UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 3

Grupo de Investigación de Web Semántica

“Indización de grafos RDF desde un SPARQL Endpoint”

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

Autor(es):
Juan Carlos Moreira De Lara
Alejandro Jesús Mariño Molerio

Tutor(es):
MSc. Yusniel Hidalgo Delgado
Ing. Ernesto Ortiz Muñoz

La Habana, junio de 2016
“Año 58 de la Revolución”

DECLARACIÓN DE AUTORÍA

Declaro ser el autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Juan Carlos Moreira de Lara

Autor

_____

Alejandro Jesús Mariño Molerio

Autor

_____

MSc. Yusniel Hidalgo Delgado

Tutor

_____

Ing. Ernesto Ortiz Muñoz

Tutor

DATOS DE CONTACTO

Síntesis del Tutor

El máster en ciencias Yusniel Hidalgo Delgado se graduó con Título de Oro en la Universidad de Ciencias Informáticas en el año 2010. En su primer año de adiestramiento desempeñó diversos roles dentro del proyecto de desarrollo del ERP Cubano. Actualmente se desempeña como profesor asistente del departamento docente de técnicas de programación de la Facultad 3. Es coordinador del grupo de investigación de Web Semántica de la UCI. Es miembro de la Asociación Cubana de Reconocimiento de Patrones, de la Sociedad Cubana de Matemática y Computación y de la *International Association for Pattern Recognition*.

DEDICATORIA

A la vida, a Dios, a mis padres*, únicos e irrepetibles, para los que la palabra amor queda pequeña.*

A mi familia *en su totalidad, cercana o lejana, por confiar en mí, dejar labrar mi propio destino, sentirse orgullosa de mis éxitos, aconsejarme y apoyarme.*

A mis amigos*, los incondicionales, que son parte de mi familia y a los quiero y respeto. Gracias por su mano amiga.*

Juan Carlos Moreira de Lara

Para Lili y Cati*, que ya no están en el reino de este mundo.*

A mi mamá, mis abuelas, mis hermanos y a Kenia*.*

Alejandro Jesús Mariño Molerio

AGRADECIMIENTOS

***A la vida**, por permitirme tener a mi alrededor a personas como mi familia, seres maravillosos como los que jamás habré de conocer, por el privilegio e inmenso placer de ser hijo de padres como los que orgullosamente tengo y por realizar el sueño que hoy se materializa.*

***A mi padre**, a quien le debo en parte todo lo que soy, su educación, su carácter, sus enseñanzas, sus gestos, el modo de ver y afrontar la vida, el saber escoger a los amigos. Cada mañana pido conforme ser sólo, un poco parecido a él.*

***A mi madre**, por velar cada uno de mis pasos, por ser luchadora incansable, guerrera de todas las batallas, fiel consejera. Gracias por el amor infinito, el cuidado y la comprensión desmedida.*

***A mi tío Humbe**, no importan las distancias, por quererme, por dame su hombro cuando he necesitado donde recostarme, la palabra de aliento que se precisaba para poder levantar la cabeza y mirar hacia el futuro.*

***A mis amigos**, los de verdad, los de siempre y para siempre, por festejar juntos mis victorias y compartir también las penas. A Isra, Rosy y la familia que también es mía, Lisy, Migue, Fleitas, Badia y familia gracias a todos por enseñarme el verdadero valor de la amistad.*

***A Yusniel**, mi tutor, por guiarme en los primeros pasos del mundo académico que escogí, por ser ejemplo y referente, por su apoyo y constancia, su preocupación y sacrificio.*

Juan Carlos Moreira de Lara

*Con mucho amor, **a mi mamá**, que nunca dijo "no puedo".*

*Especialmente, **a mi novia Kenia**, la doctora de mi vida, y **su mamá Odalis**, quien tiene en mi un hijo.*

*Sinceramente, **a Alida y Line**, mi familia "adoptiva" desde que comencé la universidad, con su ayuda el camino fue más sencillo.*

*Maternalmente, **a mis abuelas**, que siempre me quieren cerca.*

*Desinteresadamente, **a mis amigos**, los "nuevos" y los de "antes", el Flako, Osbel, Paul, Luisma, Arian, Osmany, Daniel, Lazaro, etc.*

Sin olvidar a Thais, Yamila, Yesi y Yiniam.

*Semánticamente, **a Yusniel Hidalgo**, quien nos "embarcó" en esta aventura.*

*Finalmente, pero no menos importante **a mi papá y Dayami**, mi "madrastra" de cuento de hadas.*

Alejandro Jesús Mariño Molerio

RESUMEN

Las bibliotecas han tenido un valor preponderante en la sociedad. Esto se debe a que éstas atesoran el conocimiento científico de la humanidad. En el escenario actual, las bibliotecas se agrupan en tres grandes categorías: convencionales, digitales e híbridas. En el caso de las bibliotecas digitales, éstas coleccionan, almacenan y distribuyen la información en soporte digital. La aplicación de los principios para la publicación de contenido en la Web, ha permitido una mejora en el procesamiento y gestión de la información. En este sentido, los datos enlazados se han convertido en un área de investigación activa en los últimos años, específicamente ha cobrado importancia la publicación de metadatos bibliográficos, según alguna de las serializaciones del formato RDF. Sin embargo, el contenido web publicado siguiendo esta iniciativa, no puede ser consumido por los usuarios que desconocen las tecnologías de la Web Semántica (*RDF*, *SPARQL*, etcétera), debido a que se requiere la comprensión de la estructura, naturaleza y forma de consultar los datos. El grupo de investigación de web semántica de la Universidad de las Ciencias Informáticas desarrolla un proyecto de investigación que genera grafos RDF con metadatos bibliográficos que son almacenados para su posterior uso en un almacén de tripletas. Estos almacenes no se encuentran optimizados para la realización de consultas formuladas por los usuarios en tiempos cercanos al real, por lo que se hace necesario contar con estructuras de datos optimizadas (índices) para disminuir dicho tiempo y resolver los problemas de búsqueda y recuperación de datos en grandes bases de datos. En esta investigación se propone un componente de software que integra un servidor de indización con tripletas RDF existentes en un almacén de tripletas, así como un prototipo funcional para la búsqueda textual y facetada sobre los datos en el índice.

Palabras claves: biblioteca digital, grafo RDF, índice, metadato bibliográfico, web semántica.

ABSTRACT

Libraries have played a significant value in society. This is because they treasure the scientific knowledge of humanity. In the current scenario, libraries are grouped into three broad categories: conventional, digital and hybrid. In the case of digital libraries, they collect, store and distribute digital information. The application of the principles for publishing content on the Web, has allowed an improvement in processing and information management. In this sense, the linked data have become an area of active research in recent years has gained importance specifically publishing bibliographic metadata, according to one of the RDF serialization formats. However, the web content published following this initiative cannot be consumed by users who are unaware Semantic Web technologies (RDF, SPARQL, etc.), because the understanding of the structure, nature and form of consultation is required the data. The research group semantic web University of Computer Science develops a research project that generates RDF graphs with bibliographic metadata that is stored for later use in a store triples. These stores are not optimized for performing in near time the actual queries submitted by users, so it is necessary to have structures optimized data (indices) to decrease this time and solve the problems of search and retrieval in large databases. In this research, a software component that integrates an indexing server with existing RDF triples in a triplestore and a functional prototype for textual and faceted on the data in the search index is proposed.

Keywords: *bibliographic metadata, digital library, index, RDF graph, semantic web.*

ÍNDICE

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Las bibliotecas siempre han tenido un valor preponderante en la sociedad, en unas culturas más que en otras, lo cual se debe a que éstas atesoran el conocimiento científico de la humanidad. Sus funciones, según indica (Shanhong 2000) son reunir, procesar, difundir, almacenar y usar la información documental para dar servicio a la sociedad. Como principio básico, las bibliotecas representan un eslabón indispensable en la cadena del sistema científico, toda vez que son parte directa en el proceso de investigación científica y operan como puentes para convertir los resultados de la innovación del conocimiento en fuerzas realmente productivas.

Durante la segunda mitad del siglo XIX se produjo un sensible incremento de la producción de libros y la lectura se extendió entre la nueva clase burguesa del viejo continente. Con grandes anhelos de educación, la investigación gozó de un vigoroso aumento, lo que proporcionó un auge considerable de revistas científicas (López y Perelló 2005). Esta situación ocasionó un impacto directo en la proliferación de bibliotecas públicas, universitarias y nacionales.

Como apunta (Shanhong 2000) en la era de la economía del conocimiento, la gestión de dicho conocimiento se refiere a identificar, adquirir, desarrollar, resolver, usar y participar del conocimiento de una manera efectiva. Para (Vargas 2012) la gestión del conocimiento requiere una conexión de la información con la información, la información con las actividades y la información con el hombre para compartir el conocimiento. En este contexto se ha incrementado la atención de la sociedad por la información y el conocimiento, específicamente el conocimiento científico, debido a que ambos, se han convertido en la fuerza conductora para el desarrollo social. Lo peculiar del conocimiento es que a partir de él se genera un nuevo conocimiento, pero se requiere primeramente elegir documentos con calidad para investigar y generar un nuevo conocimiento de valor. Es importante que la información cubra con ciertos requisitos, y que los usuarios que la obtengan puedan a partir de ella generar conocimientos de calidad (Vargas 2012).

Desde finales del siglo XIX hasta la actualidad, la documentación y las tecnologías de la información han evolucionado, enriqueciéndose mutuamente en su desarrollo. La visión de la web actual es fruto en parte del trabajo de científicos como Vannevar Bush, considerado como el padre de los sistemas hipertexto, quien crea en 1940 un aparato llamado *Memex*, una especie de archivo privado o biblioteca, en el que se almacenan libros, registros y comunicaciones de forma mecanizada para poder consultarlos con gran rapidez y flexibilidad. Aun cuando el *Memex* no fue construido, la idea de utilizar el ordenador para almacenar conocimiento mediante asociaciones entre los distintos documentos, fue

el origen de las posteriores investigaciones sobre los sistemas de hipertexto e hipermedia, que son ya una realidad, según revela el estudio de (Artiga 1993).

El surgimiento de Internet y el desarrollo de las *Tecnologías de la Información y las Comunicaciones* (TIC, por sus siglas en español) revolucionaron la forma en que las bibliotecas llevaban a cabo su actividad fundamental; transformando su organización en profundidad como expone (López y Perelló 2005) y articulándose con nuevas perspectivas de servicio como es el caso de: la teledocumentación, la difusión selectiva, el acceso en línea a bases de datos documentales y científicas, los servicios de Internet y correo electrónico, así como el incremento de documentos en soportes no bibliotecarios con sus correspondientes servicios (fonotecas[1], fototecas[2], cartotecas[3], videotecas[4], entre otros). Según refieren (Singh y Sharma 2015) en el escenario actual las bibliotecas se agrupan en tres grandes categorías: convencionales, digitales e híbridas. En el caso particular de las bibliotecas digitales, estas coleccionan, almacenan y distribuyen la información en soporte digital.

Como tendencia, los científicos y académicos publican los resultados de sus investigaciones en revistas científicas. En muchos casos estos resultados son almacenados en los repositorios de las instituciones a las que pertenecen o en bases de datos relacionales, donde resulta complejo el acceso al conocimiento científico y tecnológico. Como señala (Vargas 2012) las bibliotecas en todas sus modalidades y tipos afrontan el reto de centrarse por investigar acerca del desarrollo del conocimiento así como la creación de las bases del conocimiento. Con el objetivo de hacer del conocimiento un recurso universal y accesible a todos, surge la *Open Access Initiative* (OAI, por sus siglas en inglés), en español: Iniciativa de Acceso Abierto. OIA desarrolla y promueve estándares de interoperabilidad que tienen como objetivo facilitar la difusión eficiente de contenidos.

Debido a la gran diversidad y cúmulo de las fuentes y recursos[5] en Internet, se hizo necesario establecer un mecanismo para etiquetar, catalogar, describir y clasificar el ingente volumen de recursos presentes en la *World Wide Web* (WWW, por sus siglas en inglés), con el fin de facilitar la posterior búsqueda y recuperación de la información. Este mecanismo lo constituyen los llamados metadatos.

La contribución de (García 2013) destaca que en la década del 60 los metadatos se empleaban únicamente como referentes al proceso automatizado y a la descripción de la información que documenta la información contenida en las bases de datos. Los cambios en el uso y producción de la

[1] Colección de documentos sonoros, como cintas, discos, discos compactos, etcétera.
[2] Organización encargada de adquirir, organizar, conservar y catalogar fotografías.
[3] Lugar donde se conservan ordenados y clasificados los mapas para su consulta o estudio.
[4] Edificio o dependencias donde se conservan grabaciones de video para su consulta.
[5] El término de la arquitectura web “recurso” se refiere al contenido de interés en la web que se publica mediante URIs HTTP.

información son quienes imponen relevancia para los metadatos en el entorno bibliotecario. Dublin Core, tal como expresa (Delgado 2015), se consolida como el esquema de metadatos más empleado en la actualidad, no solo en el contexto de las bibliotecas, sino en otros ámbitos de aplicación. Este se distingue por su simplicidad y su flexibilidad, atributos que lo sitúan como el esquema de metadatos que más ha sido reutilizado en la definición de ontologías de dominio en el contexto de las bibliotecas digitales.

Para la recolección de estos documentos, se creó el *Open Access Initiative Protocol for Metadata Harvesting* (OAI-PMH[6], por sus siglas en inglés), es español: Protocolo de la Iniciativa de Archivos Abiertos para la Recolección de Metadatos, el cual provee un mecanismo a bajo nivel para la interoperabilidad de los repositorios digitales (Delgado 2015). Existen algunos sistemas de bibliotecas digitales de código abierto que actúan como proveedores de datos y pueden exponer metadatos mediante OAI-PMH de forma predeterminada, en este caso encontramos herramientas como Dspace[7], EPrint[8] y Fedora[9].

En el dominio de los datos bibliográficos es posible construir bibliotecas digitales con datos enriquecidos semánticamente. Se intenta proporcionar a los usuarios el acceso a bibliotecas digitales que integren diversas fuentes de datos, proporcionando servicios de valor añadido. Las bibliotecas digitales albergan actas de congresos, que constituyen recopilaciones de las ponencias y comunicaciones de congresos, simposios, seminarios, memorias de eventos científicos, entre otros. Usualmente dichas actas son almacenadas en formato *Portable Document Format* (PDF, por sus siglas en inglés) que aparece como una alternativa para la creación de documentos. Estas bibliotecas pueden proporcionar acceso al texto completo de las contribuciones, en adición a los metadatos de cada una de ellas (Delgado 2015). La calidad de los metadatos bibliográficos es un elemento crucial que afecta significativamente la visibilidad, el descubrimiento y la reutilización de los recursos descritos en el contexto de los datos enlazados.

La aplicación de los principios a la publicación de contenido en la Web, indican (Rizo y García 2013) ha permitido una considerable mejora en el procesamiento y gestión de la información tanto para humanos como para las máquinas (agentes software). Los datos enlazados se han convertido en un área de investigación activa en los últimos años, específicamente ha cobrado importancia la publicación de metadatos bibliográficos, según alguna de las serializaciones del formato RDF (del inglés: *Resource Description Framework*), en español: Marco de Descripción de Recursos. Esto se debe a la necesidad

[6] http://www.openarchives.org/pmh/
[7] http://www.dspace.org/
[8] http://www.eprints.org/uk/index.php/eprints-software/
[9] http://fedorarepository.org/

de publicar y consumir datos estructurados en la Web, potenciando de esta manera el desarrollo de la Web Semántica.

Revelan (Bizer, Jentzsch y Cyganiak 2011) que para el año 2010 la nube de datos abiertos enlazados contaba con alrededor de 26,9 billones de tripletas RDF y con 436 millones de enlaces RDF entre distintas fuentes de datos. Sin embargo, como opinan (Dadzie y Rowe 2011), está vigente una barrera técnica en relación con el uso de datos enlazados para los usuarios que no están familiarizados con las tecnologías de la Web Semántica (como RDF, ontologías, vocabularios y metadatos), comúnmente llamados usuarios no técnicos (del inglés: *lay-user*) principalmente por el desconocimiento de la naturaleza y estructura de los datos, así como la necesidad de tener que realizar complejas consultas semánticas sobre grafos RDF.

Cuando un usuario realiza la solicitud de un recurso a través de una URI (ver la sección 1.3.1) se retorna información útil relacionada con el recurso (Berners-Lee 2006). Por tanto, cuando una URI es dereferenciada, se retorna una respuesta de acuerdo a la especificación de los parámetros en la solicitud, puede ser retornada en formato HTML (ver la sección 1.3.1), en cuyo caso es fácilmente interpretada por un navegador Web y mostrada de manera intuitiva al usuario. Sin embargo, lo común es que la respuesta sea retornada en algún tipo de serialización del formato RDF: *RDF/XML* (Bray et al. 2008), *N3* (Berners-Lee y Connolly 2008), *Turtle* (Prud'hommeaux et al. 2013), entonces el cómo interpretar y utilizar este formato está restringido solamente a los usuarios técnicos (del inglés: *techsavvy-user*), y en ciertos casos, a quienes tienen conocimiento de las tecnologías de la Web Semántica, según hacen notar (Dadzie y Rowe 2011).

Varios casos de éxito han sido documentados en la literatura sobre el consumo de datos bibliográficos publicados como datos enlazados. En este sentido destacan los proyectos Europeana (Haslhofer y Isaac 2011), Biblioteca Nacional de España (Suero, Terrazas y Pérez 2013) y US Library of Congress (Summers et al. 2008). Así mismo han sido desarrolladas herramientas informáticas que facilitan la gestión, organización y almacenamiento de los documentos en de las bibliotecas digitales. Los cambios en este ámbito han hecho de la gestión de metadatos una técnica a considerar para el trabajo con información de cualquier tipo.

En el estudio de (Delgado 2015) aparecen recogidas las instituciones identificadas por el grupo Library Linked Data[10], con colecciones de metadatos bibliográficos publicados siguiendo los principios de los datos enlazados, seleccionándose una muestra de siete instituciones europeas:

[10] del portal thedatahub.ie

1. Unión de Catálogos de Suecia (Libris)
2. Biblioteca Nacional de España (BNE)
3. Biblioteca Nacional Alemana (GNB)
4. Biblioteca Nacional Británica (BNB)
5. Biblioteca Nacional de Francia (BNF)
6. Europeana
7. Open University (OU)

Por otra parte, el grupo de investigación de Web Semántica de la *Universidad de las Ciencias Informáticas* (UCI, por sus siglas en español) ejecuta el proyecto de investigación: "Extracción, publicación y consumo de metadatos bibliográficos como datos enlazados", el cual genera grafos RDF con metadatos bibliográficos que son almacenados en un triplestore para su uso posterior. Los *triplestores* (en español: almacenes de tripletas) son bases de datos que actualmente no se encuentran optimizadas para la realización de consultas formuladas por los usuarios en tiempo cercanos al real.

Los servidores de indización constituyen una solución altamente eficiente para resolver los problemas de búsqueda y recuperación de datos en grandes bases de datos. En el contexto del proyecto de investigación fue implementada por (Rizo y García 2013) una aplicación web (RACien) para el consumo de datos enlazados a partir de un triplestore, sin embargo, los resultados en tiempo de respuesta no fueron los esperados. La Tabla 1 muestra los tiempos (en segundos) que tardan en realizarse cuatro tareas específicas (ver sección 3.4) llevadas a cabo por una muestra de diez usuarios en el consumo de metadatos bibliográficos. La nomenclatura de los resultados es la siguiente: <DBLP/RACien>.

Tabla 1 Tiempos esperados en el consumo de metadatos bibliográficos.

	Tiempo (en segundos)									
Tareas	**U_1**	**U_2**	**U_3**	**U_4**	**U_5**	**U_6**	**U_7**	**U_8**	**U_9**	**U_{10}**
1	25/6	32/8	26/7	23/10	28/9	30/6	27/10	26/7	30/8	29/9
2	5/2	9/4	8/3	10/5	7/9	15/3	6/5	6/5	7/3	9/4
3	6/3	8/5	10/4	14/6	8/5	10/4	7/8	7/8	8/5	10/6
4	8/5	8/6	12/7	18/5	11/6	19/5	9/9	9/7	10/8	12/8

Nótese que a pesar de que la propuesta de (Rizo y García 2013) mejora los indicadores en comparación con DBLP[11] (tomado como ejemplo para el estudio) en algunos casos los resultados se mantienen o se empeoran. En este sentido, se hace necesario contar con estructuras de datos optimizadas (índices) para disminuir el tiempo de respuesta del triplestore a consultas formuladas por los usuarios.

Teniendo en cuenta la situación problemática descrita con anterioridad se plantea como **problema de investigación a resolver** el siguiente**:** ¿Cómo *disminuir el tiempo de respuesta a las consultas formuladas por los usuarios* durante el consumo de metadatos bibliográficos publicados como datos enlazados?

El problema de investigación se enmarca en el **objeto de estudio** de las técnicas de indización y en el **campo de acción** de la indización de grafos RDF.

Esta investigación se propone como **objetivo general** desarrollar un componente de software mediante la programación orientada a objeto que integre un servidor de indización con tripletas RDF existentes en un triplestore.

