumpla con la restricción de no permitir enlaces de otros dominios y que al final el algoritmo actualice el score tanto de las páginas ya extraídas como las nuevas y el atributo del intervalo de búsqueda.

Resultado Obtenido. Las pruebas arrojaron el resultado deseado, por lo que se cumplieron los requisitos.

Acciones. Ninguna.

Pruebas del Caso de Uso: Indexar Contenido

Descripción. En esta prueba se comprobará la ejecución del componente Indexador, delegando este proceso al framework ElasticSearch.

Requisitos a Comprobar. Dentro de esta prueba se verificarán los requisitos: **RF-05** y **RF-03**.

Pre-condiciones. Debe existir documentos en la Base de Datos con por lo menos el contenido extraído de la WEB en varios formatos.

Procedimiento. Se tomarán los datos resultantes de los demás procesos, con contenidos de todo tipo (HTML, PDF, WORD, etc.). Como en el proceso no existen validaciones; la prueba se enfocará en el resultado de la indexación; el mismo se ejecutará como una unidad de prueba de solo ese componente. El resultado se verificará por medio de un cliente visual que puede observar la indexación de ElasticSearch.

Casos de Prueba. Se estableció un caso de prueba:

1. ***Datos de Entrada.*** Datos almacenados en la base de datos. ***Procedimiento.*** Se correrá el proceso de indexación para que ElasticSearch indexe la Base de Datos. ***Resultado Esperado.*** Se espera un almacén de datos indexados.

Resultado Obtenido. Se obtuvieron los resultados deseados; ya que a través del cliente visual se observó la indexación realizada por ElasticSearch e incluso se realizó una consulta en dicho cliente.

Pruebas del Caso de Uso: Consultar Documentos

Descripción. Se evaluará la interfaz de consulta y el resultado del funcionamiento del componente Buscador.

Requisitos a Comprobar. Dentro de esta prueba se verificarán los requisitos: **RF-15**, **RF-16** y **RF-18**.

Pre-condiciones. Deben existir datos indexados y el servidor de ElasticSearch activo con el índice inverso.

Procedimiento. El usuario debe entrar a la interfaz, y en la caja de texto ingresar los datos dispuesto en el caso de prueba y dar click en el botón buscar.

Casos de Prueba. Se determinaron 2 casos de prueba.

1. ***Datos de Entrada.*** Se dejará en blanco la caja de texto de búsqueda. ***Resultado Esperado.*** Se espera que no de resultado ni error alguno.
2. ***Datos de Entrada.*** Se insertará la palabra “Software” en la caja de texto. ***Resultado Esperado.*** Se espera una lista de resultado en la interfaz, además se debe verificar que los sitios resultantes estén dentro de los dominios de las semillas.

Resultado Obtenido. La ejecución de las pruebas arrojaron los resultados esperados.

Acciones. Ninguna.

Pruebas de Integración

Descripción. Este tipo de pruebas se realizan para observar la respuesta de los módulos interconectados a fin de detectar fallos resultantes de la interacción entre los componentes. Por tal motivo, se probarán todos los componentes interrelacionados y orquestados a través de los dos Buses de Integración.

Requisitos a Comprobar. Todos

Pre-condiciones. Todos los componentes deben estar activos y ejecutando los dos Buses.

Procedimiento. La principal dificultad de las pruebas de integración es la localización de los fallos. Por ello, en favor de la detección de los errores se utilizarán técnicas incrementales, es decir; se usará el depurador y se colocarán puntos de corte al final de la ejecución de cada componente para verificar su resultado y compararlos con los de las pruebas anteriores.

El administrador hará uso de la interfaz administrativa, ingresando los datos dispuestos en los casos de prueba; para comenzar a hilvanar uno a uno los procesos y por último se debe usar la interfaz de consulta para comprobar los resultados.

Es aquí, donde cobra importancia las pruebas por casos de uso; ya que una vez realizadas dichas pruebas, al poner a prueba el sistema integrado el resultado que se espera es de una probabilidad baja de fallos o faltas.

Casos de Prueba. Para este tipo de pruebas se establecerán como datos de entrada la configuración y las semillas; planificando tres casos de prueba:

1. ***Datos de Entrada.*** En la configuración se establecerá solo uno de profundidad. ***Resultado Esperado.*** El ciclo ARANEA debe realizarse una sola vez.
2. ***Datos de Entrada.*** La configuración establecerá cinco de profundidad. ***Resultado Esperado.*** El ciclo ARANEA debe realizarse cinco veces.
3. ***Datos de Entrada.*** No se establecerán semillas. ***Resultado Esperado.*** El sistema debe buscar los documentos ya almacenados en Base de Datos y tomarlos como semillas iniciales para actualizarlos.