Para asegurar el cumplimiento del objetivo general se definen los **objetivos específicos** siguientes:

1. Elaborar el marco teórico y el estado del arte del objeto de estudio de la investigación mediante el análisis bibliográfico documental para identificar tendencias y adoptar posiciones al respecto.

2. Diseñar un componente de software para la indización de tripletas RDF a partir de un triplestore.

3. Implementar un componente de software para la indización de tripletas RDF a partir de un triplestore.

4. Validar los resultados obtenidos con la utilización del componente de software desarrollado empleando datos reales mediante la realización de un diseño experimental.

Se plantea como **idea a defender** que si se *desarrolla un componente de software que integre un servidor de indización con tripletas RDF existentes en un triplestore* entonces *disminuirá el tiempo de respuesta a las consultas formuladas por los usuarios* durante el consumo de metadatos bibliográficos publicados como datos enlazados.

Métodos Teóricos:

[11] http://dblp.uni-trier.de/

- Analítico Sintético
- Histórico Lógico
- Modelación
- Inductivo Deductivo

Se pretende explorar, aplicando el método *analítico sintético* las principales aproximaciones existentes en la literatura para resolver el problema de disminución del tiempo de respuesta a las consultas formuladas por los usuarios durante el consumo de metadatos bibliográficos publicados como datos enlazados, de manera que sea posible estudiar en profundidad cada componente, técnica o tecnología involucrada y realizar además una síntesis de los elementos comunes y más importantes de cada teoría y/o aproximación. Se prevé también la utilización del método *histórico lógico* para apreciar cómo han evolucionado en el área del conocimiento durante los últimos años los principales conceptos que forman parte de esta investigación. Para la fundamentación y elaboración del componente de software que integre un servidor de indización con tripletas RDF existentes en un triplestore, así como adicionar a la propuesta de solución aquellos elementos más generales que se abordan en la literatura y que a través del razonamiento lógico se demuestra pueden ser aplicables en este entorno, se pretende el uso de los métodos *modelación* e *inductivo deductivo*. En el primer caso, se formula una generalización de propuesta arquitectónica utilizando técnicas de visualización de la información, así como algunos artefactos de ingeniería que ayuden a comprender sus componentes y sus interrelaciones.

Métodos Empíricos:

- Estudio de Casos
- Medición

Finalmente, fue utilizado el *caso de estudio* como método para evaluar la aplicabilidad del componente de software propuesto empleando datos reales y la *medición* para el cálculo de la efectividad (tiempo) de la propuesta de solución.

Para garantizar la validez de los resultados se precisa disponer de un conjunto de datos confiables y representativos de metadatos bibliográficos que permitan afirmar que los resultados obtenidos son aplicables en los escenarios previstos. Por lo antes mencionado se selecciona como **población** los *grafos RDF con metadatos bibliográficos de revistas científicas latinoamericanas*, mientras por su parte

la **muestra** seleccionada son los *grafos RDF con metadatos bibliográficos de revistas científicas cubanas*.

El documento de tesis está estructurado en tres capítulos. A continuación una breve descripción de cada uno de ellos.

En el **Capítulo 1** se realiza un análisis de la literatura consultada, al tiempo que se presenta una taxonomía de los conceptos fundamentales asociados a la investigación, los cuales permitirán una mejor comprensión de este trabajo. Se muestra un análisis del estado del arte, donde son tratadas las principales técnicas y herramientas para la indización de datos estructurados en la web de los datos; su novedad científica, impacto social y relevancia, con especial énfasis en el dominio de los metadatos bibliográficos en el contexto de las bibliotecas digitales.

En el **Capítulo 2** se presenta un componente de software que integra un servidor de indización con tripletas RDF existentes en un triplestore. Como parte de la propuesta ha sido implementada una plataforma informática utilizando tecnologías actuales del desarrollo de software.

En el **Capítulo 3** se desarrolla un caso de estudio empleando datos reales provenientes de revistas cubanas. Adicionalmente, se realiza un pre-experimento para evaluar los tiempos de respuesta de consultas SPARQL realizadas por los usuarios a un grafo RDF en comparación con el uso de índices. Se describen en detalle los principales resultados obtenidos con el caso de estudio y el experimento desarrollado.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

La ausencia de estructuras de datos optimizadas (índices) en consultas formuladas por los usuarios en grafos RDF con metadatos bibliográficos almacenados en un triplestore, no ayuda a solventar el inconveniente de la disminución del tiempo cercano al real a dichas consultas para solucionar los problemas de búsqueda y recuperación de datos en grandes bases de datos.

En este capítulo, se realiza un análisis de la literatura consultada (epígrafe 1.2), al tiempo que se presenta una taxonomía de los conceptos fundamentales asociados a la investigación (epígrafe 1.3), los cuales permitirán una mejor comprensión de este trabajo. Se muestra un análisis del estado del arte (epígrafe 1.4), donde son tratadas las principales técnicas y herramientas para la indización de datos estructurados en la web de los datos; su novedad científica, impacto social y relevancia, con especial énfasis en el dominio de los metadatos bibliográficos en el contexto de las bibliotecas digitales. Finalmente se exponen los resultados obtenidos luego de realizado un estudio de las aproximaciones más representativas existentes en las fuentes estudiadas (epígrafe 1.5), demostrando que son insuficientes, lo cual justifica el desarrollo de la investigación.

1.2. Análisis bibliométrico y documental

Para la realización de la investigación se llevó a cabo un estudio documental que abarca principalmente la literatura publicada en los últimos cinco años. Se consultaron numerosas fuentes bibliográficas, entre las que se encuentran bases de datos referenciadas como Google Scholar[12], Springer[13] e IEEE[14]. De igual modo fueron visitados los sitios web oficiales de organizaciones de obligada consulta como W3C y tecnologías que constituyen aproximaciones al objeto de estudio de la investigación, tal es el caso de Elasticsearch[15], Apache Jena[16] y Apache Solr[17]. A continuación, se muestra en la Tabla 2 un resumen de la bibliografía consultada.

Tabla 2 Resumen de la revisión bibliográfica consultada.

Tipo de fuente consultada	Cantidad consultada	Cantidad publicada en los últimos cinco años [2012-2016]
Artículos en revistas científicas	52	35

[12] https://scholar.google.com/
[13] http://www.springer.com
[14] http://ieeexplore.ieee.org/Xplore/home.jsp
[15] https://www.elastic.co/
[16] https://jena.apache.org/index.html
[17] http://lucene.apache.org/solr/

Artículos en congresos/conferencias	12	7
Libros	17	4
Secciones de libro	6	2
Informes	1	1
Tesis de pre-grado	2	2
Tesis de maestría	1	1
Tesis de doctorado	1	1
Páginas web	4	3
TOTAL	**96**	**56**

La tabla anterior muestra que se consultaron un total de 96 trabajos científicos, de los cuales 56 fueron publicados en los últimos cinco años, lo cual representa un 58% de la bibliografía consultada. Ver gráfico de pastel de la Figura 1.

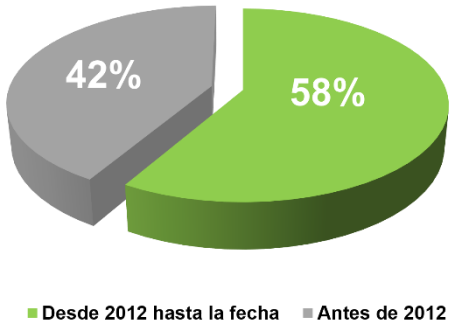


Figura 1 Actualidad de la bibliografía consultada (Fuente: elaboración propia).

1.3. Marco teórico

En este acápite se describen los principales conceptos asociados al dominio del problema con el objetivo de avalar la investigación, permitiendo de esta forma crear la base de conocimientos necesaria para su desarrollo.

1.3.1 Evolución de la web.

En el año 1989 un informático de la *European Organization for Nuclear Research* (CERN, por sus siglas en inglés), en español: Organización Europea para la Investigación Nuclear, llamado Tim Berners-Lee crea lo que hoy conocemos como la web, a fin de dar solución a la perdida de información y facilitar el intercambio de la misma entre los científicos de dicho centro de investigación, comentan (Delgado y Puente 2013).

Fue por aquellos años que surgió el *Uniform Resource Identifier* (URI, por sus siglas en inglés), en español: Identificador de Recursos Uniforme, para identificar los recursos de una red de forma unívoca. Aparecen también el *Hypertext Markup Language* (HTML, por sus siglas en inglés) para la presentación de la información y el *Hypertext Transfer Protocol* (HTTP, por sus siglas en inglés) como protocolo para el intercambio de datos.

Paralelamente al crecimiento de la web, las tecnologías que la hacen posible han experimentado una rápida evolución. Primero los contenidos eran estáticos, creados por desarrolladores web y donde los usuarios solo accedían a ella para consultarlos. Esta primera versión de la web se identifica en la literatura como Web 1.0 y se extendió desde los años 1990 al 2000. Luego, la generación de contenidos dinámicos, o lo que se conoce como Web 2.0, logró mayor interacción de los usuarios, que ya no sólo eran quienes consultaban las páginas web, sino que además las mantenían; es así que emergen de igual modo tanto los *Content Management Systems* (CMS, por sus siglas en inglés) como las redes sociales (del inglés: *social networks*) y con ellas mayores volúmenes de información compartida.

1.3.2 Limitaciones de la web actual.

Dicha información, apunta (Tello 2006), se encuentra almacenada en la web mediante lenguajes de etiquetado, entiéndase estructurada en base a HTML (que está orientado el texto y no al dato) y que únicamente describe la forma en que dicha información debe ser presentada al usuario por el navegador (díganse: colores, maquetación, interacción, entre otros), pero no pueden ser procesadas por una máquina puesto que no expresan explícitamente su significado, dicho de otra forma, su semántica[18], como señalan (Dadzie y Rowe 2011). Por otra parte la información en la Web se encuentra dispersa y no existe relación explícita entre los diferentes recursos, lo que provoca ambigüedad en la información, exponen (Dadzie y Rowe 2011), mientras que imposibilita su descubrimiento y utilización por sistemas informáticos. La situación antes descrita provoca que los motores de búsquedas más utilizados en la actualidad como Google[19] y Yahoo[20] resulten imprecisos y, en muchos casos, no satisfagan las necesidades de búsqueda de los usuarios al responder consultas basadas en palabras claves, no siendo capaces de recuperar la información a partir de consultas expresadas en lenguaje natural (Sudeepthi, Anuradha y Babu 2012).

1.3.3 La Web Semántica.

[18] Definida por la Real Academia Española (http://www.rae.es) como el: *"Conjunto de unidades léxicas de una lengua que comparten un núcleo común de rasgos de significad."*
[19] https://www.google.com.cu/
[20] https://www.yahoo.com/

La Web Semántica o Web 3.0 como igualmente se le denomina, es enunciada como concepto por Tim Berners-Lee, su creador y miembro fundador del *World Wide Web Consortium* (W3C[21], por sus siglas en inglés) de la siguiente forma: *"La Web Semántica no pretende sustituir la Web actual, sino que es una extensión de la misma en la que la información tiene un significado bien definido, posibilitando a los humanos y las computadoras trabajar en cooperación"* (Berners-Lee, Hendler y Lassila 2001).

La web de los datos, como también es reconocida, intenta resolver las limitaciones antes expuestas que impiden aprovechar todas sus potencialidades, dígase: *formato*, *integración* y *recuperación* de la información, como explican (Delgado y Puente 2013), al mismo tiempo que permita enfrentar los desafíos de la actual sociedad de la información. Ésta no es una web nueva, se soporta sobre la plataforma tecnológica de la web actual. Mientras que la web actual contiene documentos, páginas de texto e imágenes que están diseñadas únicamente para el entendimiento humano y de los navegadores web; la Web Semántica, por su parte, añade soporte para grandes colecciones de información organizadas. Por otro lado, incrementa la navegabilidad y el descubrimiento de información fuertemente relacionada, mejorando los servicios de búsqueda y recuperación de la información.

El objetivo de la Web Semántica, señala (Tello 2006) es que toda la información presente en la WWW sea comprensible no sólo por los humanos sino también por las propias máquinas, produciéndose una conversión de la estructura de la web en una gran estructura de almacenamiento de la información. Esto dicho de otro modo es catalogar la información de los recursos web, páginas HTML, documentos PDF, videos, archivos de sonido y otros, mediante lo que se denomina ontologías, entiéndase, mediante el significado de las palabras, no mediante palabras claves.

1.3.4 Las ontologías.

El término ontología es utilizado con diferentes significados en diferentes comunidades. Su origen se encuentra en la filosofía y son utilizadas para estudiar la naturaleza del ser y la existencia (Delgado 2015). La definición del diccionario de la real academia enuncia el concepto de la siguiente manera: *"parte de la metafísica que trata del ser en general y de sus propiedades trascendentales"* (Martínez 2014).

En computación, una definición ampliamente aceptada por la comunidad científica es la siguiente: una ontología es una especificación *explícita* y *formal* sobre una *conceptualización compartida* (Studer, Benjamins y Fensel 1998). Es explícita porque es descrita en términos de un lenguaje; formal ya que usa vocabularios comunes para ser comprensible por una máquina y que sea una conceptualización

[21] http://www.w3c.org

compartida se debe a que es una forma de describir y representar un dominio (cierta área de interés) de acuerdo al consenso entre un grupo o varias partes. Las ontologías definen conceptos y relaciones de algún dominio, de forma compartida y consensuada (Gruber 1993). Refieren (Chávez, Cárdenas y Benito 2005) que las ontologías definen las relaciones entre conceptos y especifican reglas lógicas para que las máquinas multipliquen su capacidad de procesar y comprender los datos. Atendiendo a estas acepciones los distintos autores tienen como puntos coincidentes: la presencia de conceptos y relaciones, enmarcados en un dominio y de forma compartida. Sin embargo la definición de (Studer, Benjamins y Fensel 1998) es la que posee un mayor grado de formalidad para el ambiente académico, y es justamente por eso la adoptada en la investigación.

Las ontologías se componen de: *los conceptos*, que son las ideas básicas que se intentan formalizar, pueden ser clases de objetos, métodos, planes, estrategias, procesos de razonamiento, etcétera; *las relaciones*, que representan la interacción y enlace entre los conceptos del dominio, suelen formar la taxonomía del dominio; *las funciones*, que son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología; *las instancias*, que utilizan para representar objetos determinados de un concepto; y *los axiomas o reglas de restricción*, como teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología.

Se puede representar con ontologías en la web: productos, catálogos, recursos humanos, material educativo, servicios, plataforma, dispositivos y perfil de usuarios, pero no es una tarea trivial, para facilitarla aparecen los editores de metadatos. Los más importantes son: Reggie, que es muy sencillo; OntoWeb, más completo; y Protegé (Chávez, Cárdenas y Benito 2005).

1.3.5 Los metadatos.

Con el propósito de solucionar el inconveniente del formato de la información disponible en la Web, de manera que sea posible representar su valor semántico y luego ser recuperado de manera automática, surgen las anotaciones semánticas. Estas constituyen un proceso mediante el cual se adicionan metadatos semánticos a los recursos web (Fernández 2009), combinando conceptos de metadatos y ontologías. Es decir, los campos de los metadatos son asociados con términos en una ontología, los cuales se utilizan para describir estos campos (Macário, de Sousa y Medeiros 2010).

En (Greenberg 2003) se definen los metadatos como datos estructurados alrededor de un objeto que es compatible con las funciones asociadas con el objeto designado; es por eso que los metadatos se define comúnmente como datos sobre datos, de acuerdo con su etimología (Sicilia 2014); en este contexto los datos describen recursos en la web. El término metadatos se define a menudo como "datos

sobre los datos". Esta definición básica no es muy informativa, sin embargo para (Tkaczyk et al. 2014), los metadatos son datos altamente estructurados que describen información, describen el contenido, la calidad, la condición y otras características de los datos. Como es apreciable son muchas las catalogaciones que recibe el término “metadato”, pero si se comparan los aquí citados (Tkaczyk et al. 2014) logra una enunciación más concreta, por cuanto es la manejada en la investigación.

Gracias a la semántica en la web, el software es capaz de procesar su contenido, razonar con éste, combinarlo y realizar deducciones lógicas para resolver problemas cotidianos automáticamente. Un término asociado es *unicode* que representa como menciona (Chávez, Cárdenas y Benito 2005) una codificación de textos que permite utilizar los símbolos de diferentes idiomas sin observar caracteres extraños. Esto permite expresar información en la Web Semántica en cualquier idioma.

A pesar de que las anotaciones semánticas constituyen una solución práctica para el problema del formato, aún existe la necesidad de integrar la información (entiéndase como la interrelación, en base a algún criterio entre recursos web), como explican (Rizo y García 2013), de modo que sea posible descubrir conocimiento a través de relaciones implícitas en el formato de los recursos. Por otro lado, la recuperación de la información es otra limitación importante y está condicionada por las limitaciones anteriores: formato e integración.

1.3.6 Los datos enlazados.

El tránsito hacia la Web Semántica requiere de una adecuada estructuración e integración de la información; esto propició que (Berners-Lee 2006) enunciara el concepto de Datos Enlazados (del inglés: *Linked Data*): *“Los datos enlazados se refieren a un conjunto de buenas prácticas para la publicación y enlazado de datos estructurados en la Web”*. La idea que persiguen los datos enlazados es utilizar la arquitectura general de la web para la compartición de datos estructurados a escala global (Heath y Bizer 2011).

En (Berners-Lee 2006) se describen los cuatro principios básicos de los datos enlazados. En primer lugar: *identificar*, donde se utiliza una URI para identificar cada recurso publicado en la web; segundo: *publicar*, cuyo propósito es tener publicados estos datos en una URI basada en HTTP para que luego puedan ser fácilmente localizados y consultados; como tercero: *describir*, a fin de proporcionar información útil, detallada o extra acerca del recurso cuando se acceda a esta URI basada en HTTP usando estándares; y cuarto punto: *enlazar*, para finalmente incluir enlaces a otras URI relacionadas con los datos contenidos en el recurso, de forma que se potencie el descubrimiento de la información sobre la web.

Para (Chávez, Cárdenas y Benito 2005) y (Tello 2006), la Web Semántica se basa en dos conceptos fundamentales: (1) la descripción del significado que tiene los contenidos en la web y (2) la manipulación automática de estos significados.

Cuando se habla del primero, entonces intervienen los conceptos como: *la Semántica*, que es el estudio y significado de los términos lingüísticos procesables por las máquinas; *los Metadatos* como contenedores de información semántica sobre los datos; y *las Ontologías* para definir conceptos y relaciones de un dominio específico. Los metadatos y las ontologías forman parte del campo de la representación del conocimiento. Para describir la semántica se requiere de un lenguaje apropiado llamado: lenguaje de representación.

1.3.7 Los lenguajes de representación.

El primer lenguaje para la construcción de la Web Semántica fue *SHOE*, creado por Jim Hendler en la Universidad de Maryland en 1997. Desde entonces, expone (Chávez, Cárdenas y Benito 2005), se han definido otros lenguajes y estándares con finalidad similar, como *XML*, *RDF*, *DARPA Agent Markup Language* (DAML+OIL, por sus siglas en inglés, referidas al nombre de un programa de EE.UU para la *Defense Advanced Research Projects Agency* (DARPA, por sus siglas en inglés)), y más recientemente OWL, por citar los más importantes.

XML

XML (del inglés: *eXtensible Markup Language*) es el estándar emergente para el intercambio de datos en la web, estructurando datos y documentos en forma de árboles de etiquetas con atributos sin jerarquía de clases. Dicho lenguaje aporta la sintaxis superficial para los documentos estructurados, pero sin dotarles de ninguna restricción sobre el significado. XML es un subconjunto de SGML (del inglés: *Standard Generalized Markup Language*), en español: Lenguaje de Etiquetado Generalizado Estándar, y define un formato de texto diseñado para la transmisión de datos estructurados.

Por otra parte, XMLNS (del inglés: *eXtensible Markup Language Namaspace Specification*) o los espacios de nombre XML, proporcionan un método simple para cualificar elementos y nombre de atributos usados en documentos XML asociando estos con un espacio de nombre identificado por referencias URI. En síntesis, proporciona un método para evitar conflictos de nombres de los elementos (Bray, Hollander y Layman 1999), mientras que XML Schema (en español: Esquema XML) es un lenguaje para definir la estructura de los documentos XML.

RDF

RDF fue desarrollado gracias a los auspicios de la W3C. En (Manola y Miller 2004) se define como un lenguaje para representar la información acerca de los recursos en la WWW. Está destinado especialmente para la representación de metadatos sobre recursos web, como el título, autor y fecha de modificación, entre otros, así como la disponibilidad para algunos recursos compartidos. RDF se encuentra recogido en 6 recomendaciones del W3C: *Primer*, *Concepts*, *Syntax*, *Semantics*, *Test Cases* y *Vocabulary (Schema)*, usado para describir las propiedades y las clases de los recursos RDF con una semántica para establecer jerarquías de generalización entre dichas propiedades y clases.

Para (Tello 2006) RDF es un modelo de datos en forma de grafo dirigido y etiquetado que permite definir relaciones semánticas entre distintas URIs como recursos, asociándoles un conjunto de propiedades y valores con el fin de representar información sobre recursos en la WWW. Por su parte (Schreiber y Raimond 2014) lo define como un marco para expresar la información acerca de los recursos. Los recursos pueden ser documentos, personas, objetos físicos, y los conceptos abstractos.

RDF está basado en la idea de que los recursos (sujeto) a describir, poseen propiedades (predicado) que a su vez tienen valores (objeto), ver Figura 2. Estos recursos pueden ser descritos formulando "declaraciones" que especifican estas propiedades y valores, en forma de grafo (del inglés: *graph*) de nodos y arcos que representan los recursos, y sus propiedades y valores (Agudelo y Reyes 2012).

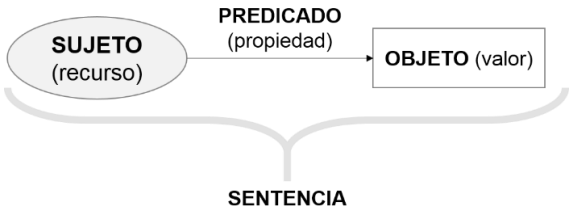


Figura 2 Modelo de datos basado en grafos RDF (Fuente: elaboración propia).

RDF y el lenguaje RDF Schema (en español: esquema RDF) se fundamentaron en investigaciones sobre metadatos realizadas por comunidades de Bibliotecas Digitales, pudiendo considerarse RDF como una implementación del *Warwick Framework* (WF, por sus siglas en inglés) donde RDF es una evolución de este último, que permite que cada vocabulario de metadatos posea una sintaxis distinta. Para (Agudelo y Reyes 2012) en RDF es fundamental utilizar palabras que transmitan un significado inequívoco con el fin de que las aplicaciones entiendan el enunciado para un procesamiento correcto. Cabe hacer notar que el lenguaje RDF mismo no es independiente de la sintaxis, ya que esta implementado sobre XML.

OWL

OWL (del inglés: *Web Ontology Language*), en español: Lenguaje de Ontologías para la Web; es un lenguaje utilizado en Web Semántica diseñado para representar conocimiento complejo acerca de cosas, grupos de cosas y relaciones entre las cosas. Es un lenguaje basado en lógica computacional de modo que el conocimiento expresado en OWL puede ser razonado por programas de computadoras que verifican la consistencia del conocimiento permitiendo que el conocimiento implícito se convierta en conocimiento explícito (Delgado y Puente 2013).

1.3.8 Índice e indización.

En los comienzos de la web etiquetar categorías y añadir metadatos no fue una prioridad, pues la mayor parte del contenido de la web estaba diseñada para leer, no para que fuera manipulada por ordenadores, robots y agentes (Chávez, Cárdenas y Benito 2005). El progreso y crecimiento masivo de las redes de computadoras y medios de almacenamiento a lo largo de los últimos años, motiva la aparición de un creciente interés por los sistemas de clasificación automática de documentos (Hernández y Hernández 2008).

La W3 como también se le asocia a la WWW, según (Lobo 1999), ha evolucionado hacia lo que podría considerarse un dinámico almacén donde albergar informaciones muy diversas en contenido, relevancia y utilidad. Pronto se ha visto que, ante las colosales dimensiones de la web y la heterogénea información que contiene, es necesario no sólo organizar la información, sino también clasificarla y categorizarla con el fin de poder realizar búsquedas y recuperar la información de forma automática. Esto sólo es posible mediante una indización y clasificación previas, mediante la generación de documentos previamente estructurados formal y semánticamente, y utilizando un lenguaje que sea independiente de la plataforma o aplicación empleadas (Chávez, Cárdenas y Benito 2005).

A decir de (Lobo 1999), las herramientas de búsqueda de información en la W3 desarrollan diferentes métodos y mecanismos para la recopilación e indización de la información que incorporan a sus bases de datos. La diversidad documental, de contenidos y formatos dificultan el proceso. Los buscadores de la W3 trabajan moviéndose de un documento a otro, descendiendo progresivamente a través de los hiperenlaces; un programa de indización que indiza la información de los millones de páginas web ubicadas en los servidores conectados a la red y enormes bases de datos a las que acceden los usuarios a través de la interfaz del buscador. Por tanto, los buscadores no sólo deben facilitar la localización de los recursos, sino que, además, deben compilarlos. Salvo en el caso de los directorios temáticos como Yahoo, Olé y otros, la *indización automática* es el método predominante utilizado por las herramientas de consulta de la W3. Cuando se requiere hacer uso de búsquedas en todos los

archivos, palabras o frases relacionadas, para el caso de archivos muy grandes, como resultado es muy probable que ocurran fallas o imprecisiones (Hernández y Hernández 2008).

Frente a este panorama, considera (Lobo 1999), no sería factible pretender hoy día, la *indización manual* de todo el espacio web, pues por ejemplo la indización a texto completo llevada a cabo por los buscadores generales no permite incluir, entre otros: ficheros con formato de tipo PDF. Con las formas actualmente adoptadas para representar la información en Internet, se están desaprovechando las potencialidades, aún no explotadas totalmente, con que cuenta la web. El uso correcto y normalizado de los metadatos y la adopción de estructuras de almacenamiento ricas semánticamente serían alternativas útiles para paliar gran parte de los inconvenientes derivados del uso de robots y favorecer una indización de calidad.

En opinión de (Hernández y Hernández 2008), los índices, como estructuras de datos optimizadas, permiten transformar el texto en un formato donde la búsqueda sea más rápida, eliminando el proceso de exploración lento a consultas formuladas por los usuarios. Este proceso de conversión es llamado *indización* mientras al archivo resultante se le llama *índice*. La indización es, entonces, un requisito necesario para un adecuado almacenamiento y recuperación de la información contenida en un fondo documental. Definiéndolo de otra manera, (Cleveland y Cleveland 2013) lo asumen como una lista de información bibliográfica o citas hacia un cuerpo literario, usualmente arreglados en orden alfabético y basado en algunos datos específicos, tales como autor, tema o palabras claves. Si bien ambos pensamientos están en similar línea de pensamiento, en la investigación se asume la declaración de (Hernández y Hernández 2008) acerca del término por considerarse más ajustada.

Un índice separa las palabras del documento en campos y permite el acceso rápido a los datos que fueron almacenados en el proceso de indización. Su trabajo consiste en distinguir la información relevante contenida en los documentos. Por otra parte, un índice se forma por segmentos, documentos, campos y términos. Cada índice contiene uno o más segmentos. Un segmento se forma por uno o más documentos. Cada documento tiene uno o más campos y cada campo se forma por uno o más términos. Cada término es un par de secuencias que representan un nombre de campo y un valor. Un segmento consiste en una serie de archivos. El número exacto de los archivos que constituyen cada segmento varía de índice a índice, y depende del número de los campos que el índice contiene.

La indización, aseguran (Suárez, Navarrete y León 2007), es una de las etapas del procesamiento analítico sintético de la información. Se define como la enumeración sucesiva de los diferentes encabezamientos (términos) que expresan el(los) tema(s) contenido(s) en un documento, y que requiere de la aplicación de criterios uniformes; así como del establecimiento previo de una lista de

términos en la cual se basa dicha indización. Su importancia radica esencialmente en la necesidad de habilitar un sistema de búsqueda y recuperación de la literatura científica existente en los fondos documentarios de las entidades informativas. El producto final de este proceso es generalmente un índice bibliográfico, una base de datos automatizada o simplemente un catálogo alfabético de materias manual, indispensables para asegurar el acceso y consulta de la información a los usuarios.

Para lograr la indización correcta de un documento o solicitud de búsqueda es necesario utilizar los lenguajes de indización existentes. Estos lenguajes artificiales, llamados lenguajes de búsqueda informativa, lenguajes de indización, lenguajes documentales, lenguajes de almacenamiento y recuperación, entre otras denominaciones, son herramientas auxiliares, creadas por el hombre con el propósito de expresar el contenido semántico fundamental de los documentos o solicitudes de información y localizar la información que responda a las necesidades de los usuarios.

Para considerar que un documento se indizó correctamente es necesario considerar dos aspectos fundamentales en la indización: (1) *la exhaustividad* y (2) *la especificidad*. La exhaustividad se define como la cantidad de conceptos considerados que son representativos del contenido íntegro de un documento. La especificidad es el nivel de detalle y exactitud de la representación de un concepto particular. Indica (Peña 2003) que en la indización automática, la máquina separa cadenas de caracteres ya sea en el título, en el resumen, descartando únicamente las llamadas palabras vacías o reconociendo en el texto completo los sintagmas nominales.

La búsqueda es en sí, el proceso de entrar al índice y buscar palabras relacionadas, para encontrar documentos donde aparezcan. En este sentido los *servidores de indización*, constituyen una solución eficiente para resolver los problemas de búsqueda y recuperación de datos en grandes bases de datos; problemas para los que las bibliotecas digitales trabajan con el objetivo de elevar su eficiencia, como dejan saber (Suárez, Navarrete y León 2007)

La satisfacción de las necesidades de información de los usuarios, es el elemento fundamental para la realización de búsquedas basadas en datos estructurados. Los usuarios que conocen la estructura y el esquema de los datos, son capaces de formular consultas que responden a sus necesidades de información. Sin embargo, aún es necesario proveer de interfaces intuitivas a aquellos usuarios que desconocen de estos elementos, en aras de que también les sea posible satisfacer sus necesidades de información.

En el estudio realizado por (Carrasco et al. 2004) se plantea que existen tres tipos de fuentes de indización.

1. **Las bases de referencias**; entre las que se encuentran Medline (Index Medicus y bases especializadas), EMBASE (Excerpta Medica), BIOSIS (Biological Abstracts), LILACS (Literatura Latinoamericana y del Caribe de Información en Ciencias de la Salud), ERIC (Education Resources Information Center), EconLit, Sociological Abstracts, PsycINFO, FSTA (Food Science & Technology Abstracts), Compendex, y MathSc.

2. **Las bases de datos de texto completo**; como SciELO.

3. **Índices de citaciones**; como la Science Citation Indexes y el Journal of Citation Reports (JCR), del Institute for Scientific Information (ISI-Thomson).

1.4. Estado del arte

El desarrollo de proyectos para la indización de grafos RDF en la web implica tomar un conjunto de decisiones técnicas y metodológicas complejas. ¿Cuáles son las técnicas y herramientas existentes para la indización de grafos RDF en el dominio de los metadatos bibliográficos? ¿Cuáles son las principales limitaciones que poseen estas herramientas? ¿Qué otros elementos significativos giran en torno a iniciativas de este tipo? ¿Cuáles son los pasos principales que se deben tener en cuenta a la hora de acometer el desarrollo de un proyecto con estas características? En este epígrafe se realiza un análisis crítico de las principales aproximaciones existentes para la indización de grafos RDF como datos estructurados en la web de los datos, con énfasis en el dominio de los metadatos bibliográficos.

1.4.1 Trabajos relacionados.

La forma de potenciar el consumo (visualización, presentación, utilización) de las fuentes de datos publicadas como datos enlazados de manera intuitiva y amigable para los usuarios de la web ha sido una temática abordada por varios proyectos. Las aplicaciones de software desarrolladas con este propósito pueden ser clasificadas básicamente en dos grandes grupos: (1) aplicaciones genéricas y (2) aplicaciones para un dominio específico (Heath y Bizer 2011). El primer tipo de aplicaciones permite realizar el consumo de datos enlazados desde cualquier dominio temático, de modo que es irrelevante la naturaleza del dato, pueden ser datos geográficos, de ciencias de la vida, bibliotecas, entre otros.

Existen dos tipos de aplicaciones que pertenecen a la clasificación de aplicaciones genéricas: (1.1) navegadores de datos enlazados y (1.2) los motores de búsqueda de datos enlazados. Al primer grupo pertenecen aplicaciones como Disco Hyperdata-Browser (Heath 2008), Tabulator (Berners-Lee 2006), Marbles[22], entre otras. Los navegadores de datos enlazados son útiles pues proveen una experiencia

[22] http://marbles.sourceforge.net

de navegación fluida y similar a la navegación sobre hipertexto de la Web tradicional, sin embargo, estos no permiten una visión general del conjunto de datos que se está navegando, ya que solo muestra la información del recurso actual, es decir solo visualiza información del recurso al cual se haya accedido a través de la URI que lo representa.

Los motores de búsquedas de datos enlazados por su parte, mejoran la experiencia del usuario en relación a las tradicionales formas de búsqueda, al realizar la misma sobre datos estructurados. Sin embargo, en estas aplicaciones tampoco es posible conocer a priori a través de una vista general del conjunto de datos sus principales recursos, propiedades y valores. A este grupo pertenecen aplicaciones como: *Sig.ma* (Tummarello et al. 2010), *VisiNav* (Harth 2010) y *Swoogle* (Heath y Bizer 2011)

Las aplicaciones para un dominio específico cubren las necesidades de determinadas comunidades de usuarios, dentro de esta categoría se encuentran: (2.1) los integradores de datos enlazados y (2.2) otras aplicaciones de dominio específico que incluyen funcionalidades de búsqueda y navegación. Los integradores de datos enlazados son aplicaciones creadas con la finalidad de integrar información desde fuentes heterogéneas (conjuntos de datos entrelazados en la web de los datos) para utilizarlas con el propósito de satisfacer las necesidades de información de una determinada comunidad de usuarios. A esta categoría pertenecen aplicaciones como: US Global Foreign Aid Mashup[23] y Paggr (Nowack 2009). Los integradores de datos enlazados son herramientas útiles, puesto que permiten recuperar información interrelacionada entre diversos conjuntos de datos. Sin embargo, estas aplicaciones tampoco resuelven el problema de caracterizar el conjunto de datos a través de una vista general de sus recursos y propiedades.

De modo que las herramientas existentes para el consumo de datos enlazados dificultan que los usuarios no técnicos puedan explorar el conjunto de datos con la finalidad de conocer qué tipos de recursos se encuentran en el mismo, cuáles son sus propiedades y cómo estas se interrelacionan.

1.4.2 Paradigmas de búsqueda.

Para los usuarios no técnicos que desconocen la estructura y forma de consultar los datos, resulta indispensable contar con mecanismos que permitan la búsqueda de información sin un conocimiento previo de estos dos elementos. Por tanto, se hace necesario desarrollar interfaces con soporte para realizar búsqueda intuitiva, que sean además fáciles de usar y sobre todo capaces de satisfacer las necesidades de información de los usuarios que interactúan con ellas. Una solución a este problema

[23] http://data-gov.tw.rpi.edu/demo/USForeignAid/demo-1554.html

es la inclusión de funcionalidades basadas en paradigmas de búsqueda en las interfaces de usuario.

En la concepción tradicional de la web se utilizan paradigmas de búsqueda, cuya aplicación ha sido extendida al contexto de los datos enlazados. Los paradigmas de búsqueda existentes se clasifican en tres categorías: (1) **palabras clave**, (2) **iterativo-exploratorio** y (3) **lenguaje natural** (Tran y Mika 2012).

1.4.2.1 Búsqueda textual

La formulación de necesidades de información a través de palabras claves, es un paradigma que está siendo aplicado a las aproximaciones de búsqueda en la Web Semántica que operan con datos enlazados. Muchos motores de búsqueda semánticos realizan búsqueda de entidades, por ejemplo: *Falcons* (Cheng y Qu 2009) y *Sig.ma* proporcionan una interfaz que responde a necesidades de información de los usuarios a través de la búsqueda por palabras claves.

Este paradigma ha sido implementado también en motores de búsqueda como SemSearhPro (Tran, Herzig y Ladwig 2011), Tastier (Li et al. 2009) e IBM's Avatar (Kandogan et al. 2006). Estos permiten satisfacer necesidades de información complejas al relacionar entre sí las entidades obtenidas mediante la búsqueda por palabras claves.

La ventaja principal de este paradigma está dada en que permite utilizar mecanismos de indización, lo que hace que solo sea necesario indizar una sola vez, y luego consultar el índice cada vez que se realice una consulta, esto agiliza y simplifica el proceso de búsqueda.

No existe una sintaxis estándar para expresar consultas de búsqueda textual en SPARQL (solo es posible utilizar expresiones regulares para buscar coincidencias con cadenas de texto en objetos del grafo RDF). Sin embargo, cada proveedor de almacén de tripletas RDF proporciona extensiones específicas para la búsqueda textual. Por tanto, es posible realizar consultas híbridas que combinan SPARQL y búsqueda textual, estas consultas arrojan mejores resultados que los obtenidos por consultas expresadas en SPARQL utilizando expresiones regulares para soportar la búsqueda textual sobre RDF (Maali, Cyganiak y Peristeras 2011).

El paradigma de búsqueda a través de palabras claves presenta dos desventajas fundamentales. La primera de ellas es la posible ocurrencia de ambigüedades en las búsquedas (cuando una palabra tiene más de un significado). La segunda desventaja radica en que este paradigma asume un escenario de búsqueda donde el usuario tiene precisión en sus necesidades de información, de modo que requiere de un conocimiento previo del dominio de interés para la búsqueda. Esto puede resultar un obstáculo

en las situaciones más comunes, es decir, en escenarios donde las necesidades de información no son precisas o están vagamente definidas (el usuario desconoce con certeza qué debe buscar). En estos casos es conveniente utilizar el paradigma iterativo/exploratorio.

El paradigma iterativo/exploratorio permite realizar una búsqueda iterativa mediante la navegación exploratoria de un conjunto de datos. La navegación puede ser realizada mediante facetas o a través de la visualización de grafos (Tran y Mika 2012).

1.4.2.2 Búsqueda facetada

La navegación facetada constituye una técnica para la exploración de datos estructurados basada en la teoría de la faceta (Ranganathan y Palmer 1959). Esta técnica permite explorar conjuntos de datos a través de dimensiones conceptuales ortogonales, también llamadas facetas, las cuales no son más que representaciones de las características importantes de los elementos o recursos. La principal ventaja de la navegación facetada radica en las facilidades que esta ofrece al responder a necesidades de información en un escenario de búsqueda donde el usuario no tiene claro qué desea buscar, ni cómo debe hacerlo (escenario de búsqueda vago).

En la búsqueda por facetas, no es necesario un conocimiento a priori del esquema de los datos, ya que, al implementar el paradigma exploratorio, las necesidades de información del usuario se satisfacen a través de la exploración (navegación) del conjunto de datos. La navegación facetada, además, evita la ambigüedad y las consultas se construyen de forma estructurada, utilizando el lenguaje de consultas SPARQL (Oren, Delbru y Decker 2006).

1.4.3 Principales aproximaciones existentes.

Entre las principales aproximaciones existentes de esta investigación fueron estudiadas las técnicas y herramientas de indización, así como los patrones de interacción y los componentes de la arquitectura de la información.

1.4.3.1 Técnicas de indización.

Inverted Index

Un índice invertido (también denominado archivo de publicaciones o archivo invertido) es una estructura de datos de almacenamiento de un mapeo de contenido, tales como palabras o números, en relación a sus ubicaciones en un archivo de base de datos o en un documento o un conjunto de los documentos (nombrado en contraste con un índice de Forward, que mapea a partir de documentos de contenido).

El propósito de un índice invertido es permitir búsquedas rápidas de texto completo, a un costo de aumento del procesamiento cuando se añade un documento a la base de datos. Es la estructura de datos más popular utilizada en sistemas de recuperación de documentos.

Existen dos variantes principales de índices invertidos: (1) un índice de nivel de grabación invertida (o índice de archivo invertido o simplemente archivo invertido) que contiene para cada palabra una lista de referencias a los documentos donde esta aparece y (2) un índice invertido a nivel de palabra (o índice invertido total o lista invertida) que contiene, además, las posiciones de cada palabra dentro de un documento. La última forma ofrece más funcionalidades (como búsquedas de frase), pero necesita más potencia de procesamiento y espacio para su creación.

La estructura de datos de índice invertido es un componente central de un algoritmo típico de indización de los motores de búsqueda. Un objetivo de una aplicación de motor de búsqueda es la optimización de la velocidad de la consulta: encontrar los documentos donde se produce una palabra cualquiera. Una vez que se desarrolla un índice de avance, que almacena listas de palabras por documento, se invierte la próxima para desarrollar un índice invertido. Consultar el índice hacia adelante requiere iteración secuencial a través de cada documento y para cada palabra para verificar un documento coincidente. El tiempo, la memoria y los recursos de procesamiento para llevar a cabo una consulta de este tipo no siempre son técnicamente realista. En lugar de enumerar las palabras por documento en el índice hacia adelante, la estructura de datos índice invertido se enumeran los documentos por palabra.

Suffix tree

Un árbol de sufijos (también llamado árbol de PAT o, en una forma anterior, árbol de posición) es un *trie* (estructura de datos en tipo de árbol) comprimido que contiene todos los sufijos del texto dado como sus claves y posiciones en el texto como sus valores. Éstos implementan particularmente rápido muchas operaciones de cadenas importantes y proporcionan una de las primeras soluciones en tiempo lineal para el problema de la subcadena común más larga. Estas aceleraciones tienen un costo: el almacenamiento de un árbol así normalmente requiere mucho más espacio que el almacenamiento de la propia cadena.

La construcción de un árbol de este tipo para la cadena S lleva tiempo y espacio lineal en la longitud de S. Una vez construido, varias operaciones se pueden realizar rápidamente, como: la localización de una subcadena en S, la localización de una subcadena si se permite un cierto número de errores, la localización de coincidencias en un patrón de expresión regular, entre otras. Los árboles de sufijos se

pueden utilizar para resolver un gran número de problemas de la secuencia que se producen en edición de textos, búsqueda de texto libre, biología computacional y otras áreas de aplicación.