Resultado Obtenido. El proceso de prueba recogió los resultados esperados; excepto en el segundo caso de prueba, donde al comienzo del segundo ciclo de ejecución, los nuevos URLs encontrados no se trasladaban de un proceso a otro, lo que resultaba solo unas pocas páginas extraídas; en vista de esto, a través de los puntos de corte se pudo determinar que los errores estaban en el generador.

Acciones. Se procedió a solventar el problema modificando el algoritmo del generador y ejecutar nuevamente los tres casos de prueba, con resultados satisfactorios.

Prueba de Evaluación y Comparación de Resultados

Con el objetivo de verificar la eficacia del sistema se hará una comparación de los resultados generados por ARANEA con dos buscadores tradicionales, como son: Google y Bing; para tal fin, se seleccionarán tres pruebas con distinto ámbitos de búsqueda, suministrando palabras claves para cada una. El resultado de los respectivos buscadores se evaluará por medio de las recomendaciones de calidad de contenidos realizado en la Fase de Diseño; lo que por motivo de tiempo solo se seleccionarán las cinco primeras entradas de los resultados por buscador. En la siguiente *Cuadro No. 31*, recoge el resumen de los resultados obtenidos de las pruebas.

Para tal fin, se configuró el sistema ARANEA, agregando semillas iniciales de sitios Web confiables, con propósitos relacionados al dominio de búsqueda; es importante recordar que no es alcance de la investigación el despliegue del sistema en un ambiente de producción; por lo que el procesos se realizó en un ambiente de prueba, donde en cada caso el sistema evaluó una profundidad de ejecución máxima de 3. Por tal motivo se esperan pocas cantidades de entradas resultantes en las consultas pero sin embargo, serán de sitios confiables.

Cuadro No. 31
Resultado de Pruebas Comparativas de Motores de Búsqueda

No	Descripción	Resultado ARANEA	Resultado Google	Resultado Bing
1	El buscador ARANEA se configuró tomando como base repositorios confiables relacionados con redes. La palabra clave que se usó para la búsqueda fue “Network Security”	Se encontraron 23 entradas. De las cuales las 5 primeras evaluadas, toda la información proviene de repositorios confiables.	El resultado fue de 936 millones de entradas de las cuales las 5 primeras; 2 son de repositorios confiables, 2 de repositorios no confiables y 1 dominio comercial.	Se produjeron 267 millones de entrada; de las cuales 1 pertenecía a un repositorio no confiable y 4 a repositorios comerciales de venta de productos.
2	ARANEA se configuró con sitios confiables de temas de Base de Datos y Desarrollo de Software. La entrada a buscar era “Advanced Database”	El resultado fue de 8 entradas con información relevante y confiable sobre el tema.	Se encontraron 681 millones de resultados, con 4 de ellos de sitios comerciales y 1 de repositorio no confiable.	El resultado fue de 106 millones de entradas de los cuales los 5 primeros eran de sitios Web comerciales.
3	Se pretendió buscar información sobre historia. Por tanto, se configuró ARANEA con sitios de museos, historia y documentales. La palabra clave a buscar fue “transaltecas”	Resultaron 5 entradas, con información confiable.	Solo 2 entradas fueron resultantes de las cuales 1 era de contenido pornográfico y la segunda de contenido confiable.	No arrojó resultado.

Fuente. El Investigador.

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

En esta tesis cuyo objetivo principal fue la construcción de un motor de búsqueda dirigido, basado en Crawlers, haciendo uso de una filosofía SOA y un bus de mensajes como middleware de integración, con el fin de extraer información confiable en La Web; se pretendió solventar el problema que presenta ubicar contenido de calidad en la Internet.

Por tal motivo, se establecieron cuatro fases para alcanzar los objetivos propuestos; donde a través de una investigación bibliográfica, se estudió el fenómeno de la Web Profunda o Invisible, la cual contiene según los estudios realizados, una gran información valiosa que no es indexada por los buscadores populares; así mismo se investigó sobre los métodos existentes de los motores de búsqueda, tecnología asociadas a SOA como WebServices y patrones de integración, almacenes de datos y un análisis sobre los atributos de calidad que pueden medir la confiabilidad de un repositorio. Esta investigación fue importante ya que sentó las bases del desarrollo del motor de búsqueda por medio de una metodología XP, empleando buenas prácticas de desarrollo.

En este sentido, el desarrollo del presente trabajo de investigación, permite establecer las siguientes conclusiones:

- El uso de una metodología de desarrollo XP, permitió ejecutar el proceso de desarrollo de manera rápida y con la mínima cantidad de artefactos de diseño. En donde el modelo iterativo sirvió de estrategia efectiva para incorporar de manera incremental las funcionalidades al sistema, ayudando a responder a nuevos requerimientos y cambios.
- Los métodos de enfoque global de extracción de información de los motores de

búsqueda no permiten abarcar la totalidad de los recursos de la Web, además que sus métodos de ranking, penalizan comúnmente de manera negativa a sitios de contenido confiables; por ello los métodos dirigidos permiten direccionar el ámbito de búsqueda, lo que contribuyó al logro de una herramienta que permite la extracción de contenido de la Web Invisible, ayudando de manera importante al usuario en su proceso de investigación.

- Por medio del análisis de varios estudios realizados por organismos, sobre la evaluación de repositorios confiables a través de características y parámetros, permite la determinación de calidad de los repositorios, contribuyendo a la elección de los dominios que se puedan establecer como semillas.
- En la investigación se puso en práctica el análisis y diseño orientado a servicio propuesto por Earl, usando la metodología Ágil; lo que confirmó la premisa de que estas etapas son las más importantes en el ciclo de desarrollo de una SOA, además apoyaron en gran medida al descubrimiento de los servicios.
- La metodología de diseño de arquitectura orientada a componentes, propuesta por Garland (2003), proporcionó una guía adecuada para entender cuáles eran los artefactos UML útiles para el desarrollo del sistema, tales como Diagramas de Casos de Uso, Clase, Interacción y Actividad, que facilitaron la percepción arquitectural del sistema; además se logró palpar que dicha metodología es acoplable con XP. Por otra parte, la arquitectura resultante de ARANEA, permite un desarrollo mantenible, en un dominio de problema muy dinámico, cifrado por necesidades cambiantes.
- El estudio, proporciono un espacio para dar un aporte al Proyecto Nutch de la comunidad Apache, coadyuvando en el desacoplamiento de sus componentes para darle escalabilidad y eficiencia; donde dentro del bus se pueden incorporar y/o reemplazar componentes que respondan a los mensajes del mismo, sin que resulte traumático.
- En el estudio se permitió evaluar la integración de conceptos como SOA, ESB, NoSQL e Indexación, en el contexto de búsqueda de información Web; lo que garantizó la incrementabilidad y rapidez de respuesta del sistema.
- Los patrones de integración basados en mensajes, generaron una arquitectura

basada en servicios con componentes desacoplados, colaborando en el proceso de orquestación y garantizando alguno de los requisitos no funcionales.

- Las pruebas comparativas arrojaron resultados prometedores, cuya evaluación se centro en la medida de características objetivas de acuerdo al *Cuadro No. 30*. A pesar de que solo se evaluaron los 5 primeros resultados de cada prueba; los enlaces resultantes de ARANEA fueron de repositorios confiables, en contra-parte de los obtenidos por Google y Bing (*Ver Cuadro No. 31*); entre las pruebas, es importante resaltar aquella cuya palabra clave fue "transaltecas", donde Google solo obtuvo una entrada a tomar en cuenta y Bing ni siquiera arrojó resultado; así se puede deducir de acuerdo a los resultados, que el modelo propuesto proporciona una probabilidad significativa de encontrar páginas que son verdaderamente de calidad, en comparación con los métodos de propósito general que arrojan una cantidad elevada de páginas en varios contextos. Por último, es importante destacar que las conclusiones de estas pruebas no se pueden extrapolar al comportamiento de estos motores de búsqueda ante cualquier tema; sería necesario realizar otros estudios exhaustivos utilizando la misma metodología pero con diferentes temas de búsqueda para averiguar si el comportamiento de los motores de búsqueda ofrece los mismos resultados que en este trabajo.

Recomendaciones

- Motivado a la naturaleza cambiante del dominio del problema planteado, y que los esfuerzos para acceder a la información contenida en la Web Oculta son valiosos, estos deben de ser continuados.
- A pesar de que el sistema ARANEA puede ser desplegado en un solo servidor, se aconseja seguir el modelo planteado en la fase de diseño, para dar más escalabilidad y eficiencia.
- Se sugiere realizar un estudio más detallado, acerca de los atributos que determinan la confiabilidad del repositorio; orientado al desarrollo de robots

inteligentes que permitan deducir de manera automática las semillas de configuración e incorporarlas como fuente de datos.

- Se aconseja el uso de modelos ontológicos especializados en ciertas áreas de conocimiento, para lograr una mejor relevancia en los resultados y por ende mayor exactitud en la búsqueda de información; agregando funcionalidades de la Web 3.0, por medio de inferencia de dominio.
- Se recomienda realizar pruebas de indexación con otras tecnologías, aprovechando los beneficios que otorga SOA, en cuanto al desacoplamiento de los componentes; permitiendo realizar cambios de una manera sencilla.