Document-term matrix

Una matriz de término de documentos, es una matriz matemática que describe la frecuencia de los términos que se producen en una colección de documentos. Hay varios esquemas para determinar el valor que cada entrada de la matriz debe tomar. Una de estas iniciativas es TF-IDF. Son útiles en el campo del procesamiento del lenguaje natural. Al crear una base de datos de términos que aparecen en un conjunto de documentos, la matriz contiene filas correspondientes a los documentos y columnas correspondientes a los términos.

Las técnicas de indización tratadas difieren en cuanto a características y la forma de realizar la indización. Los índices invertidos destacan como la técnica empleada por las herramientas de indización estudiadas (ver sección 1.4.3.2), por cuanto, es valorado su uso en el mejor interés de la investigación.

1.4.3.2 Herramientas de indización.

Apache Solr[24]

El *plugin* Solr Grails integra un modelo de dominio Grails con el motor de búsqueda de Apache Solr a través de la API (del inglés: *Application Programming Interface*), en español: Interfaz de Programación de Aplicaciones, SolrJ. Con él es posible indizar Grails al modelo de dominio; consultar tanto los datos de dominio Grails y cualquier otro documento en el índice Solr; y es lo suficientemente flexible como para trabajar con despliegues/índices existentes en Solr. Un índice en Solr posibilita llevar a cabo de manera óptima: la búsqueda de texto completo, agregados y filtrado. Solr acepta documentos JSON, pudiendo transformar su documento RDF en un documento JSON-LD (que es formato de serialización RDF).

Por otra parte, SolrRDF (entiéndase Solr + RDF) es un conjunto de extensiones para la gestión de Solr (índice y búsqueda) de datos RDF siendo posible indizar triples al clúster y realizar consultas SPARQL (ASK, CONSTRUCT, SELECT y DESCRIBE) y actualizaciones (por ejemplo, INSERT, DELETE) a cualquier nodo del clúster obteniendo como respuesta un XML, siendo compatible con SPARQL Endpoint 1.1.

[24] http://lucene.apache.org/solr/

Elasticsearch[25]

Elasticsearch es un motor de búsqueda y análisis distribuido en tiempo real de código abierto y construido encima de Apache Lucene. Permite explorar sus datos a una velocidad y en una escala nunca antes había sido posible. Se utiliza para la búsqueda de texto completo, búsqueda estructurada, análisis, y los tres en combinación. Elasticsearch también se puede describir como un almacén de documentos donde cada campo es indizado y buscado, siendo capaz de escalar a cientos de servidores y petabytes de datos estructurados y no estructurados. Est escrito en Java y utiliza Lucene internamente para la totalidad de su indización y búsqueda, pero tiene como objetivo hacer búsqueda de texto completo fácil ocultando las complejidades de Lucene detrás de una API simple y coherente.

Java API brinda dos clientes incorporados: (1) el cliente de nodo se une a un grupo local como un nodo sin datos. En otras palabras, no lleva a cabo ningún dato en sí, pero se sabe qué datos vive en el cual los nodos del clúster, y puede reenviar solicitudes directamente al nodo correcto; (2) el cliente de transporte para enviar peticiones a un clúster remoto. Que no se une al grupo en sí, sino que simplemente reenvía las solicitudes a un nodo en el clúster.

ARQ Jena[26]

En Jena, toda la información de estado proporcionada por un conjunto de tripletas RDF está contenido en una estructura de datos llamada Modelo. El modelo representa un grafo RDF, llamado así porque contiene una colección de nodos RDF, unidos entre sí por relaciones marcadas. En Jena, el recurso de interfaz Java representa tanto a los recursos ordinarios y los nodos de URI. ARQ Jena, mientras tanto, es un motor de búsqueda de Jena que admita el lenguaje SPARQL RDF de consulta. Permite: búsqueda de texto libre a través de Lucene; actualización, acceso y la extensión del álgebra de SPARQL; apoyo a las funciones de filtro personalizados; funciones de propiedad para un tratamiento personalizado de relaciones semánticas; y apoyo al cliente para el acceso remoto a cualquier SPARQL Endpoint.

Apache Lucene[27]

Lucene es el principal motor de búsqueda de código abierto y se utiliza en muchas empresas, proyectos y productos, originalmente implementada en Java. Es útil para cualquier aplicación que requiera la indización y las búsquedas a texto completo. En esencia, el índice se compone de los documentos que

[25] https://www.elastic.co/
[26] https://jena.apache.org/documentation/query/index.html
[27] https://lucene.apache.org/

se componen de campos. Las consultas de Lucene tienen que pasar a través de los mismos analizadores que se utilizaron durante la indización, de lo contrario términos idénticos podrían no coincidir. Se puede usar Lucene en Grails, como es el caso de los *plugins* para la integración de Solr y Elasticsearch con Grails.

De las herramientas estudiadas destaca entre académicos e investigadores como tendencia, el uso de Elasticsearch y Apache Solr en la realización de iniciativas semejantes a los objetivos de esta investigación. La Tabla 3 refleja una comparativa de ambas herramientas atendiendo a varios tópicos representativos.

Tabla 3 Comparativa entre las herramientas de indización examinadas.

Aspectos a comparar	**Elasticsearch**	**Apache Solr**
Modelo de BD	Motor de búsqueda (basado en Apache Lucene)	Motor de búsqueda (basado en Apache Lucene)
Licencia	Código abierto	Código abierto
Lenguaje de implementación	Java	Java
APIs	Java API RESTful HTTP/JSON API	Java API RESTful HTTP/JSON API, SolrJ
Lenguajes soportados	.Net,Erlang, Go, Groovy, Java, JavaScript, Lua, Perl, PHP, Python, Ruby, Scala, XML/JSON	.Net, Erlang, Java, JavaScript, XML/JSON, Perl, PHP, Python, Ruby, Scala
Sistema operativo	Todos con JVM	Todos con JVM
Lugar (DBMS/Motor de búsqueda)	11/1	14/2
Esquema de datos	Definiciones de tipos flexibles. Una vez que se define un tipo, es persistente	Campos dinámicos permiten adicionar otros campos sobre la marcha
Concurrencia	Si	Si
Durabilidad	Si	Si
Técnica de indización	Inverted Index	Inverted Index

La Figura 3 refleja los índices de uso entre los servidores de indización mejor posicionados y empleados por la comunidad científica, según publica DB-Engines. Cabe notar que en el presente escenario Elasticsearch ha ido en ascenso superando a Apache Solr, tal como puede apreciarse.

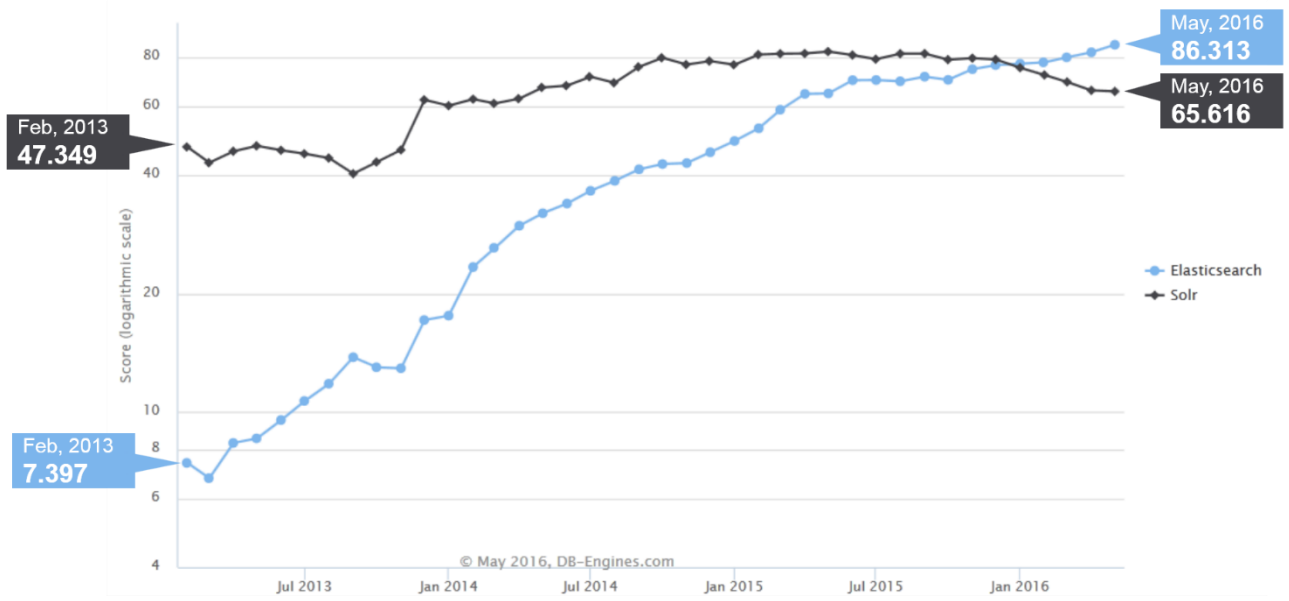


Figura 3 Comparativa de Elasticsearch y Apache Solr (Fuente: DB-Engines[28]).

Una vez analizadas las herramientas se propone el uso de Elasticsearch. Esta decisión está condicionada por una restricción del dominio del problema, ya que su integración con el marco de trabajo adoptado por el proyecto de investigación ocurre de manera natural mediante la Java API.

1.4.3.3 Patrones de interacción.

La disciplina de la *Interacción Persona-Ordenador* (IPO, por sus siglas en español) es el área de conocimiento científico encargada de estudiar el fenómeno de uso de productos interactivos, con el objetivo de definir técnicas y metodologías que aseguren empíricamente que estos productos cumplen con los niveles de usabilidad requeridos; es decir, que puedan ser usados de forma efectiva, eficiente, segura y satisfactoria.

Entre los patrones de interacción presentes en el estudio de (Rizo y García 2013) destacan:

Navegación facetada

El sistema de clasificación facetada permite representar una materia o tema combinando los conceptos que representan diferentes aspectos (facetas) de la misma. La búsqueda facetada se basa en el principio de que cualquier materia compleja se puede dividir en conceptos simples. Este principio se aplica a cada uno de los dominios cubiertos por el sistema (clases principales) que de esta forma quedan divididos en grupos de conceptos simples que comparten un atributo o propiedad común. Cada

[28] http://db-engines.com/en/system/Elasticsearch%3BSolr

uno de estos grupos se denomina faceta. Los conceptos agrupados en cada faceta se organizan de forma jerárquica, reflejando relaciones paradigmáticas (todo/parte). Dentro de la comunidad académica, la búsqueda facetada ha despertado el interés principalmente entre bibliotecas y ciencias de la información, los investigadores, y en cierta medida entre los investigadores de informática especializados en la recuperación de información.

Detalles sobre la demanda

La manera en que se brinda la información de un recurso al usuario es uno de los aspectos a considerar en la concepción de iniciativas que aprovechan las tecnologías de la web semántica. Una vez indicados los elementos sobre los cuales filtrar la búsqueda corresponde mostrar la información detallada de los mismos, que dicho de otra forma no es más que listar el conjunto de propiedades y sus respectivos valores.

Migajas de pan

Las migajas de pan (del inglés: *breadcrumbs*) son una técnica de navegación usada en muchas interfaces gráficas de usuario, además de páginas web. Aunque su diseño puede adoptar infinidad de variantes, en términos generales consiste en una línea de texto en la que se indica el recorrido seguido y la forma de regresar. Permite que el usuario conozca la ruta de su ubicación en directorios y subdirectorios, y navegue a través de ella (Nielsen 2007).

Auto-completamiento

Es común encontrar casos en los que los usuarios deban introducir valores en un cuadro de texto. En la mayoría de casos, utilizar una lista desplegable que muestre todos los valores resulta poco viable, ya que pueden existir miles de posibles valores. Por otra parte, un cuadro de texto simple resulta de poca utilidad para el usuario. La solución consiste en combinar un cuadro de texto y una lista desplegable mediante AJAX (del inglés: *Asynchronous JavaScript And XML*).

Al usuario se le presenta un cuadro de texto simple en el que puede introducir la información. A medida que el usuario escribe en el cuadro de texto, la aplicación solicita al servidor aquellos términos que estén relacionados con lo escrito por el usuario. Cuando la aplicación recibe la respuesta del servidor, la muestra al usuario a modo de ayuda para autocompletar la información.

1.4.3.4 Componentes de la Arquitectura de Información.

El término *Arquitectura de la Información* (AI, por sus siglas en español) fue acuñado por Richard Saul Wurman en 1976. Es un concepto utilizado en su forma más amplia para expresar el diseño, organización y distribución de los sistemas informáticos (Gonzáles-Cam 2003). La AI es un término difícil de definir, y para cada persona puede tener una acepción particular. Este término es usado para describir la planificación de la experiencia del usuario frente a un sitio web. La combinación de los elementos de la AI y su impacto en el diseño gráfico, permiten organizar la información de forma que el usuario pueda obtener adecuadamente y en forma rápida los datos dentro de su sitio web.

En la investigación de (Rizo y García 2013) son listados algunos de los componentes de la AI siguientes:

Facetas

Las facetas corresponden a propiedades de los elementos de información. A menudo se derivan de análisis del texto de un elemento utilizando técnicas de extracción de entidad o de pre-existentes campos en una base de datos, tales como autor, descriptor, el idioma y el formato.

Control de pestañas

El uso del control de pestañas (del inglés: *tab control*) resulta conveniente cuando la cantidad de información para ser mostrada es numerosa y mediante su empleo se logra una mejor organización en varias pestañas sin recargar de manera excesiva el área donde dicha información es mostrada. Los controles de pestaña son contenedores de una colección de objetos, de modo que cuando se elige una pestaña esta se activa pudiéndose realizar cambios en los objetos que ésta contiene y quedando el resto inhabilitada.

Migajas de pan

Las migajas de pan resultan útiles para ubicar al usuario en el espacio de búsqueda sobre el que se encuentra, de modo que pueda ver el curso de su búsqueda y regresar de manera más fácil y directa a los pasos anteriores.

Cajas de texto

Una caja de texto, campo de texto o caja de entrada de texto es un elemento común de una interfaz gráfica de usuario. El propósito de la caja de texto es permitir al usuario la entrada de información textual para ser usada por el programa. Es recomendable usar una caja de texto de una sola línea cuando solo una línea de entrada es requerida, y una caja de texto multilínea solo si más de una línea

de entrada puede ser requerida. Una caja de texto típica es un rectángulo de cualquier tamaño con una línea vertical que parpadea (conocida como *caret*), indicando la región actual del texto que se está editando.

1.5. Conclusiones parciales

Luego de ser consultada y estudiada la literatura especializada, y haberse confeccionado el marco teórico y estado del arte de la investigación, se concluye lo siguiente:

1. La revisión bibliográfica evidenció que la publicación de metadatos bibliográficos como datos enlazados, es un área de investigación en constante desarrollo, principalmente en el contexto de las bibliotecas digitales. Sin embargo, no se encontró en la literatura consultada abundantes investigaciones que demuestren su aplicación óptima para su consumo a partir de un triplestore.

2. Las técnicas de indización tratadas difieren en cuanto a características y la forma de realizar la indización, sin embargo, los índices invertidos destacan como la técnica empleada por las herramientas de indización analizadas.

3. El estudio de las herramientas mencionadas en la bibliografía consultada y el enfoque utilizado por cada una de ellas, evidenció que la herramienta Elasticsearch posee características superiores a otras similares.

CAPÍTULO 2. DESCRIPCIÓN DE LA PROPUESTA

2.1. Introducción

En este capítulo se presenta un componente de software que integra un servidor de indización con tripletas RDF existentes en un triplestore (epígrafe 2.2). Como parte de la propuesta ha sido implementada una plataforma informática utilizando herramientas y tecnologías actuales del desarrollo de software (epígrafe 2.4), detallándose los artefactos generados por la metodología de desarrollo adoptada (AUP variación para la UCI).

Se aborda acerca del modelo de datos de la propuesta (epígrafe 2.5), los estándares de código empleados (epígrafe 2.6), las técnicas de captura y validación de requisitos, como también el levantamiento de los requisitos funcionales (historias de usuario y sus estimaciones de esfuerzo) y no funcionales (epígrafe 2.7). Las tareas, patrones de interacción y componentes de la arquitectura de información (epígrafe 2.3), la arquitectura propuesta del componente (epígrafe 2.8) y la planificación de pruebas (epígrafe 2.9) son apartados tratados de igual modo en este acápite. Se concluye resumiendo los resultados obtenidos durante el diseño e implementación de la propuesta de solución (epígrafe 2.10).

2.2. Descripción general de la propuesta

El proyecto “Extracción, publicación y consumo de metadatos bibliográficos como datos enlazados” tiene como objetivo construir una Biblioteca Digital Semántica basada en datos enlazados. Este es el primer proyecto de su tipo en Cuba por lo que no se cuenta con colecciones de artículos en formato PDF. El proyecto cuenta con cuatro actividades fundamentales, que según (Delgado 2015) quedarían definidas de la siguiente manera: (1) *Extracción de datos*, (2) *Pre-procesamiento de datos*, (3) *Publicación de datos* y finalmente (4) *Consumo de datos*. Ver Figura 4.

Lo primero es extraer y almacenar los metadatos provenientes de fuentes de datos heterogéneas. En esta etapa, es necesario analizar varios aspectos de cada fuente de datos, tales como: la interoperabilidad de datos, la calidad de los datos, los esquemas de datos, la frecuencia de actualización y la sostenibilidad en el tiempo. El resultado para esa actividad es una base de datos relacional intermedia con los metadatos extraídos. En un segundo momento son analizados los metadatos provenientes de la tarea anterior, puesto que la calidad de los datos de la biblioteca es un punto crucial que afecta significativamente la visibilidad y el descubrimiento de los recursos descritos. El propósito que se tiene es limpiar y normalizar algunos campos de metadatos mejorando considerablemente su calidad. Esta actividad incluye la transformación de datos, tales como fechas,

volúmenes y números de revistas; así mismo, la detección de registros duplicados y la desambiguación de autores y afiliaciones. Consecuentemente se obtiene la base de datos con los metadatos limpiados y normalizados. Como tercera instancia es importante determinar la(s) ontología(s) a utilizar para el modelado de datos de la biblioteca. La recomendación más importante en este contexto es reutilizar lo más posible los vocabularios disponibles. Como salida a este proceso se obtiene un modelo ontológico. Con el modelo ontológico y el esquema de base de datos, lo próximo es la realización de tres importantes tareas: (1) la transformación, (2) el enlazado y (3) la publicación de dichos datos. Como consecuencia de acometer estas tareas se generan grafos RDF que son almacenados en un almacén de tripletas RDF. Finalmente, una vez los datos listos para su consumo, es posible obtener mejores resultados mediante búsquedas textuales y facetadas, así como de manera más rápida mediante el empleo de índices para las consultas realizadas por los usuarios.

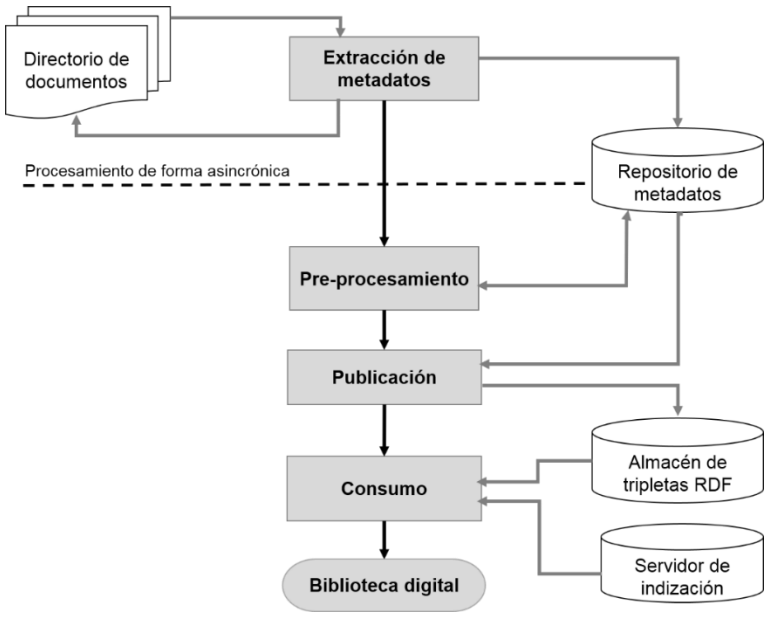


Figura 4 Interacción de los componentes principales del proyecto "Extracción, publicación y consumo de metadatos bibliográficos como datos enlazados". (Fuente: elaboración propia).

La propuesta de solución implementa el componente **sdl-index** con el propósito de llevar a cabo tres actividades puntuales, incluidas en la etapa: *consumo de datos*, las cuales se indican seguidamente:

1. **Obtener el grafo RDF procedente del SPARQL Endpoint**.

   Esta tarea tiene por objetivo incorporar al motor de búsqueda Elasticsearch por primera vez, los grafos RDF previamente generados y almacenados en el triplestore Apache Jena Fuseki (punto de acceso a consultas SPARQL). Estos grafos RDF se encuentran en formato JSON, que usa por defecto Elasticsearch.