REFERENCIAS BIBLIOGRÁFICAS

Abiteboul, S., Preda, M. y Cobena, G. (2003). Adaptive on-line page importance computation [Documento en Línea]. Duodécima Conferencia de la World Wide Web, Budapest, Hungría. Disponible: http://www2003.org/cdrom/papers/refereed/p007/p7-abiteboul.html [Consulta: 2011, Marzo 4].

Acosta, L. (2006, Enero 17). Mensaje de vicerrector TIC respecto a los formatos públicos [Mensaje en línea]. Disponible: http://osl.ull.es/comunicado-formatos-abiertos [Consultada: 2011, Febrero 23].

Afp (2011, Enero). Ya hay 2.000 millones de internautas. El Mundo [Periódico en línea]. Disponible: http://www.elmundo.es/elmundo/2011/01/26/navegante/1296045597.html [Consulta: 2011, Abril 02].

Altuve, S. y Rivas, A. (1998). Metodología de la Investigación. Módulo Instruccional III. Caracas: Universidad Experimental Simón Rodríguez.

Álvarez, M. (2007). Arquitectura para crawling dirigido de información contenida en la web oculta. Tesis doctoral. Universidad Da Coruña, La Coruña, España.

Álvarez, P. (2008). Modelo de optimización evolutiva bicriterio para el entrenamiento de un spider enfocado. Trabajo de grado de maestría. Universidad Autónoma de Sinaloa, Sinaloa, México.

Arquitecturas SOA: BEA Systems (2004). Computing España. 14 (10). 17-23.

Barbarian et al (2007). Implementing Web Search With Nutch [Documento en Línea]. Disponible: http://mllab.csa.iisc.ernet.in/html/downloads/implementation/implementation.html#sec:nutch [Consulta: 2011, Octubre 22].

Bergman, M. (2000). The Deep Web: Surfacing Hidden Value. San Diego, USA: BrightPlanet.

Bergman, M. (2001). *The Deep Web: Surfacing Hidden Value* [Documento en Línea]. University of Michigan Library, Michigan. Disponible: http://quod.lib.umich.edu/cgi/t/text/text-idx?c=jep;view=text;rgn=main;idno=3336451.0007.104 [Consulta: 2011, Marzo 5].

Berkeley University (2011). *Evaluating Web Pages: Techniques to Apply & Questions to Ask* [Documento en línea]. Disponible: http://www.lib.berkeley.edu/TeachingLib/Guides/Internet/Evaluate.html [Consulta: 2012, Febrero 22].

Borso, M., Casciato, V., Rodríguez, H., Papaleo, M., Suárez, C., Suárez, J. (2006, Julio). Service Oriented Architecture (SOA): pautas y recomendaciones (Versión 0.9). Montevideo, Uruguay: Banco de Prevención Social, Comité Técnico de Arquitectura de Sistemas de Información.

Boswell, W. (2010). The Invisible Web: What It Is, How You Can Find It.

[Documento en Línea]. Disponible: http://websearch.about.com/od/invisibleweb/a/ invisible_web.htm [Consulta: 2011, Septiembre 20].

BrightPlanet (2011). The Deep Web FAQs [Documento en línea]. Autor. Disponible: http://brightplanet.com/the-deep-web/deep-web-faqs/ [Consulta: 2011, Septiembre 22].

Calvillo, J.A. (2010). Desarrollo de un índice espacial para la extensión JASPA sobre H2, Barcelona, España: Universidad Oberta de Catalunya.

Caponi, M., Rodríguez, P. y Zamudio, P. (2008). Mensajería en sistemas de información. Trabajo de grado de licenciatura. Universidad de la República, Facultad de Ingeniería, Montevideo, Uruguay.

Castillo, C. (2004). *Effective web crawling*. Tesis de Doctorado. Universidad de Chile, Santiago de Chile, Chile.

Castillo, C., Marin, M., Rodríguez, A., y Baesa-Yates R. (2004, Octubre 12-15). Scheduling Algorithms for Web Crawling: 10th Brazilian Symposium on Multimedia and the Web & 2nd Latín American Web Congress (pp. 10-17). Sao Paulo, Brasil; IEEE Computer Society.

Cattell, R. (2001, Enero). Scalable SQL and NoSQL data stores.

Cázeres, L., Christen, M., Jaramillo, E., Villaseñor, L. y Zamudio, L. E. (1980). Técnicas actuales de investigación documental. México: Universidad Autónoma Metropolitana.

Cejas, J. (2008, Junio). Un caso de éxito de implementación de SOA, dentro de un ambiente complejo y heterogéneo. Ponencia presentada en el foro latinoamericano de sistemas, tecnología y comunicaciones, Caracas.

Center for Reasearch Libraries (CRL). (sf). [Página Web en Línea]. Disponible: www.crl.edu/ [Consulta: 2011, Marzo 20].

Ciurana, E. (2007, Enero 1). Mule: A Case Study [Articulo en Línea]. The Server Side. Disponible: http://www.theserverside.com/news/1365047/Mule-A-Case-Study [Consultada: 2011, Octubre 2].

Chen H., Chung Y. Ramsey M. y Yang C. (1998). A Smart Itsy Bitsy Spider for the Web. Journal of the American Society of Information Science. 49(7).604-618.

Chaudhri, T. (2012, Febrero 17). Dublin Core [Articulo en Línea]. Thechnical Fundations. Disponible: http://technicalfoundations.ukoln.info/subject/dublin-core [Consultada: 2012, Marzo 5].

Codina, L. (2008). Estructura y funciones de un motor de búsqueda. En C. Rovira (Dir.). Master online en buscadores: selección de unidades didacticas 2007/2008 (pp F092-1 – F092-11). Barcelona, España: Universidad Pompeu Fabra.

Consultative Committee for Space Data Systems (CCSDS). (2002, Enero). Reference

model for an open archival information system (OAIS) (650.0-B-1). Washigton, USA: Autor.

Cutting, D. (2004, Mayo 22). Nutch: Open Source Web Search. Ponencia presentada en la 13th World Wide Web Conference, New York.

Earl, T. (2005). Service-Oriented Architecture: Concepts, Technology, and Design. Boston, MA: Prentice Hall PTR.

Fielding, R. (2000). Architectural Styles and the Design of Network-based Software Architectures. Tesis Doctoral, Universidad de California, Irvine, Estados Unidos.

García, M. (2002). Estructura de datos: métodos avanzados de representación de conjuntos [Documento en línea]. Universidad de Murcia, Departamento de Informática y Sistemas, Murcia, España. Disponible: http://www.dis.um.es/ ~ginesgm/temas/tema3-1/index.htm [Consultada: 2011, Abril 02].

Garlan, D., Shaw, M. (1994). An Introduction to Software Architecture. Vol 1. Pittsburgh, Estados Unidos: Carnegie Mellon University, School of Computer Science.

Garland, J., Anthony, R. (2003). Large-Scale Software Architecture. West Sussex, Inglaterra: Jhon Wiley and Son.

Genkin, M. (2007, Marzo 20). Best practices for service interface design in SOA, Part 1: Exploring the development, interfaces, and operation semantics of services [Articulo en Línea]. IBM DeveloperWorks. Disponible: http://www.ibm.com/developerworks/architecture/library/ar-servdsgn1/ [Consultada: 2011, Octubre 15].

Glaser, M. (2010). Sobre la tecnología detrás de Google Instant [Documento en línea]. Disponible: http://www.maxglaser.net/sobre-la-tecnologia-detras-de-google-instant/ [Consulta: 2011, Marzo 12].

González, J. y Sadier, P. (2006). Curiosear el compromiso de crear datos estructurados sobre los datos. Consejo Latinoamericano de Ciencias Sociales (CLACSO), Buenos Aires, Argentina.

Greenwood, A. y Steyn, D. (2011). *Evaluating Internet Sources* [Documento en línea]. UBC Library, University of British Columbia. Disponible: http://help.library.ubc.ca/researching/evaluating-internet-sources/#fragment-1-4 [Consulta: 2011, Septiembre 22].

Gruchawka, S. (2005, Noviembre 16). *Using the Deep Web: A How-To Guide for IT Professionals* [Documento en línea]. Disponible: http://techdeepweb.com/ [Consulta: 2011, Septiembre 25].

Guardo, M. E. y Pentón, R. (2008). Fundamentos de la investigación científica y sus particularidades en la cultura física.

Guarino, N. (1998, Junio 7-8). Formal Ontology and Information Systems. Formal Ontology in Information Systems. Ponencia realizada en The 1st International

Conference, Trento, Italia.

Hansen, M. (2007). *SOA Using Java Web Services*. New Jersey, Estados Unidos: Prentice Hall.

Hohpe, G. (2010). [Página Web en Línea]. Disponible: http://www.eaipatterns.com/ [Consulta: 2011, Noviembre 8]

Hohpe, G. y Woolf, B. (2003). *Enterprice Integrations Patterns*. Boston, Estados Unidos: Addison-Wesley.