2. **Sincronizar el grafo RDF proveniente del SPARQL Endpoint con el índice en el motor de búsqueda Elasticsearch**.

   El propósito de esta actividad es consultar el SPARQL Endpoint para verificar los últimos cambios e indizar sólo esos grafos RDF al índice del motor de búsqueda de Elasticsearch. Esto se puede especificar estableciendo el valor de “indexType” a “sync” en lugar de “full”, que es la configuración por defecto.

3. **Realizar búsquedas textuales y facetadas en SPARQL sobre el índice de Elasticsearch**.

   Esta actividad es el núcleo del componente, puesto que la realización de búsquedas textuales y facetadas como técnicas para acceder a la información organizada permite a los usuarios explorar una colección de información mediante la aplicación de varios filtros, lo cual tributa en la disminución del tiempo para consultar dicha información.

En una primera versión se pretende implementar un prototipo funcional que reutiliza y adapta componentes de la AI, con el fin de proveer una interfaz web para posibilitar que los usuarios que no poseen conocimientos técnicos sobre las tecnologías de la Web Semántica, pero sí de las tecnologías de la informática y las comunicaciones puedan consumir (navegar, consultar, etcétera) a través de la búsqueda textual y facetada los metadatos bibliográficos que han sido publicados a priori. Estos componentes de la AI están presentes en la mayoría de las páginas web, entiéndanse: las facetas de navegación, las migajas de pan, los menús de navegación, etcétera.

Los resultados obtenidos de la aplicación de este componente en la Biblioteca Digital Semántica se detallan en el Capítulo 3 de esta investigación.

2.3. Tareas, patrones de interacción y componentes de la AI.

Con el propósito de construir interfaces intuitivas, amigables y que satisfagan los criterios básicos de usabilidad para aplicaciones web, la información que se muestra al usuario ha sido estructurada en base a patrones de interacción comunes en la web y que pueden ser fácilmente extendidos al contexto de los datos enlazados. A continuación, se muestran las principales tareas utilizadas para el análisis del conjunto de datos, definidas por (Rizo y García 2013), junto al patrón interactivo que la satisface y el componente de la AI que implementa dicho patrón.

Filtrar los elementos de interés, y obviar aquellos que no son de interés para el usuario: aquí la propuesta es utilizar el patrón interactivo *navegación facetada*[29], *las facetas* son el componente de la AI que permite a los usuarios filtrar los elementos dentro del conjunto de datos, obviando los elementos que no son de su interés. Con esta estrategia se reduce el espacio de búsqueda por cada restricción aplicada hasta que se obtiene el elemento o elementos de interés. Ver Figura 5.

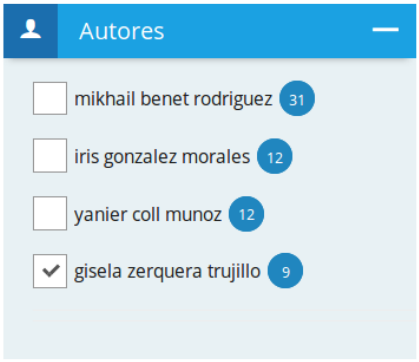


Figura 5 Vista de la faceta Autor. (Fuente: elaboración propia).

Mostrar los detalles de los recursos de interés: una vez que el usuario ha obtenido el(los) elemento(os) que requiere a través del patrón interactivo navegación facetada, es necesario que se muestre información detallada del elemento, esto en el contexto de los datos enlazados se reduce a listar el conjunto de propiedades y sus respectivos valores para cada uno de los elementos filtrados. Aquí se aplica el patrón interactivo *detalles sobre la demanda*[30] y el componente de la AI que lo implementa es el *control de pestañas*. Ver Figura 6.

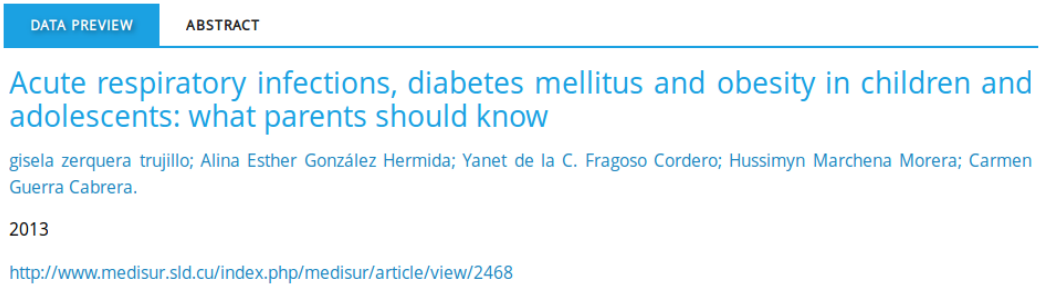


Figura 6 Listado de las contribuciones resultantes de la búsqueda. (Fuente: elaboración propia).

Contextualizar el espacio de navegación: cuando el usuario "navega" el conjunto de datos a través de las facetas o a través de otro tipo de búsqueda (como la búsqueda textual), puede ocurrir que este requiera orientarse y saber en qué parte de la búsqueda se encuentra y hacia dónde puede llegar desde

[29] http://www.welie.com/patterns/showPattern.php?patternID=faceted-navigation
[30] http://www.welie.com/patterns/showPattern.php?patternID=details-on-demand

su ubicación actual. Con este propósito se aplica el patrón interactivo *migajas de pan*[31], este ofrece un punto de referencia para el usuario durante toda la navegación y los procesos de búsqueda, así sabrá en cualquier momento en qué parte de la búsqueda se encuentra y dispondrá de enlaces directos a posiciones anteriores del camino recorrido. El componente de la AI que implementa este patrón interactivo recibe el propio nombre del patrón: *migajas de pan*. Ver Figura 7.

Figura 7 Ruta de navegación del usuario en la búsqueda. (Fuente: elaboración propia).

Sugerir opciones presentes en el conjunto de datos a medida que el usuario introduce el criterio de búsqueda: a los efectos de la búsqueda textual, resulta útil al usuario que a medida que se introduce algún criterio de búsqueda, se brinden un conjunto de opciones que coincidan con el criterio que se va introduciendo. Con esta finalidad se aplica el patrón interactivo *auto-completamiento*[32], el cual puede ser implementado a través de las *cajas de texto* como componente de la AI. Ver Figura 8.

Figura 8 Auto-completamiento de la búsqueda. (Fuente: elaboración propia).

Seguidamente, en la Tabla 4 se ofrece un resumen de las tareas, patrones de interacción y componentes de la AI explicados con anterioridad.

Tabla 4 Tarea, patrón de interacción y componente de la AI.

Tarea	**Patrón de interacción**	**Componente de la AI**
Filtrar los elementos de interés, y obviar aquellos que no son de interés para el usuario	Navegación facetada	Facetas
Mostrar los detalles de los recursos de interés	Detalles sobre la demanda	Control de pestañas
Contextualizar el espacio de navegación	Migajas de pan	Migajas de pan
Sugerir opciones presentes en el conjunto de datos a medida que el usuario introduce el criterio de búsqueda	Auto-completamiento	Cajas de texto

[31] http://www.welie.com/patterns/showPattern.php?patternID=crumbs
[32] http://www.welie.com/patterns/showPattern.php?patternID=autocomplete

Estos patrones de interacción y componentes de la AI se pueden ver implementados en la propuesta de solución tal y como se evidencia en el Anexo 1.

2.4. Metodología, herramientas y técnicas

En cada una de las tres actividades mencionadas a priori intervienen: metodología, herramientas y tecnologías y para su consecución.

2.4.1 Metodología de desarrollo.

El desarrollo de todo software debe estar guiado por una metodología de desarrollo. De esta depende, en gran medida, que el software tenga la calidad requerida. Existen dos grupos de metodologías: ágiles y tradicionales. No existe una metodología universal para cada tipo de proyecto. Se define una metodología según las características del equipo de desarrollo, el dominio de aplicación, el tipo de contrato, la complejidad y la envergadura del proyecto.

Dada la necesidad de desarrollar la propuesta de solución en un breve período de tiempo, garantizando además la flexibilidad necesaria en cuanto a la variación de los requisitos y el manejo de los riesgos técnicos, así como reducir la generación de documentos y artefactos, y no habiendo un contrato tradicional, siendo el cliente parte del equipo de desarrollo; se hace necesario optar por un enfoque ágil de desarrollo de software en lugar de un enfoque tradicional o pesado. Teniendo en cuenta lo anterior han sido evaluadas varias metodologías que siguen el enfoque ágil de desarrollo de software, tal es caso de *Scrum* (Sutherland 2015), *Agile Unified Process* (AUP, por sus siglas en inglés) (Edeki 2013), *Extreme Programming*[33] (XP, por sus siglas en inglés) (Kniberg 2015) y *AUP-UCI*. Finalmente, el colectivo de autores de la investigación adopta la variación de la metodología **AUP** para la UCI, específicamente en el escenario cuatro; puesto que se ajusta para proyectos que no modelan un negocio sino modelan el sistema con las *Historias de Usuario* (HU, por sus siglas en español), siendo éste el caso que ocupa.

2.4.2 Lenguajes de consulta para RDF.

Un lenguaje de consultas RDF permite recuperar y manejar datos almacenados en formato RDF. Para el desarrollo de la propuesta de solución se han analizado varios de estos lenguajes: *RQL* (Karvounarakis et al. 2013), *SeRQL (Mateus, Ruiz y Plaza 2015)* y *RDQL* (Khan y Malik 2012). En su conjunto estos lenguajes presentan inconvenientes tales como: su semántica no es totalmente compatible con la de RDF, son vulnerables a inyecciones de código y no hacen una distinción estricta

[33] http://www.extremeprogramming.org , www.xprogramming.com

entre las consultas y las reglas (Haase et al. 2004). Con la aplicación de los datos enlazados se ha extendido el uso de un nuevo lenguaje de consulta RDF recomendado por la W3C: **SPARQL**.

SPARQL (del inglés: *SPARQL Protocol and RDF Query Language*) es un lenguaje de consultas estructuradas sobre uno o múltiples grafos RDF, (Harris, Seaborne y Prud'hommeaux 2013) lo definen como un conjunto de especificaciones que proporcionan lenguajes y protocolos para consultar y manipular el contenido gráfico de RDF en la Web o en triplestore.

La sintaxis de SPARQL es similar a la del lenguaje SQL (del inglés: *Structured Query Language*) aunque orientado a tripletas RDF (de la forma Sujeto – Predicado – Objeto). Los resultados de las consultas SPARQL pueden ser conjuntos de tripletas RDF, grafos RDF, URIs de recursos o simplemente valores (cadenas de texto, números, etcétera). La especificación actual de este lenguaje es SPARQL v.1.1 (Harris, Seaborne y Prud'hommeaux 2013).

Como declaran (Harris, Seaborne y Prud'hommeaux 2013), el lenguaje SPARQL incluye *Internationalized Resource Identifier* (IRI, por sus siglas en inglés), un subconjunto de Referencias URI RDF que omiten los espacios. Debe tenerse en cuenta que en las consultas SPARQL todas las referencias a IRIs son absolutas; pueden o no incluir un identificador de fragmento. Las referencias IRI incluyen a las referencias URI y URL (del inglés: *Uniform Resource Locator*). En la sintaxis de SPARQL, las formas abreviadas (referencias IRIs relativas y nombres con prefijos) se resuelven para producir IRIs absolutas. Un SPARQL Endpoint[34], o punto de acceso a base de datos, es la interfaz que permite realizar consultas a una tripleta.

2.4.3 Almacén de tripletas RDF.

Hay una gran variedad de triplestores y no es trivial encontrar el más adecuado para las necesidades exactas en el cumplimiento de un proyecto real. Cuando se trata el término triplestore, entiéndase una herramienta que tiene alguna forma de almacenamiento persistente de datos RDF y le permite ejecutar consultas SPARQL contra esos datos. El apoyo SPARQL o bien puede ser incorporado como parte de la principal herramienta, o un complemento instalado por separado.

Para la selección del almacén de tripletas RDF de la propuesta de solución se han analizado tres soluciones de este tipo: *Apache Jena Fuseki*[35], *Sesame2*[36], *Virtuoso*[37]. De ellas ha sido adoptada **Apache Jena Fuseki** (v.2.0) que es un servidor de SPARQL y proporciona seguridad (usando Apache

[34] http://semanticweb.org/wiki/SPARQL_endpoint
[35] https://jena.apache.org/documentation/fuseki2/index.html
[36] https://www.w3.org/2001/sw/wiki/Sesame
[37] http://virtuoso.openlinksw.com

Shiro) además de contar con una interfaz de usuario para el control y la administración del servidor. Fuseki está estrechamente integrado con TDB (un componente de Jena para el almacenamiento y consulta de RDF; compatible con toda la gama de API Jena) para proporcionar una sólida capa de almacenamiento persistente transaccional. Se puede utilizar para proporcionar el motor de protocolo para otros sistemas de consulta y almacenamiento RDF.

2.4.4 Entorno de trabajo.

Un *Integrated Development Environment* (IDE, por sus siglas en inglés), en español: Entorno de Desarrollo Integrado, es un programa compuesto por una serie de herramientas utilizadas por programadores para desarrollar aplicaciones (Mahmood y Reddy 2014). Para acometer la propuesta de solución se emplea como IDE **IntelliJ IDEA**[38] (v.14.0.1), desarrollado por JetBrains. Se decide el uso de IntelliJ IDEA Community Edition 14.0.1 puesto que analiza constantemente su código, en busca de conexiones entre los símbolos a través de todos los archivos y los idiomas del proyecto, proporcionando ayuda para la codificación en profundidad, una navegación rápida, el análisis de errores con soluciones rápidas a mano, además de refactorizaciones inteligentes que aseguren la completa coherencia de los cambios.

2.4.5 Lenguaje de programación.

Apache Groovy[39] (v.2.5) es un lenguaje poderoso, opcionalmente mecanografiado y dinámico, con la tipificación estática y las capacidades de compilación estática, para la plataforma Java orientado a mejorar la productividad del desarrollador gracias a una sintaxis concisa, familiar y fácil de aprender. Se integra sin problemas con cualquier programa de Java, e inmediatamente se entrega a la aplicación de características de gran alcance, incluyendo las capacidades de *scripting*, de autoría de lenguaje específico de dominio, tiempo de ejecución y tiempo de compilación meta-programación y la programación funcional. Por las amplias posibilidades que ofrece se elige como lenguaje de programación del componente a lograr.

2.4.6 Marco de trabajo.

Grails[40] (v.2.5.3) es un marco de trabajo (del inglés: *framework*) de desarrollo web de gran alcance, para la plataforma Java destinada a multiplicar la productividad de los desarrolladores gracias a un convenio sobre configuración, los parámetros por defecto y las API. Se integra sin problemas con la

[38] https://www.jetbrains.com/idea/
[39] http://www.groovy-lang.org/
[40] https://grails.org/

Java Virtual Machine (JVM, por sus siglas en inglés), en español: Máquina Virtual de Java, que le permite ser productivos de inmediato mientras que proporciona características de gran alcance, incluyendo *Object Relational Mapping* (ORM, por sus siglas en inglés), en español: Mapeo de Objeto Realacional, idioma específico de dominio, tiempo de ejecución y meta-programación en tiempo de compilación y programación asíncrona.

2.5. Modelo de datos

El modelo de datos de la propuesta de solución se compone de recursos, propiedades y valores representados en forma de grafo RDF, como ilustra la Figura 9.

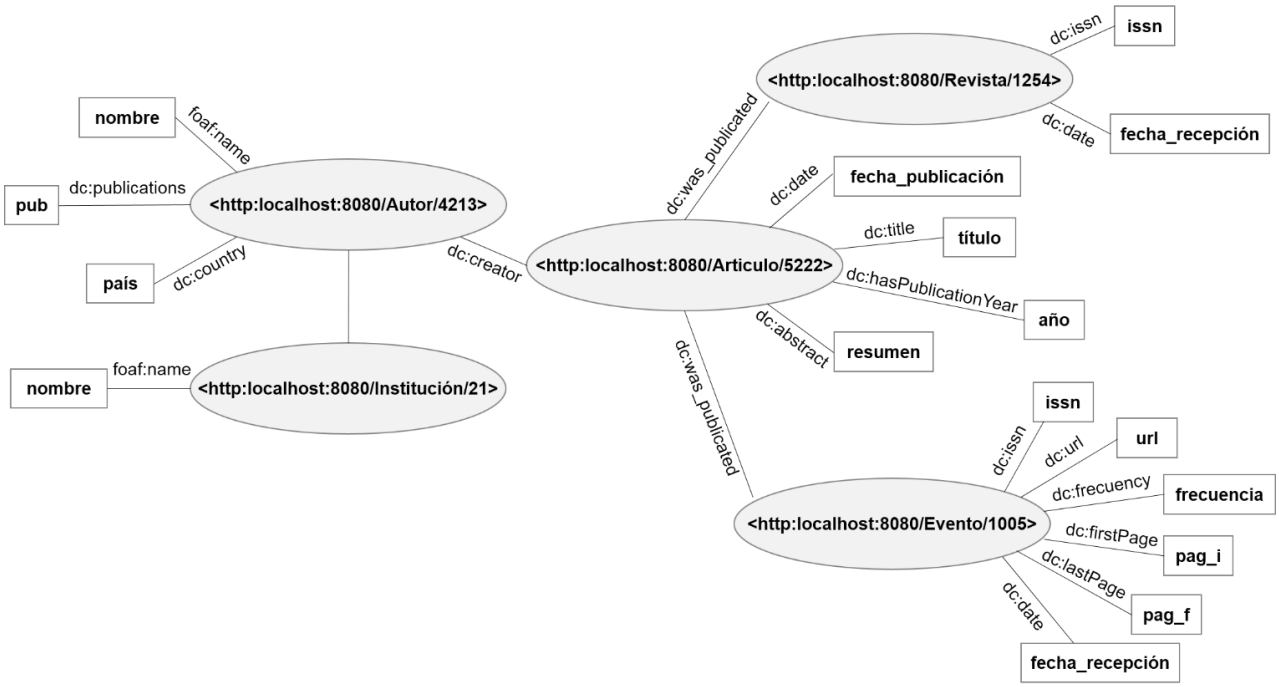


Figura 9 Modelo de datos del proyecto "Extracción, publicación y consumo de metadatos bibliográficos como datos enlazados" basado en grafos RDF. (Fuente: elaboración propia).

2.6. Estándares de código.

Un estándar de código se basa en la estructura y apariencia física de un programa con el fin de facilitar la lectura, comprensión, mantenimiento del código, reutilización a lo largo del proceso de desarrollo de un software y no en la lógica del programa. Un estándar de programación no solo busca definir la nomenclatura de las variables, objetos, métodos y funciones, sino que también tiene que ver con el orden y legibilidad del código escrito (Guerrouj 2013). Partiendo de lo dicho anteriormente, se definen tres partes principales dentro de un estándar de programación: (1) nomenclatura de las clases, (2)

nomenclatura según el tipo de clases y (3) nomenclatura de las funcionalidades y atributos. Una explicación detallada de cada una de las partes puede encontrase en el Anexo 2.

2.7. Requisitos.

La definición de requisito en la literatura científica cuenta con varias acepciones. La Real Academia Española[41] lo define como: circunstancia o condición necesaria para algo. Por otra parte, IEEE en (ISO 2011) enuncia el concepto de la siguiente manera: declaración que se traduce o expresa una necesidad y sus limitaciones y las condiciones correspondientes. A continuación de listan las técnicas empleadas para la captura de requisitos, los requisitos funcionales y no funcionales identificados y las técnicas para su validación.

2.6.1 Técnicas de captura de requisitos.

Las técnicas de captura de requisitos trabajadas en la presente investigación son:

Tormenta de ideas

Las tormentas de ideas (del inglés: *brainstorming*) son una técnica de reuniones en grupo sencilla y fácil de aplicar, cuyo objetivo es que los participantes muestren sus ideas en un ambiente libre de críticas o juicios. Consiste en la mera acumulación de ideas y/o información sin evaluar las mismas. Las tormentas de ideas suelen ofrecer una visión general de las necesidades del sistema, pero normalmente no sirve para obtener detalles concretos del mismo, por lo que suele aplicarse en los primeros encuentros.

Reunión con el cliente

Resulta como una técnica muy aceptada dentro de la ingeniería de requisitos y su uso está ampliamente extendido. A través de esta técnica el equipo de trabajo se acerca al problema de una forma natural y le permite comprender los objetivos de la solución buscada.

Prototipado

Algunas propuestas se basan en obtener de la definición de requisitos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema con el usuario. Un prototipo en software es un modelo del comportamiento del

[41] http://dle.rae.es/?id=W6xh4wt

sistema que puede ser usado para entenderlo completamente o ciertos aspectos de él y así clarificar los requisitos.

2.6.2 Requisitos funcionales.

Las historias de usuario son utilizadas en las metodologías de desarrollo ágiles puesto que son una forma rápida para la especificación y administración de requisitos, sin necesidad de elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Las historias de usuario permiten responder rápidamente a los requisitos cambiantes y deben cumplir varias restricciones, entre ellas: ser independientes unas de otras, negociables, estimables, pequeñas y verificables, por mencionar algunas. Seguidamente se enumeran las historias de usuario definidas para las tres actividades del componente propuesto. Ver Tablas de la 5 a la 7.

Tabla 5 Historia de usuario del requisito: Obtener grafo RDF procedente del SPARQL Endpoint

<table>
<tr><td>Código: HU-01</td><td colspan="2">Nombre del requisito: Obtener grafo RDF procedente del SPARQL Endpoint.</td></tr>
<tr><td colspan="2">Programador(es):
Alejandro Jesús Mariño Molerio
Juan Carlos Moreira De Lara</td><td>Iteración asignada: 1</td></tr>
<tr><td colspan="2">Prioridad: alta</td><td>Tiempo estimado: 30 días</td></tr>
<tr><td colspan="2">Riesgo en desarrollo: alto</td><td>Tiempo real: 3 semanas</td></tr>
<tr><td colspan="3">Descripción: Esta tarea tiene por objetivo incorporar al motor de búsqueda Elasticsearch por primera vez, los grafos RDF previamente generados y almacenados en el triplestore Apache Jena Fuseki (punto de acceso a consultas SPARQL). Estos grafos RDF se encuentran en formato JSON, que usa por defecto Elasticsearch.</td></tr>
<tr><td colspan="3">Prototipo de interfaz</td></tr>
</table>

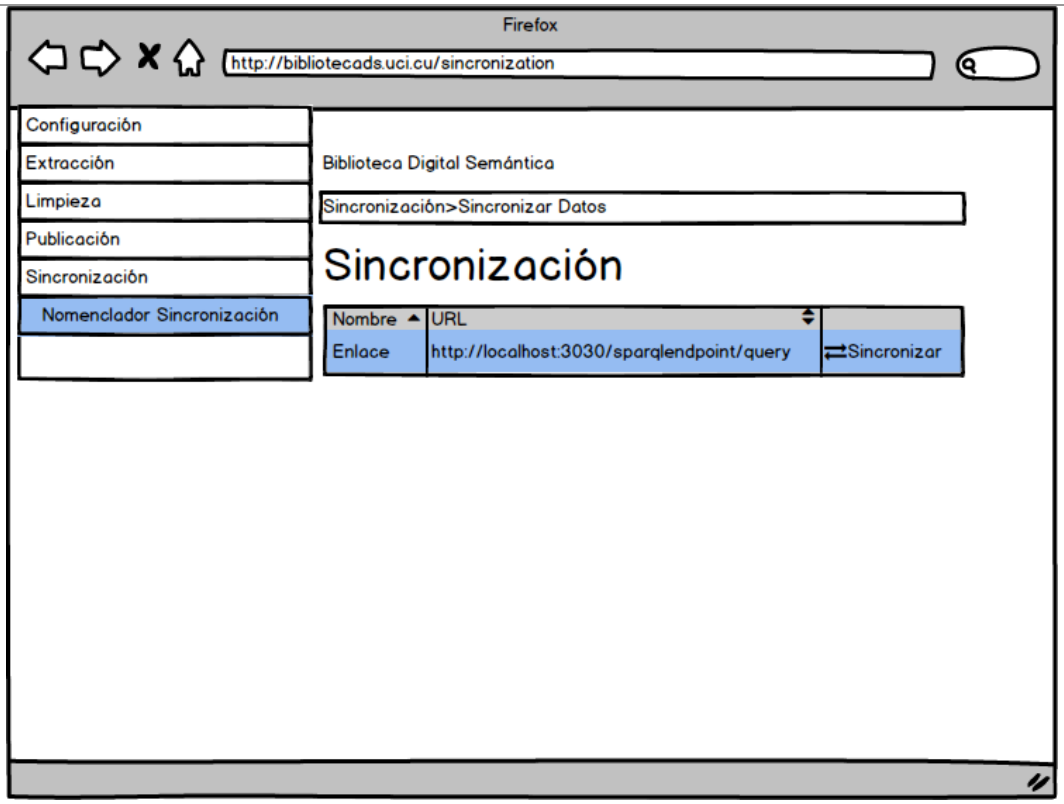


Tabla 6 Historia de usuario del requisito: Sincronizar grafo RDF proveniente del SPARQL Endpoint con el índice en el motor de búsqueda Elasticsearch

Código: HU-02	**Nombre del requisito:** Sincronizar grafo RDF proveniente del SPARQL Endpoint con el índice en el motor de búsqueda Elasticsearch.
Programador(es): Alejandro Jesús Mariño Molerio Juan Carlos Moreira De Lara	Iteración asignada: 1
Prioridad: alta	Tiempo estimado: 30 días
Riesgo en desarrollo: alto	Tiempo real: 3 semanas
Descripción: El propósito de esta actividad es consultar el SPARQL Endpoint para verificar los últimos cambios e indizar sólo esos grafos RDF al índice del motor de búsqueda de Elasticsearch. Esto se puede especificar estableciendo el valor de “indexType” a “sync” en lugar de “full”, que es la configuración por defecto.	
Prototipo de interfaz:	

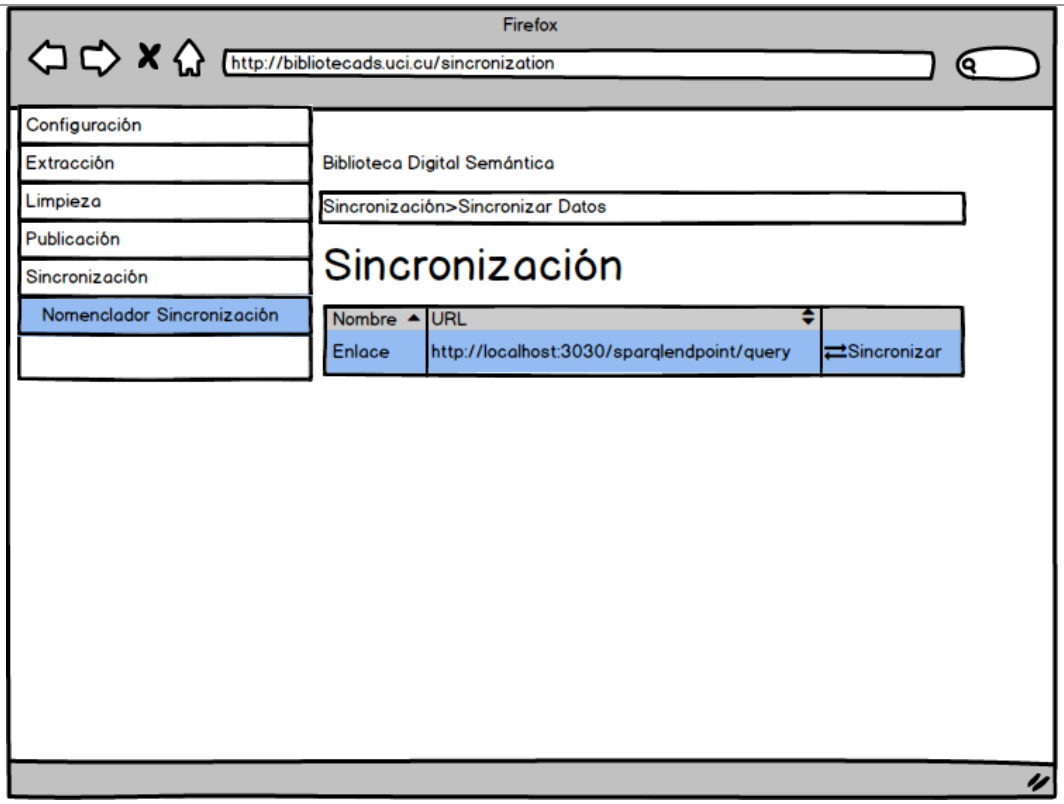
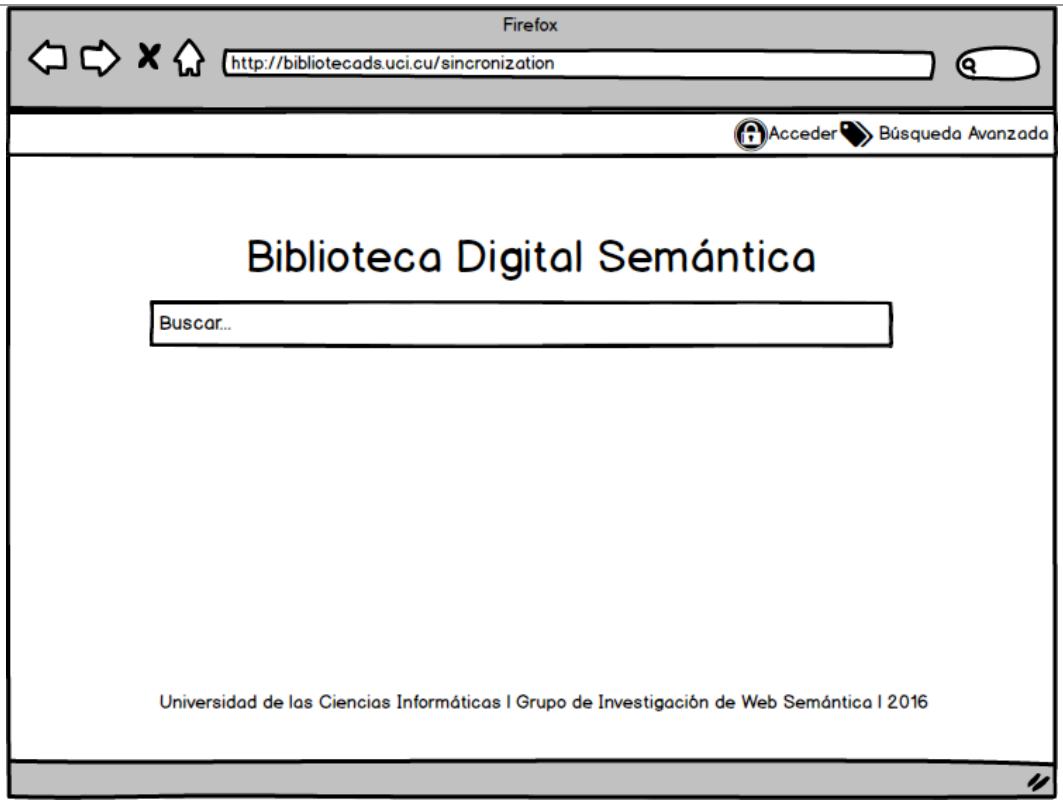
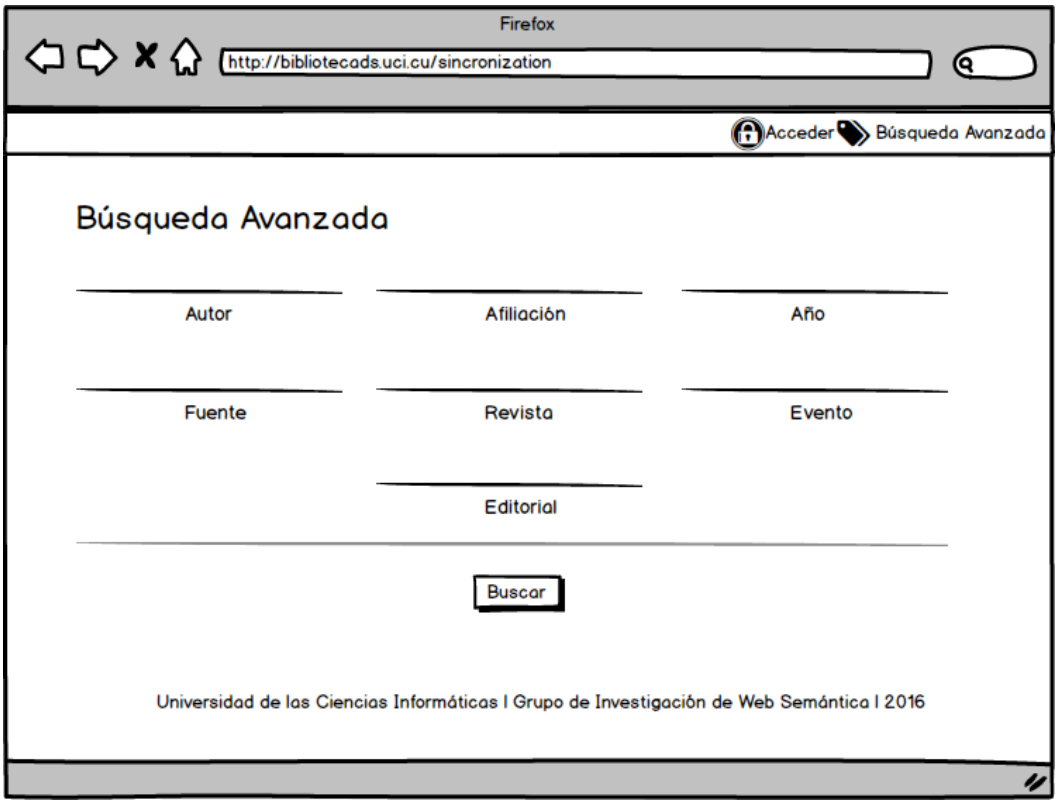
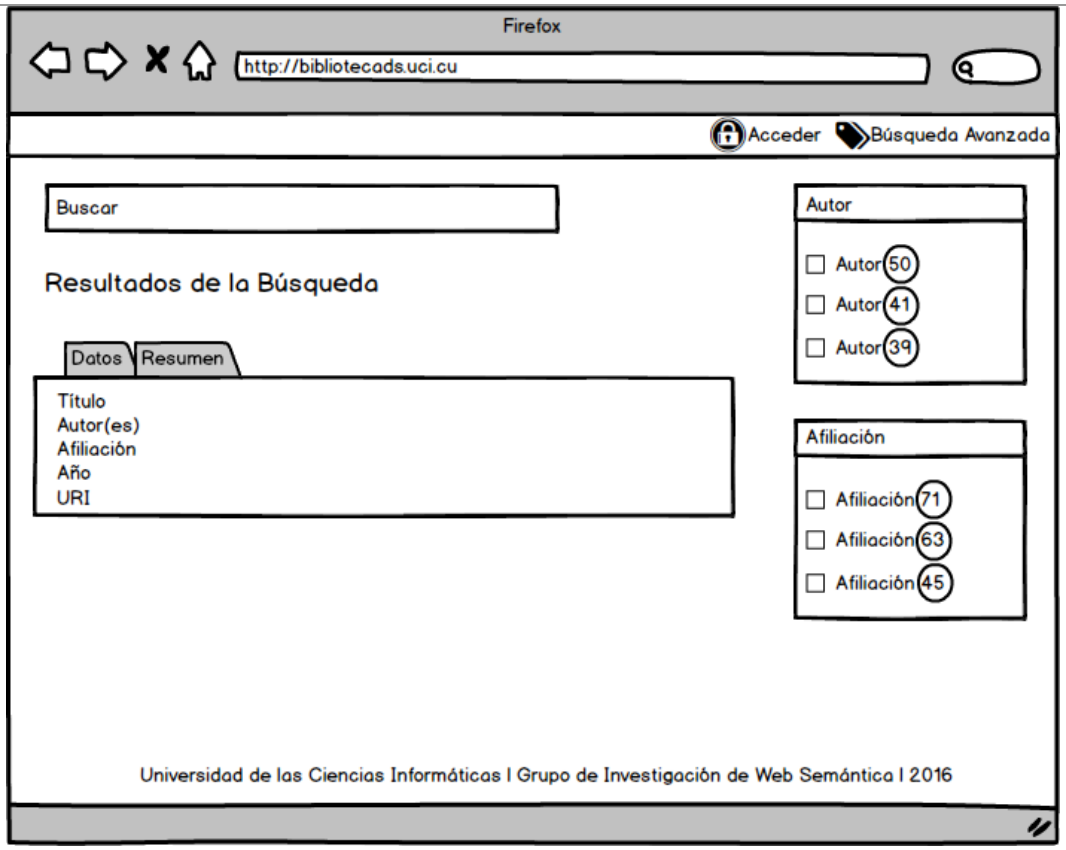


Tabla 7 Historia de usuario del requisito: Realizar búsquedas textual y facetada en SPARQL sobre el índice de Elasticsearch

Código: HU-03	**Nombre del requisito:** Realizar búsquedas textual y facetada en SPARQL sobre el índice de Elasticsearch.
Programador(es): Alejandro Jesús Mariño Molerio Juan Carlos Moreira De Lara	Iteración asignada: 1
Prioridad: alta	Tiempo estimado: 42 días
Riesgo en desarrollo: alto	Tiempo real: 4 semanas
Descripción: Esta actividad es el núcleo del componente, puesto que la realización de búsquedas textuales y facetadas como técnicas para acceder a la información organizada permite a los usuarios explorar una colección de información mediante la aplicación de varios filtros, lo cual tributa en la disminución del tiempo para consultar dicha información.	
Prototipo de interfaz:	


Firefox
http://bibliotecads.uci.cu/sincronization
Acceder
Búsqueda Avanzada
Biblioteca Digital Semántica
Buscar...
Universidad de las Ciencias Informáticas I Grupo de Investigación de Web Semántica I 2016


Firefox
http://bibliotecads.uci.cu/sincronization
Acceder
Búsqueda Avanzada
Búsqueda Avanzada
Autor
Afiliación
Año
Fuente
Revista
Evento
Editorial
Buscar
Universidad de las Ciencias Informáticas I Grupo de Investigación de Web Semántica I 2016



Estimación de esfuerzo por historias de usuario.

Dependiendo de la prioridad asignada por el cliente a cada historia de usuario y atendiendo a la complejidad y riesgo determinado por el programador, se define la estimación de cada una de las historias de usuario identificadas. La Tabla 8 muestra los resultados de la estimación realizada. La unidad de estimación es el punto, y un punto equivale a una semana ideal de programación.

Tabla 8 Estimación de esfuerzo por historias de usuario

Historias de usuario	**Puntos de estimación**
Obtener grafo RDF procedente del SPARQL Endpoint	3
Sincronizar grafo RDF proveniente del SPARQL Endpoint con el índice en el motor de búsqueda Elasticsearch	3
Realizar búsquedas textual y facetada en SPARQL sobre el índice de Elasticsearch	4

2.6.3 Requisitos no funcionales.

La ingeniería de requisitos comprende todas las tareas relacionadas con la determinación de las necesidades o de las condiciones a satisfacer para un software nuevo o modificado, tomando en cuenta los diversos requisitos. El propósito de la ingeniería de requisitos es hacer que los mismos alcancen un estado óptimo antes de alcanzar la fase de diseño en el proyecto. Los buenos requisitos deben ser medibles, comprobables, sin ambigüedades. A continuación, son especificados los trece requisitos no funcionales con que debe cumplir el componente que se propone, agrupados en las siguientes categorías: software, hardware, diseño y usabilidad. Ver Tabla 9.

Tabla 9 Relación de los requisitos no funcionales identificados.

Software	**Hardware**
El servidor donde se despliegue la aplicación debe tener previamente instaladas las siguientes herramientas: Apache Jena Fuseki, como almacén de tripletas RDF en su versión 2.0 o superior. El servidor de indización Elasticsearch en su versión 2.3.1. La máquina virtual de Java (*JVM*, por sus siglas en inglés) versión 7.0. La versión 7.9 de Tomcat como servidor de aplicaciones.	El servidor donde se despliegue la aplicación debe tener las siguientes propiedades: Microprocesador: Intel Core i5 de cuarta generación a 2.33 GHz. Memoria RAM: 4 GB DDR3 SDRAM. Capacidad de almacenamiento: 1 terabyte (TB). Tarjeta de red: Fast Ethernet a 100 Mbps (megabits por segundo).
Diseño	**Usabilidad**
La apariencia de las vistas debe ser de color negro, azul y blanco. El color de los botones debe ser azul para propósitos generales y verde para mostrar resultados de búsqueda.	La propuesta de solución debe poseer un diseño web adaptativo (del inglés: *responsive web design*) a fin de garantizar la adecuada visualización en múltiples computadoras personales, *tablets* y teléfonos inteligentes (del inglés: *smartphones*). El sistema debe poseer interfaces gráficas intuitivas.

El tipo de letra a utilizar en las interfaces y los mensajes de la aplicación es Sans-Serif, mientras serán empleados como colores el blanco, azul y negro indistintamente.	

2.6.4 Técnicas de validación de requisitos

Como técnicas de validación de requisitos adoptadas en este trabajo se tienen:

Casos de prueba

Los casos de prueba son un conjunto de condiciones o variables bajo las cuáles un analista determinará si una aplicación o sistema software es parcial o completamente satisfactoria. Con el propósito de comprobar que todos los requisitos de una aplicación son revisados, debe haber al menos un caso de prueba para cada requisito a menos que un requisito tenga requisitos secundarios. En ese caso, cada requisito secundario deberá tener por lo menos un caso de prueba. Lo que caracteriza un escrito formal de caso de prueba es que hay una *entrada conocida* y una *salida esperada*, los cuales son formulados antes de que se ejecute la prueba. La *entrada conocida* debe probar una precondición y la *salida esperada* debe probar una postcondición.

Prototipado

El prototipado de interfaz de usuario es una técnica de representación aproximada de la interfaz de usuario de un sistema software que permite a clientes y usuarios entender más fácilmente la propuesta de los ingenieros de requisitos para resolver sus problemas de negocio. Los dos tipos principales de prototipos de interfaz de usuario son: (1) *desechables*: que se utilizan sólo para la validación de los requisitos y posteriormente se desechan, y (2) *evolutivos*: una vez utilizados para la validación de los requisitos, se mejora su calidad y se convierten progresivamente en el producto final.