IEEE (2003). [Página Web en Línea]. Disponible: http://suo.ieee.org/ [Consultada: 2011, Noviembre 19]

Interpristor. (2010). *Web Crawler.* Estados Unidos: Autor.

Isopixel (2010). La historia de los motores de búsqueda - infografía [Documento en Línea]. Autor. Disponible: http://isopixel.net/archivo/2010/10/la-historia-de-los-motores-de-busqueda-infografia/ [Consulta: 2011, Marzo 28].

Jantz, R. Y Giarlo, M. (2005). Digital Preservation: architecture and technology for trusted digital repositories. D-Lib Magazine [Revista en línea]. 11(6). Disponible: http://www.dlib.org/dlib/june05/jantz/06jantz.html [Consulta: 2011, Marzo 16].

Kaisler, L. (2003). Traditional Sources [Documento en línea]. Disponible: http://21cif.com/tutorials/micro/mm/traditional/ [Consulta: 2011, Septiembre 15].

King, J. (2008). Seach engine content analysis. Tesis doctoral. Queensland University of Technology, Brisbane, Australia.

Klaudinyi, J. (2007). Evaluating Internet Sources [Documento en línea]. Disponible: http://www.wou.edu/provost/library/clip/tutorials/eval_internet.htm [Consulta: 2011, Septiembre 15].

Klas, J. (2010). Recuperación de Información sobre Modelos de Dominio. Trabajo de Grado. Universidad de Buenos Aires, Facultad de Ingeniería, Buenos Aires, Argentina.

Lackshman, A. y Malik, P. (2009, Octubre). Cassandra: structured storagesystem over P2P network. Ponencia presentada en LADIS 2009: the 3rd ACM SIGOPS international workshop on large scale distributed systems and middleware, Montana, USA.

Lamarca, M. J. (2010). Hipertexto: El nuevo concepto de documento en la cultura de la imagen [Tesis en línea]. Tesis Doctoral, Universidad Complutense de Madrid. Disponible: http://www.hipertexto.info/ [Consultada 2011, Marzo 30].

Lyman, P y Varian H. (2003). How Much Information 2003? [Documento en Línea]. Berkeley University, California. Disponible: http://www2.sims.berkeley.edu/ research/projects/how-much-info-2003/ [Consulta: 2011, Marzo 10].

Mahmoud, Q. (2005, Abril). *Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI)* [Articulo en Línea]. Oracle.

Disponible: http://www.oracle.com/technetwork/articles/javase/soa-142870.html [Consultada: Octubre 15]

Manpower. (2009). El Impacto de las Redes Sociales de Internet en el Mundo del Trabajo: México. D.F., México: Autor.

McBride, I. (2010). Web Crawling and Search with Nutch and Solr. Ponencia presentada en DrupalCon 2010, Chicago, Estados Unidos.

Meneses, E. (2003). Estructuras de datos 2: Hashing. CenfoTEC, Costa Rica.

Metz, C. (2010, Septiembre). Google search index splits with MapReduce: welds BigTable to file system "Colossus". The Register [Revista en línea]. Disponible: http://www.theregister.co.uk/2010/09/09/google_caffeine_explained/ [Consulta: 2011, Marzo 12].

Microsoft (2005). Descripción de la disponibilidad, la confiabilidad y la escalabilidad [Documento en línea]. Disponible: http://technet.microsoft.com/es-es/library/aa996704%28EXCHG.65%29.aspx [Consulta: 2011, Febrero 23].

Microsoft (2009). Patterns & Practices [Libro en Línea]. Autor. Disponible: http://msdn.microsoft.com/en-us/library/ff921345.aspx [Consulta: 2011, Noviembre 8].

Montero, J. (2009). Arañas web (Crawlers) [Documento en Línea]. Disponible: http://recuperacionorganizacioninformacionacces.net78.net/aranas_web_%28 crawlers%29/aranas_web_%28crawlers%29_definicion.html [Consulta: 2011, Febrero 02].

Montilva, J. Y Barrios, J. (2007). Desarrollo de Software Empresarial. Mérida, Venezuela: Universidad de los Andes.

Najork M. y Wiener J. (2001). Breadth-first crawling yields high-quality pages [Documento en Línea]. Décima Conferencia de la World Wide Web, Hong Kong, China. Disponible: http://www10.org/cdrom/papers/pdf/p208.pdf [Consulta: 2011, Marzo 4].

Navarro, G. (2007). *Implementación de Máquinas de Búsqueda I: Índices y Compresión.* Santiago de Chile, Chile: Universidad de Chile, Centro de Investigación de la Web.

Online Computer Librarie Center (OCLC). (2011). [Página Web en Línea]. Disponible: http://www.oclc.org/ [Consulta: 2011, Marzo 20].