2.8. Arquitectura del componente

El componente sdl-index como propuesta de solución sigue un estilo arquitectónico de *flujo de datos*, que enfatiza la reutilización y modificalidad; siendo apropiada para sistemas que implementan transformaciones de datos en pasos sucesivos. Se recomienda ser aplicada en escenarios donde se demande el uso de un índice para lograr mejores tiempos de respuesta a consultas formuladas por usuarios para el acceso a un recurso. La arquitectura utilizada es de *tuberías y filtros*, que consiste en

ir transformando un flujo de datos en un proceso comprendido por varias fases secuenciales, siendo la entrada de cada una la salida de la anterior. Una tubería (del inglés, *pipeline*) conecta componentes computacionales (filtros) a través de conectores (del inglés, *pipes*), de modo que las computaciones se ejecutan a la manera de un flujo. Los datos se transportan a través de las tuberías entre los filtros, transformando gradualmente las entradas en salidas. En la Figura 10 se muestra la propuesta para la indización de grafos RDF desde un triplestore. Se refleja el orden de las actividades que lo componen, los resultados que serán obtenidos en cada una y las tecnologías que intervienen en las mismas.

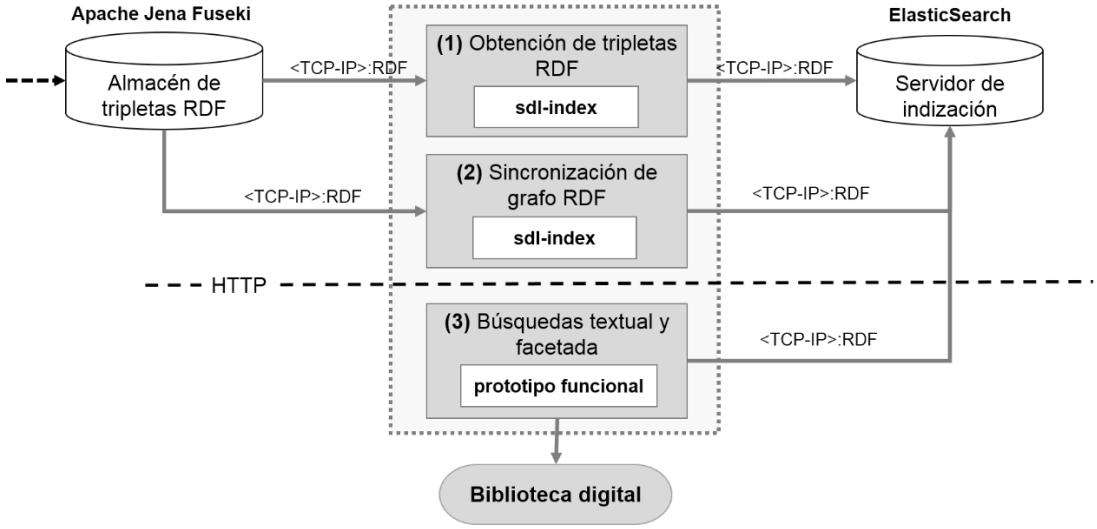


Figura 10 Arquitectura del componente sdl-index. (Fuente: elaboración propia).

La arquitectura presentada, contempla dos elementos fundamentales: (1) el triplestore Apache Jena Fuseki y (2) el motor de búsqueda Elasticsearch. La interacción entre ambos se debe a la necesidad de encontrar una alternativa que resuelva el inconveniente del tiempo de respuesta que tardan las consultas realizadas por los usuarios en pos de búsquedas más eficientes. Siendo así se precisa el empleo de un índice.

Apache Jena Fuseki almacena los grafos RDF generados a priori. Gracias a la Java API que provee Elasticsearch, es posible obtener dichos grafos RDF en el motor de búsqueda. La tarea de obtención de grafos RDF se acomete en primera instancia, ya que seguidamente basta sincronizar los grafos RDF del triplestore con los del índice en el motor de búsqueda. Este proceso es ventajoso en el sentido de que no se precisa el almacenamiento repetido de los grafos RDF del triplestore, sino solamente los últimos cambios efectuados. Las búsquedas textual y facetada benefician a centralizar la búsqueda de un determinado recurso. Desde el motor de búsqueda es permitido realizar consultas SPARQL al SPARQL Endpoint, lo cual se debe a que el formato JSON respeta la estructura del grafo RDF.

2.9. Planificación de pruebas

El proceso de pruebas de AUP constituye una de sus fortalezas, puesto que permite aumentar la calidad del sistema reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. En la variación de la metodología AUP para la UCI este proceso se desagrega en tres disciplinas: (1) pruebas internas, (2) pruebas de liberación y por último (3) pruebas de aceptación. En el caso específico del componente propuesto, solo se tratan las pruebas internas y las pruebas de aceptación, debido a que las pruebas de liberación son diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.

2.8.1 Pruebas internas

En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Es aplicable a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que ellos funcionen como se espera. El objetivo de las pruebas internas es el aislamiento de partes del código y la demostración de que estas partes no contienen errores.

Las pruebas de **caja blanca**, denominada a veces “prueba de caja de cristal”, a consideración de (Pressman 2005), es un método de diseño de casos de prueba que se basa en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo **casos de prueba** que ejerciten conjuntos específicos de condiciones y/o bucles. Esto genera gran cantidad de caminos posibles por lo que hay que dedicar esfuerzos a la determinación de las condiciones de prueba que se van a verificar.

Este tipo de prueba será aplicada a la estructura de control del diseño procedimental (código fuente), a través de la técnica del **camino básico**. La prueba del camino básico es una técnica de prueba de caja blanca propuesta por Tom McCabe, como expone (Pressman 2005). Esta técnica, permite obtener una medida de la complejidad lógica de un diseño y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Mediante su aplicación: se garantiza la ejecución por lo menos una vez de los caminos independientes de cada módulo, se ejercitan las decisiones lógicas y estructuras internas de datos para asegurar su validez, así como ejecución de los bucles.

La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se

construye el grafo de flujo asociado y se calcula su *complejidad ciclomática*. Los pasos que se siguen para aplicar esta técnica son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

2.8.2 Pruebas de aceptación

Estas pruebas las realiza el cliente. En este caso el cliente es el Grupo de Investigación de Web Semántica. Las pruebas de aceptación, no se realizan durante el desarrollo, pues no se podrían presentar al cliente; sino que se ejecutan sobre el producto terminado e integrado o bien una versión del producto o una iteración funcionad pactada previamente con el cliente. Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

La experiencia muestra que aún después del más cuidadoso proceso de pruebas por parte del desarrollador, quedan una serie de errores que aparecen cuando el cliente comienza a usarlo, por lo que las pruebas de aceptación tienen un mayor grado de importancia que las pruebas unitarias (Ruiz, Almanza y Pons 2011).

Las pruebas de aceptación, son pruebas funcionales, sobre el sistema, y buscan una cobertura de las historias de usuario. Durante una iteración la historia de usuario seleccionada en la planificación de iteraciones se convertirá en una prueba de aceptación. El cliente o usuario especifica los aspectos a probar cuando una historia de usuario ha sido correctamente implementada. Es responsabilidad del cliente verificar la corrección de las pruebas de aceptación y tomar decisiones acerca de las mismas (Pressman 2005).

Las pruebas de **caja negra**, también denominada “prueba de comportamiento”, se centran en los requisitos funcionales del software (Pressman 2005). Mediante las técnicas de prueba de caja negra se obtiene un conjunto de casos de prueba que intentan encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos externas, errores de rendimiento y errores de inicialización y de terminación.

La **partición equivalente** es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar **casos de prueba**. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. Una *clase de equivalencia* representa un conjunto de estados validos o no válidos para condiciones de entrada.

2.10. Conclusiones parciales

En este capítulo se propuso un componente para la integración de un servidor de indización con tripletas RDF existentes en un triplestore. Luego de presentada la propuesta, se concluye que:

1. El componente sdl-index como propuesta de solución para la indización de grafos RDF a partir de un triplestore, está basado en tres fases fundamentales. Se implementa un prototipo funcional con el fin de proveer una interfaz web para la búsqueda textual y facetada sobre los datos en el índice.

2. El empleo de índices como estructuras de datos, es una solución eficiente para disminuir el tiempo de respuesta a consultas formuladas por los usuarios, en el consumo de datos enlazados a partir de un triplestore.

3. La propuesta de solución evita indizar, luego de una primera vez, todos los grafos RDF al índice del motor de búsqueda y en su lugar, sólo indiza los últimos cambios realizados en el triplestore mediante la sincronización, por lo que aporta una solución viable a los problemas de escalabilidad y rendimiento.

CAPÍTULO 3. VALIDACIÓN DE LA PROPUESTA

3.1. Introducción

En el capítulo antecedente fueron definidos los tipos y técnicas de pruebas para su empleo posterior. Este capítulo, tiene como objetivo validar la propuesta de solución aplicando dichas pruebas (epígrafe 3.2). Se desarrolla un caso de estudio empleando datos reales provenientes de revistas cubanas (epígrafe 3.3). Adicionalmente, se realiza un pre-experimento para evaluar los tiempos de respuesta de consultas SPARQL realizadas por los usuarios a un grafo RDF en comparación con el uso de índices. Se describen en detalle los principales resultados obtenidos con el caso de estudio y el experimento desarrollado (epígrafe 3.4). Por último, se relacionan aspectos representativos resultantes de la validación de la propuesta de solución (epígrafe 3.5) y se emiten las consideraciones generales de esta sección (epígrafe 3.6).

Para una mejor comprensión del contenido de este capítulo se hace necesario definir el término experimento. Una definición simple pero acertada es: un estudio que involucra la manipulación intencional de una acción para analizar sus posibles efectos (Grau, Correa y Rojas 2004). Los experimentos se dividen en tres grupos: (1) pre-experimentos, (2) cuasi-experimentos y (3) experimentos puros.

Los *pre-experimentos* se distinguen por no poseer un grupo de control o patrón para realizar la comparación. La principal característica de los *cuasi-experimentos* es que la asignación de los participantes a los grupos no se hace de forma aleatoria ni por emparejamiento. Los *experimentos puros* difieren de los pre-experimentos y los cuasi-experimentos en el control sobre la situación experimental; en ellos se debe: manipular una o más variables independientes, medir el efecto de la variable independiente sobre la variable dependiente y controlar la validez interna de la situación experimental.

3.2. Pruebas de software

3.2.1 Prueba de caja blanca

Las pruebas de caja blanca, fueron aplicadas haciendo uso de la técnica del camino básico, como se mencionó en el capítulo anterior, con el objetivo de evaluar la complejidad lógica de un diseño procedimental y usar esta medida como guía para la definición de un conjunto básico de caminos de ejecución (Pressman 2005). Esta prueba permite garantizar que en los casos de prueba obtenidos a través del camino básico se ejecute cada sentencia del programa por lo menos una vez.

El uso de esta técnica es mostrado en el ejemplo siguiente. Se analizan y enumeran las sentencias de código del método **SparqlSincronization** contenidas en la clase **ConnectionsToEnd.groovy**. Este método chequea las actualizaciones ocurridas en el almacén de tripletas RDF para indizar al servidor de indización solamente esas ocurrencias en lugar del grafo íntegro. Ver la Figura 11.

```
public boolean SparqlSincronization(String uri) {
    url = uri;
    exec = QueryExecutionFactory.sparqlService(url, query);
    //if (testConection(uri)) {
        try {
            Settings settings = Settings.settingsBuilder()
                    .put("client.transport.ignore_cluster_name", true).build();
            client = TransportClient
                    .builder()
                    .settings(settings)
                    .build()
                    .addTransportAddress(
                            new InetSocketTransportAddress(InetAddress
                                    .getByName("localhost"), 9300));

            resctructureIndex(); //Refactoring code
            exec = QueryExecutionFactory.sparqlService(
                    uri, query); // Put URL
            // via
            // params
            results = exec.execSelect();


                qs = results.nextSolution();
                node = qs.get("s");
                sujeto = node.toString();
                authors.add(qs.get("name").toString());
                affiliations.add(qs.get("affiliation").toString());

            while (results.hasNext()) {
                qs = results.nextSolution();

                node = qs.get("s");
                aux = node.toString();
                node_2 = qs.get("name");
                node_3=qs.get("year");
```

Figura 11 Función que verifica si la dirección es correcta y contiene datos. (Fuente: elaboración propia).

Seguidamente se presenta el seudocódigo del método SparqlSincronization.

Algoritmo: Sincronización SPARQL.

Entrada: URI (dirección del SPARQL): http://localhost:3030/backyard/query.

Salida: Incorporar a Elasticsearch los metadatos bibliográficos contenidos en Apache Jena Fuseki.

1: **Inicializar:** configurar el Elasticsearch.

2: Realizar consulta SPARQL: *exec = QueryExecutionFactory.sparqlService*

3: **mientras** *haya resultados de la consulta SPARQL* **hacer**

4: **si** *URI de autor actual = URI del autor anterior* **entonces**

5: Añadir autores y afiliaciones a *L1*

```
6:          sino
7:              para Autor1 hacer
8:                  si cont%2!=0 entonces
9:                      Extraer el nombre del autor y de la lista de afiliaciones el primer nombre
10:                 sino
11:                     Crear un nuevo autor
12:             fin para
13:             para i=0 con i<listaAutores hacer
14:                 si i+1 alcanza el tamaño de listaAutores (llega al final) entonces
15:                     Concatenar los string para insertarlos en Elasticsearch
16:                 Sino (se encuentra en el medio)
17:                     Concatenar los string para insertarlos en Elasticsearch
18:             fin para
19:         fin sino
20:     fin mientras
21:     Devolver: verdadero o falso
```

Luego de este paso, se requiere representar el grafo de flujo asociado al código antes presentado a través de nodos, aristas y regiones, ver Figura 12.

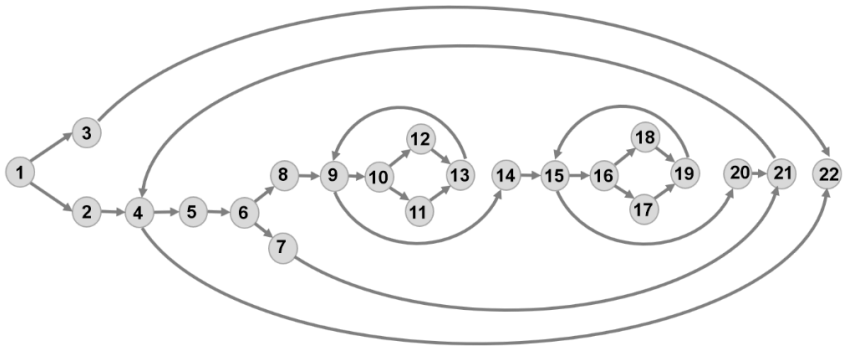


Figura 12 Grafo de flujo asociado al método SparqlSincronization. (Fuente: elaboración propia).

Una vez construido el grafo de flujo asociado al procedimiento anterior se determina la complejidad ciclomática, la cual es una métrica de software útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa (Pressman 2005). El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar.

Cálculo de la complejidad ciclomática para el grafo de flujo de la Figura 12.

La complejidad ciclomática se puede realizar de tres formas:

$$V(g) = a - n + 2\ (I) \qquad V(g) = p + 1\ (II) \qquad V(g) = r\ (II)$$

Donde *"V(g)"* es el valor de la complejidad ciclomática, *"a"* la cantidad total de aristas, *"n"* la cantidad total de nodos, *"p"* la cantidad total de nodos predicados (nodos de los cuales parten dos o más aristas) y *"r"* la cantidad total de regiones.

$$V(g) = 28 - 22 + 2 = 8\ (I) \qquad V(g) = 7 + 1 = 8(II) \qquad V(g) = r = 8\ (II)$$

La evaluación de las fórmulas I, II y III arroja un valor de complejidad ciclomática igual a 8, de manera que existen 8 posibles caminos por donde el flujo puede circular. Este valor representa el número mínimo de casos de pruebas para el procedimiento tratado. Seguidamente, es necesario especificar los caminos básicos que puede tomar el algoritmo durante su ejecución.

Camino básico No.1: 1-3-22

Camino básico No.2: 1-2-4-5-6-8-9-10-11-13-9-14-15-16-17-19-15-20-21-4-22

Camino básico No.3: 1-2-4-5-6-8-9-10-11-13-9-14-15-16-18-19-15-20-21-4-22

Camino básico No.4: 1-2-4-5-6-8-9-10-12-13-9-14-15-16-17-19-15-20-21-4-22

Camino básico No.5: 1-2-4-5-6-8-9-10-12-13-9-14-15-16-18-19-15-20-21-4-22

Camino básico No.6: 1-2-4-22

Camino básico No.7: 1-2-4-5-6-7-21-4-22

Camino básico No.8: 1-2-4-5-6-8-9-14-15-20-21-4-22

Se procede a ejecutar los casos de pruebas para cada uno de los caminos básicos determinados en el grafo de flujo. Para definir los casos de prueba es necesario tener en cuenta: descripción, condición de ejecución, entrada y resultados esperados. Ver Tablas 10 y 11.

Tabla 10 Caso de prueba del camino básico No. 1.

Código: CPCB-01

Descripción:	Los datos de entrada cumplirán el siguiente formato: La dirección de un SPARQL que termine en “query” o “sparql”
Condición de ejecución:	Ocurrencia de una excepción: “No pueden ser insertados datos en el índice”
Entrada:	URI (dirección del SPARQL): http://localhost:3030/backyard/query
Resultados esperados:	Mensaje: “Sincronización incompleta”
Evaluación de la prueba: Satisfactoria	

Tabla 11 Caso de prueba del camino básico No. 2.

Código: CPCB-02	
Descripción:	Los datos de entrada cumplirán el siguiente formato: La dirección de un SPARQL que termine en “query” o “sparql”
Condición de ejecución:	- La URI del artículo en cuestión es distinta a la URI de artículo antes analizado. - El contador no es un número par. - El valor comparado (i+1) es igual al tamaño de la lista de autores
Entrada:	URI (dirección del SPARQL): http://localhost:3030/backyard/query
Resultados esperados:	Indizar los metadatos bibliográficos correspondientes.
Evaluación de la prueba: Satisfactoria	

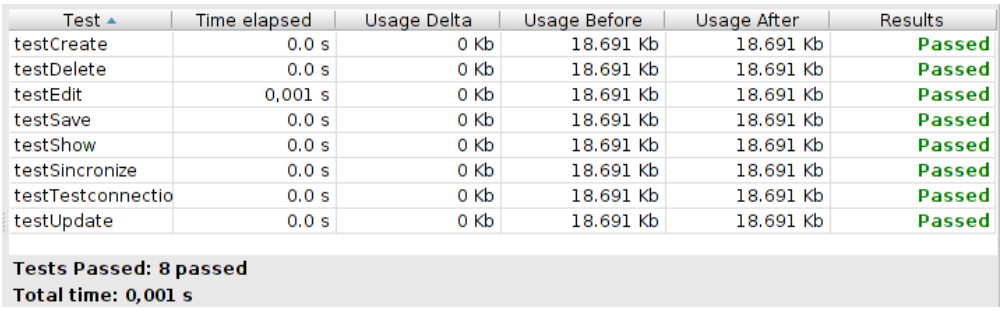
El Anexo 3 recoge los seis casos de prueba de caja blanca restantes. Ver Tablas de la 17 a la 22.

Resultados

En una primera iteración se realizaron un total de 8 casos de pruebas de caja blanca, de los cuales resultaron satisfactorios 6, lo que representa el 75% del total de los casos de prueba. Ver Figura 13.

Figura 13 Resultados de las pruebas de caja blanca. (Fuente: elaboración propia).

El uso del marco de trabajo Groovy JUnit, constituye un entorno para ejecutar pruebas internas en el lenguaje de programación Groovy. Una vez corregidos los errores detectados en la primera iteración de las pruebas se realiza una segunda iteración con un total de 8 casos de prueba mediante el empleo de Groovy JUnit, resultando el 100% de ellos satisfactorios y siendo corregidos los errores detectados tras la primera iteración. Ver Figura 14.



Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
testCreate	0.0 s	0 Kb	18.691 Kb	18.691 Kb	Passed
testDelete	0.0 s	0 Kb	18.691 Kb	18.691 Kb	Passed
testEdit	0,001 s	0 Kb	18.691 Kb	18.691 Kb	Passed
testSave	0.0 s	0 Kb	18.691 Kb	18.691 Kb	Passed
testShow	0.0 s	0 Kb	18.691 Kb	18.691 Kb	Passed
testSincronize	0.0 s	0 Kb	18.691 Kb	18.691 Kb	Passed
testTestconnectio	0.0 s	0 Kb	18.691 Kb	18.691 Kb	Passed
testUpdate	0.0 s	0 Kb	18.691 Kb	18.691 Kb	Passed

Tests Passed: 8 passed
Total time: 0,001 s

Figura 14 Resultados de las pruebas internas mediante el marco de trabajo Groovy JUnit. (Fuente: elaboración propia).

3.2.2 Prueba de caja negra

Para la realización de las pruebas de caja negra se empleó la técnica partición de equivalencia. Esta técnica permite examinar los valores válidos e inválidos de las entradas existentes en el software. A continuación, es sometida a pruebas de aceptación HU-03 definidas en el capítulo anterior. Ver Tablas de la 12 y 13.