Pacheco, D. (2009, Marzo). O papel do ESB em uma solução SOA [Documento en línea]. Disponible: http://diego-pacheco.blogspot.com/2009/03/o-papel-do-esb-em-uma-solucao-soa.html [Consultada: 2011, Febrero 08].

Pandia. (2007, Enero 15). The Search Engine Scene in 2015 [Documento en línea]. Disponible: http://www.pandia.com/sew/353-search-2015.html [Consultada: 2011, Septiembre 20].

Papazoglou, M. y Den Heuvel W. Van (2005). Services oriented architecture: approaches, technologies and research issues. The VLDB Journal, 16(3), 389-415.

Papazoglou, M. y Den Heuvel W. Van (2006). Service-oriented design and development methodology. Int. J. of Web Engineering and Technology (IJWET), 2(4), 412-442.

Parra, J., Sánchez, S., Sanjuán, O. y Joyanes, L. (2005). RAWS Architecture: Reflective and Adaptable Web Service Model. Madrid, España: Universidad Pontificia de Salamanca/Universidad de Alcalá.

Pedraza, R. (2009). Taxonomías: una nueva estrategia para la organización de la información en sitios web con contenido cultural [Documento en línea]. Universitat Pompeu Fabra, Barcelona, España. Disponible: http://www. ricardmonistrol.cat/Rafael.Pedraza.ppt [Consulta: 2011, Abril 2].

Pedraza, R. Y Codina L. (2009). Motores de búsqueda para usos académicos [Documento en línea]. Universidad Politécnica de Valencia, Valencia, España. Disponible: www.lluiscodina.com/BuscadoresAcademicos_2009.ppt [Consulta: 2011, Febrero 22].

Ramírez, A. (2009). Bases de datos especializadas e internet invisible [Documento en Línea]. Universidad ICESI. Disponible: http://www.icesi.edu.co/blogs/egatic/tag/ buscadores-especializados/ [Consulta: 2011, Febrero 24].

Requena, C. (2010). Qué es NoSQL? [Documento en Línea]. Disponible: http://www.nosql.es/blog/nosql/queW3C (2004) - Disco-es-nosql.html [Consulta: 2011, Febrero 02].

Rose India (2006, Abril 3). Multithreading in Java [Documento en Línea]. Disponible: http://www.roseindia.net/java/thread/Java-Multithreading.shtml [Consulta: 2011, Octubre 4].

Rodhenizer, D., y Trudel, A. (2007). *How Big is The World Wide Web?*. Wolfville, Canada: Jodrey School of Computer Science, Acadia University.

Ruiz, J. M. (2009). BigTable. Linux Magazine, 39, 47-50.

Schmidt, D., Stal, M., Rohnert, H. y Buschmann, F. (2000). Pattern-Oriented Software Architecture, Patterns for Concurrent and Networked Objects. Volumen 2. West Sussex, Inglaterra: Jhon Wiley and Son.

Seeger, M. (2010). Building blocks of a scalable web crawler. Trabajo de grado de maestría. Stuttgart Media University, Stuttgart, Inglaterra.

Shestakov, D. (2008, Mayo). Search Interfaces on the Web: Querying and Characterizing [Número Especial]. TUCS Dissertations, 104.

Sholler, D. (2010, Agosto 4). Hype Cycle for Application Architecture, 2010 [Articulo en Línea]. Gartner. Disponible: http://www.gartner.com/DisplayDocument?id=1416814 [Consultada: 2011, Octubre 22].

Short, J., Bohn, H. Y Baru Ch. (2011, Enero). How Much Information? 2010: Report on Enterprice Server Information. San Diego, Estados Unidos: Global Information Industry Center UC San Diego.

Solar, M. y Figueroa, L. (2010). Estructura de datos: Arboles B, Arboles B*, Arboles B+ [Documento en línea]. Universidad Técnica Federico Santa María, Viña del Mar, Chile. Disponible: www.ramos.utfsm.cl/doc/860/sc/ED-Cap5_Arboles12010_.pdf [Consultada: 2011, Abril 02].

Soler, J. (2008). La Preservación de los Documentos Electrónicos. Barcelona, España: UOC.

Suárez, I. y Plasencia A. (2010, Abril). Consideraciones para el desarrollo de un buscador especializado en internet con software libre. Ponencia presentada en el V Seminario Internacional sobre Estudios Cuantitativos y Cualitativos de la Ciencia y la Tecnología, La Habana, Cuba.

Sullivan, D. (2005). Search engines sizes [Documento en Línea]. SearchEngineWatch. Disponible: http://searchenginewatch.com/2156481#SWI [Consulta: 2011, Marzo 18].

Tyagi, S. (2006, Agosto). RESTful Web Services [Articulo en Línea]. Oracle. Disponible:http://www.oracle.com/technetwork/articles/javase/index-137171.html [Consulta: 2011, Noviembre 14].

Universidad Centroccidental Lisandro Alvarado (UCLA) (2002). Manual para la Elaboración del Trabajo Conducente a Grado Académico de Especialización, Maestría y Doctorado. Sesión Ordinaria No 1353.

Universidad Pedagógica Experimental Libertador (UPEL), Vicerrectorado de Investigación y Postgrado (2002). Manual de Trabajo de Grado y Tesis Doctorales. Caracas: Fondo Editorial de la Universidad Pedagógica Experimental Libertador.

University of South Florida (2010). *Evaluating Sources.* Florida, Estados Unidos: Autor.

Urrutia, J. (2006, Octubre). Bus de Servicios Empresariales (ESB), SOA, BPM – Relacionando todas estas siglas [Documento en línea]. Disponible: http://www.misbytes.com/wp/2006/10/08/buses-de-servicios-empresariales-esb-soa-bpm-relacionando-todas-estas-siglas/ [Consultada 2011, Febrero 09].

Valera, R. (2010). Arquitectura de Software Para Automatizar los Registros Académicos en la Universidad Centroccidental “Lisandro Alvarado”. Trabajo de ascenso, Universidad Centroccidental Lisandro Alvarado, Barquisimeto, Venezuela.

Vázquez, J. (2010, Mayo). ¿Por qué un Enterprise Service Bus (ESB)? [Documento en línea]. Disponible: http://blogs.tecsisa.com/articulos-tecnicos/por-que-un-enterprise-service-bus/ [Consultada: 2011, Marzo 10].

Venzke, M. (2003). Automatic Validation of Web Services. Tesis doctoral, Technische

Universität Hamburg, Hamburg, Alemania.

VirginiaTech University (2011). *Evaluating Internet Information* [Documento en línea]. Disponible: http://www.lib.vt.edu/instruct/evaluate/ [Consulta: 2012, Febrero 23].

W3C (2004, Febrero 10). OWL Web Ontology Language [Documento en Línea]. Autor. Disponible: http://www.w3.org/TR/2004/REC-owl-features-20040210/ [Consultada: 2011, Octubre 2].

W3C (2004, Febrero 11). Web Service Architecture. Autor.

W3C (2008). Guía Breve de Tecnologías XML [Documento en Línea]. Autor. Disponible: http://www.w3c.es/divulgacion/guiasbreves/tecnologiasxml [Consultada: 2011, Octubre 23].

Wall, A. (2003). History of search engines:from 1945 to Google today [Documento en Linea]. SearchEngineHistory. Disponible: http://www.searchenginehistory.com/ [Consulta: 2011, Marzo 16].

Zanarini, D. (2010). Tablas hash: La estructura de datos de tus sueños.

ANEXOS

A. Currículum Vitae del Autor

Edward José Valera Giménez, Portador de la C.I: N° V-14.810.884, nace en Quibor el 20 de Agosto de 1981. Realiza estudios de Primaria en el Colegio Presbiteriano “Divino Salvador”. Obtiene el título de bachiller en el Colegio “Nueva Segovia” de Barquisimeto e Ingresa a la Universidad Centroccidental “Lisandro Alvarado” de Barquisimeto en el año 1997, para iniciar estudios de pregrado en la carrera de Ingeniería en Informática, egresando en el año 2003 y obteniendo el 8vo lugar de la promoción. En ese mismo año comienza su actividad profesional como desarrollador de aplicaciones en la empresa Marna, S.A, rol que cumple hasta el año 2006; cuando ingresa como personal administrativo de la Universidad Centroccidental “Lisandro Alvarado”, cumpliendo el cargo de Jefe de Desarrollo del proyecto CumLaude, adscrito a Secretaria General. Paralelamente en el año 2009 inicia la carrera docente, desempeñándose en el cargo de Docente Tiempo Convencional adscrito al Departamento de Sistemas del Decanato de Ciencias y Tecnología de la UCLA, específicamente en el área de base de datos. Luego, en el 2010 ingresa como Personal Docente Ordinario tiempo completo, a través de concurso de oposición, cargo que desempeña en la actualidad. Durante su carrera de pregrado obtuvo varias distinciones de cuadro de honor por rendimiento académico; y en el 2004 y 2005 fue participante en dos evento de IBM donde ganó dos campeonatos mundiales de desarrollo, destacando por encima de participantes de Rusia, India, Reino Unido, Estados Unidos, entre otros; haciendo uso del IDE WebPhere, certificación que también obtuvo en el 2005.