Tabla 12 Caso de prueba de aceptación para la búsqueda facetada.

Código: CPCN-01	**Historia de Usuario:** HU-03	
Nombre: Realizar búsquedas textual y facetada en SPARQL sobre el índice de Elasticsearch		
Descripción: En este caso de prueba se verifica el procedimiento que se realiza cuando un usuario marca o desmarca uno o varios de los autores listados en la faceta Autor.		
Acción a probar:	*Datos de entrada:*	*Resultados esperados:*
Seleccionar un autor perteneciente a la faceta Autor	Yusniel Hidalgo Delgado	1. Se deben encontrar todos los artículos del autor seleccionado. 2. Se actualizan las facetas Año y Fuente. 3. Se actualizan las migajas de pan.
Seleccionar múltiples autores pertenecientes a la faceta Autor	Yusniel Hidalgo Delgado Ernesto Ortiz Muñoz	1. Se deben encontrar todos los artículos del autor seleccionado. 2. Se actualizan las facetas Año y Fuente. 3. Se actualizan las migajas de pan.

Deseleccionar un autor perteneciente a la faceta Autor	Yusniel Hidalgo Delgado	1. Solo se deben mostrar artículos para el(los) autor(es) que queda(n) marcado(s), si no queda ninguno marcado volver al estado inicial. 2. Se actualizan las facetas Año y Fuente. 3. Se actualizan las migajas de pan. 4. Se actualizan los resultados.
Evaluación de la prueba: Satisfactoria		

Tabla 13 Caso de prueba de aceptación para la búsqueda textual.

Código: CPCN-02	**Historia de Usuario:** HU-03	
Nombre: Realizar búsquedas textual y facetada en SPARQL sobre el índice de Elasticsearch		
Descripción: En este caso de prueba se verifica el procedimiento que se realiza cuando un usuario inserta un criterio para efectuar una búsqueda textual.		
Acción a probar:	*Datos de entrada:*	*Resultados esperados:*
Inserción de una palabra clave	“semántica”	1. Se deben encontrar todos los artículos donde aparezca la palabra “semántica”. 2. Se actualizan las facetas Autor, Año y Fuente. 3. Se actualizan las migajas de pan.
Inserción de una frase	“web semántica”	1. Se deben encontrar todos los artículos donde aparezca la palabra “web semántica”. 2. Se actualizan las facetas Autor, Año y Fuente. 3. Se actualizan las migajas de pan.
Inserción caracteres	“ç$query”	1. Se deben mostrar un mensaje indicando al usuario que no puede insertar caracteres diferentes a los especificados.
Evaluación de la prueba: Satisfactoria		

Resultados

Se realizaron un total de 12 casos de pruebas de caja negra, de ellos 2 resultaron no satisfactorios, lo cual representa el 17% del total de casos de prueba de caja negra realizados, mientras los 10 casos de prueba restantes resultaron satisfactorios para un 83% del total. Ver Figura 15.

Figura 15 Resultados de las pruebas de caja negra. (Fuente: elaboración propia).

Durante la realización de las pruebas de caja negra, en los 2 casos de prueba que resultaron no satisfactorios fueron detectados 9 errores, de ellos: 5 funciones incorrectas, 3 errores de interfaz y 1 error de rendimiento. Cada error encontrado durante las pruebas realizadas fue mitigado.

3.3. Caso de estudio

Con el objetivo de validar la solución al problema de investigación se diseña un caso de estudio. Se utiliza un grafo RDF con 51329 tripletas RDF almacenadas a priori en el almacén de tripletas y posteriormente incorporadas al servidor de indización, siendo provenientes de 8 revistas científicas cubanas como muestra. Para el caso de estudio se cuenta con un equipo de cómputo con las siguientes prestaciones, las cuales son similares a las empleadas por RACien:

- Tipo de CPU: Intel Core i5 (cuarta generación) a 2.33GHz
- Memoria del sistema: 4GB RAM DDR3 SDRAM

Se proponen los siguientes escenarios para la evaluación:

1. Realizar consultas al grafo RDF sin el empleo de la propuesta de solución empleando RACien.
2. Realizar consultas al grafo RDF utilizando la propuesta de solución como estímulo.

3.4. Diseño experimental

De acuerdo a la clasificación de los experimentos mostrada al inicio del capítulo, se emplea en la investigación un pre-experimento, dado que se precisa el resultado de una observación inicial que será comparada en otro momento con la aplicación de un estímulo. Se definen cuatro tareas a realizar, enumeradas seguidamente:

1. Seleccionar el autor X que más ha publicado con el autor Y en el año A y en la afiliación F.

2. Encontrar los artículos del autor X que pertenecen a la afiliación F.
3. Encontrar la afiliación F que más artículos haya publicado en el año A.
4. Encontrar las dos afiliaciones que más artículos hayan publicado.

A continuación, la Tabla 14 muestra el diseño experimental propuesto.

Tabla 14 Diseño experimental propuesto

Fuente de datos	**Tareas**	**Observación simple**	**Estímulo**	**Observación con estímulo**
G	T_1	OS_1	E	OE_1
	T_2	OS_2		OE_2
	T_3	OS_3		OE_3
	T_4	OS_4		OE_4

La simbología empleada en la tabla anterior es la siguiente:

- **G:** Grafo RDF como fuente de datos.
- **T_i:** Tareas (consultas SPARQL) realizadas sobre G. El subíndice i representa el número de la consulta.
- **OS_i:** Resultado de la observación luego de acometer T. El indicador es el tiempo en segundos que tarda el resultado en ser mostrado.
- **E:** Tratamiento o estímulo. En este caso la aplicación del componente propuesto.
- **OE_i:** Observación realizada tras aplicar E. El indicador es el tiempo en segundos que tarda el resultado en ser mostrado.

3.5. Análisis de resultados

La Tabla 15 refleja los tiempos (en segundos) que tardan en realizarse las cuatro tareas definidas en la sección 3.4 por una muestra de diez usuarios mediante el uso de RACien como el primero de los escenarios de prueba.

Tabla 15 Tiempo que demora el usuario U_i en realizar cuatro tareas mediante el uso de RACien.

	Tiempo (en segundos)									
Tareas	**U_1**	**U_2**	**U_3**	**U_4**	**U_5**	**U_6**	**U_7**	**U_8**	**U_9**	**U_{10}**
1	6	8	7	10	9	6	10	7	8	9
2	2	4	3	5	9	3	5	5	3	4

3	3	5	4	6	5	4	8	8	5	6
4	5	6	7	5	6	5	9	7	8	8

La Tabla 16 corresponde a la aplicación de la propuesta de solución como estímulo en el segundo escenario de prueba. Nótese que los tiempos son mejorados una vez que se reducen a la mitad en la mayoría de los casos. Los resultados obtenidos reflejan que la solución propuesta se comporta de mejor forma que RACien como referente de la investigación para iguales tareas.

Tabla 16 Tiempo que demora el usuario U_i en realizar cuatro tareas mediante el empleo de la propuesta de solución como estímulo.

	Tiempo (en segundos)									
Tareas	**U_1**	**U_2**	**U_3**	**U_4**	**U_5**	**U_6**	**U_7**	**U_8**	**U_9**	**U_{10}**
1	3	4	3	5	4	3	5	3	4	4
2	1	2	1	2	4	1	2	2	1	2
3	1	2	2	3	2	2	4	4	2	3
4	2	3	3	2	3	2	4	3	4	4

3.6. Conclusiones parciales

En este capítulo se diseñó un caso de estudio y se propuso un diseño experimental con el propósito de validar la propuesta de solución presentada en el capítulo anterior. Tras aplicar las pruebas de software definidas y el diseño experimental descrito, se concluye lo siguiente:

1. Se realizaron 8 casos de prueba de caja blanca, de los cuales resultaron satisfactorios 6 representando el 75% del total y 2 no satisfactorios que constituyen el 25% restante, logrando ser corregidos en su totalidad para la segunda iteración.

2. Se realizaron 12 casos de prueba de caja negra, de ellas 10 resultaron satisfactorias representando un 83% del total y 2 no satisfactorias que componen el 17% restante, consiguiendo ser corregidas en su totalidad.

3. El análisis del experimento aplicado demostró que la aplicación de un índice como estructura de dato optimizada reduce el tiempo de las consultas formuladas por los usuarios en el consumo de metadatos bibliográficos publicados como datos enlazados en relación con los índices alcanzados por RACien.

CONCLUSIONES GENERALES

Atendiendo a los objetivos propuestos con esta investigación, se concluye que:

1. El estudio de las técnicas y herramientas a partir de la literatura científica consultada, permitió seleccionar la herramienta Elasticsearch por ser la que más se ajusta a los requerimientos de la investigación. Ésta herramienta proporciona un índice invertido como estructura de datos optimizada para el almacenamiento de grafos RDF.

2. El diseño del componente sdl-index como propuesta de solución para la indización de grafos RDF, integra un almacén de tripletas con un servidor de indización. La implementación de un prototipo funcional provee una interfaz web para la búsqueda textual y facetada sobre los datos en el índice.

3. El análisis del experimento aplicado demostró que la aplicación de un índice como estructura de dato optimizada reduce el tiempo de las consultas formuladas por los usuarios en el consumo de metadatos bibliográficos publicados como datos enlazados en relación con los indicadores alcanzados por RACien.

RECOMENDACIONES

La búsqueda textual y las facetas generadas por la propuesta de solución son dependientes de las ontologías con las que se describen el conjunto de datos. De modo que, si se adicionan nuevas ontologías para describir nuevas clases dentro de los datos, sería necesario redefinir las consultas SPARQL que recuperan los recursos y propiedades desde el grafo RDF almacenado en el servidor. Por tanto, se recomienda definir un modelo ontológico con el propósito de generar un grafo RDF que contenga los campos de metadatos necesarios para la recuperación de la información o bien la implementación de un lenguaje de alineación de datos entre el grafo RDF y el servidor de indización Elasticsearch a través del lenguaje SPARQL.

Se recomienda implementar un algoritmo para la generación de facetas de manera dinámica con el fin de adaptarse a nuevos grafos RDF según alguna de las métricas empleadas para dicho propósito.

REFERENCIAS BIBLIOGRÁFICAS

AGUDELO, C.A.M. y REYES, Y.S., 2012. Prototipo de buscador semántico aplicado a la búsqueda de libros de ingeniería de sistemas y computación en la biblioteca Jorge Roa Martínez de la Universidad Tecnológica de Pereira. [en línea], [Consulta: 4 febrero 2016]. Disponible en: http://repositorio.utp.edu.co/dspace/handle/11059/2671.

ARTIGA, V.A., 1993. Evolución histórica de las Tecnologías de la Información y su aplicación en el proceso documental. *Revista general de información y documentación*, vol. 3, no. 2, pp. 131.

BERNERS-LEE, T., 2006. Linked data-design issues. [en línea], [Consulta: 4 febrero 2016]. Disponible en: http://www.citeulike.org/group/8357/article/3421195.

BERNERS-LEE, T. y CONNOLLY, D., 2008. Notation3 (N3): A readable RDF syntax. W3C recommendation. *World Wide Web Consortium*,

BERNERS-LEE, T., HENDLER, J. y LASSILA, O., 2001. The semantic web. *Scientific american*, vol. 284, no. 5, pp. 28–37.

BIZER, C., JENTZSCH, A. y CYGANIAK, R., 2011. State of the LOD Cloud. *Version 0.3 (September 2011)*, vol. 1803.

BRAY, T., HOLLANDER, D. y LAYMAN, A., 1999. *Namespaces in XML* [en línea]. S.l.: s.n. [Consulta: 4 febrero 2016]. Disponible en: http://www.immagic.com/eLibrary/ARCHIVES/SUPRSDED/W3C/W980916D.pdf.

BRAY, T., PAOLI, J., SPERBERG-MCQUEEN, C.M., MALER, E. y YERGEAU, F., 2008. *Extensible markup language (XML) 1.0*. S.l.: W3C recommendation.

CARRASCO, G., IGNACIO, E., LLEDÓ, R., LORENZO, S., MIRA, J.J., PARRA, P. y PEIRÓ, S., 2004. La indización en Index Medicus/MEDLINE: un reto posible. *Revista de Calidad Asistencial*, vol. 19, no. 2, pp. 55–56.

CHÁVEZ, M.E., CÁRDENAS, O. y BENITO, O., 2005. La web semántica. *Revista de investigación de Sistemas e Informática*, vol. 2, no. 3, pp. 43–54.

CHENG, G. y QU, Y., 2009. Searching linked objects with falcons: Approach, implementation and evaluation. [en línea], [Consulta: 17 mayo 2016]. Disponible en: https://books.google.com/books?hl=es&lr=&id=tP8HLETgbKcC&oi=fnd&pg=PA259&dq=Y.+QU,+2009.+Searching+linked+objects+with+falcons:+Approach,+implementation+and+evaluation&ots=-gPuSpxWFH&sig=QbH7oR78qw0RQ_vdUhf9V7vbTIk.

CLEVELAND, A.D. y CLEVELAND, D.B., 2013. *Introduction to indexing and abstracting* [en línea]. S.l.: ABC-CLIO. [Consulta: 29 marzo 2016]. Disponible en: https://books.google.com/books?hl=es&lr=&id=JfPXAQAAQBAJ&oi=fnd&pg=PP1&dq=Introduction+to+indexing+and+abstracting&ots=3n0sxO9bUK&sig=YATOAQDQAJat9YDFiXXheA6KdaI.

DADZIE, A.-S. y ROWE, M., 2011. Approaches to visualising linked data: A survey. *Semantic Web*, vol. 2, no. 2, pp. 89–124.

DELGADO, Y.H., 2015. Marco de trabajo basado en los datos enlazados para la interoperabilidad semántica en el protocolo OAI-PMH. [en línea], [Consulta: 29 marzo 2016]. Disponible en: http://eprints.rclis.org/28755/.

DELGADO, Y.H. y PUENTE, R.R., 2013. La web semántica: una breve revisión. *Revista Cubana de Ciencias Informáticas*, vol. 7, no. 1, pp. 76-85. ISSN 2227-1899.

EDEKI, C., 2013. Agile Unified Process. *INTERNATIONAL JOURNAL OF COMPUTER SCIENCE* [en línea], vol. 1, no. 3. [Consulta: 31 mayo 2016]. Disponible en: http://www.ijcsma.com/publications/september2013/V1I304.pdf.

FERNÁNDEZ, L.C., 2009. Procedimiento semi-automático para transformar la web en web semántica. [en línea], [Consulta: 4 febrero 2016]. Disponible en: http://e-spacio.uned.es/fez/eserv/tesisuned:IngInf-Lcriado/Documento.pdf.

GARCÍA, A.A.R., 2013. El aprovechamiento de los metadatos en las bibliotecas. *e-Ciencias de la Información*, pp. 1–13.

GONZÁLES-CAM, C., 2003. Arquitectura de la información: diseño e implentación. *Bibliodocencia: Revista de Profesores de Bibliotecología*, vol. 1, no. 5, pp. 15–18.

GRAU, R., CORREA, C. y ROJAS, M., 2004. Metodología de la Investigación (Segunda Edición ed.). *Ibagué: El POIRA Editores SA*,

GREENBERG, J., 2003. Metadata and the world wide web. *Encyclopedia of library and information science*, vol. 3, pp. 1876–1888.

GRUBER, T.R., 1993. A translation approach to portable ontology specifications. *Knowledge acquisition*, vol. 5, no. 2, pp. 199–220.

GUERROUJ, L., 2013. Normalizing source code vocabulary to support program comprehension and software quality. *Proceedings of the 2013 International Conference on Software Engineering* [en línea]. S.l.: IEEE Press, pp. 1385–1388. [Consulta: 15 junio 2016]. Disponible en: http://dl.acm.org/citation.cfm?id=2487012.

HAASE, P., BROEKSTRA, J., EBERHART, A. y VOLZ, R., 2004. A comparison of RDF query languages. *The Semantic Web–ISWC 2004* [en línea]. S.l.: Springer, pp. 502–517. [Consulta: 4 febrero 2016]. Disponible en: http://link.springer.com/10.1007%2F978-3-540-30475-3_35.

HARRIS, S., SEABORNE, A. y PRUD'HOMMEAUX, E., 2013. SPARQL 1.1 query language. *W3C Recommendation*, vol. 21.

HARTH, A., 2010. VisiNav: A system for visual search and navigation on web data. *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 8, no. 4, pp. 348–354.

HASLHOFER, B. y ISAAC, A., 2011. data. europeana. eu: The europeana linked open data pilot. *International Conference on Dublin Core and Metadata Applications* [en línea]. S.l.: s.n., pp. 94–104. [Consulta: 29 marzo 2016]. Disponible en: http://dcpapers.dublincore.org/index.php/pubs/article/view/3625.

HEATH, T., 2008. How will we interact with the web of data? *Internet Computing, IEEE*, vol. 12, no. 5, pp. 88–91.

HEATH, T. y BIZER, C., 2011. Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: theory and technology*, vol. 1, no. 1, pp. 1–136.

HERNÁNDEZ, J.P.R. y HERNÁNDEZ, G.A., 2008. Indización y Búsqueda a través de Lucene. *Veracruz, Sinaloa* [en línea], [Consulta: 29 marzo 2016]. Disponible en: https://intranet.adsib.gob.bo/proyectos/attachments/931/3013167-Indizacion-y-Busqueda-a-traves-de-Lucene.pdf.

ISO, I., 2011. IEEE. 29148: 2011-Systems and software engineering-Requirements engineering. . S.l.: Technical report.

KANDOGAN, E., KRISHNAMURTHY, R., RAGHAVAN, S., VAITHYANATHAN, S. y ZHU, H., 2006. Avatar semantic search: a database approach to information retrieval. *Proceedings of the 2006 ACM SIGMOD international conference on Management of data* [en línea]. S.l.: ACM, pp. 790–792. [Consulta: 17 mayo 2016]. Disponible en: http://dl.acm.org/citation.cfm?id=1142591.

KARVOUNARAKIS, G., MAGKANARAKI, A., ALEXAKI, S., CHRISTOPHIDES, V., PLEXOUSAKIS, D. y SCHOLL, M., 2013. 18. RQL: A Functional Query Language for. *The Functional Approach to Data Management: Modeling, Analyzing and Integrating Heterogeneous Data*, pp. 435.

KHAN, H.U. y MALIK, T.A., 2012. Finding resources from middle of RDF graph and at Sub-Query level in suffix array based RDF indexing using RDQL queries. *International Journal of Computer Theory and Engineering*, vol. 4, no. 3, pp. 369.

KNIBERG, H., 2015. *Scrum and XP from the Trenches* [en línea]. S.l.: Lulu. com. [Consulta: 31 mayo 2016]. Disponible en: https://books.google.com/books?hl=es&lr=&id=R4oXCgAAQBAJ&oi=fnd&pg=PA11&dq=scrum&ots=ecGraujOoh&sig=C8H7fDDc0drscW3CH9pWPbLGulk.

LI, G., JI, S., LI, C. y FENG, J., 2009. Efficient type-ahead search on relational data: a tastier approach. *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data* [en línea]. S.l.: ACM, pp. 695–706. [Consulta: 17 mayo 2016]. Disponible en: http://dl.acm.org/citation.cfm?id=1559918.

LOBO, M.D.O., 1999. Métodos y técnicas para la indización y recuperación de los recursos de la World Wide Web. *Boletín de la Asociación Andaluza de Bibliotecarios* [en línea], no. 57. [Consulta: 29 marzo 2016]. Disponible en: http://eprints.rclis.org/5980.

LÓPEZ, P.L. y PERELLÓ, J.G., 2005. *Información, conocimiento y bibliotecas en el marco de la globalización neoliberal* [en línea]. S.l.: Trea. [Consulta: 4 febrero 2016]. Disponible en: https://www.researchgate.net/profile/Rosa_San_Segundo/publication/33401520_Informacin_conocimiento_y_bibliotecas_en_el_marco_de_la_globalizacin_neoliberal_book_review/links/0912f50db41a9e11ec000000.pdf.

MAALI, F., CYGANIAK, R. y PERISTERAS, V., 2011. Re-using Cool URIs: Entity Reconciliation Against LOD Hubs. *LDOW* [en línea], vol. 813. [Consulta: 17 mayo 2016]. Disponible en: http://events.linkeddata.org/ldow2011/papers/ldow2011-paper11-maali.pdf.

MACÁRIO, C.G.N., DE SOUSA, S.R. y MEDEIROS, C.B., 2010. Play It Again, SAM–Using Scientific Workflows to Drive the Generation of Semantic Annotations. *e-Science (e-Science), 2010 IEEE Sixth International Conference on* [en línea]. S.l.: IEEE, pp. 284–291. [Consulta: 4 febrero 2016]. Disponible en: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5693929.

MAHMOOD, J. y REDDY, Y.R., 2014. Automated refactorings in Java using IntelliJ IDEA to extract and propogate constants. *Advance Computing Conference (IACC), 2014 IEEE International* [en línea]. S.l.: IEEE, pp. 1406–1414. [Consulta: 30 marzo 2016]. Disponible en: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6779532.

MANOLA, F. y MILLER, E., 2004. RDF Primer. [en línea]. [Consulta: 4 febrero 2016]. Disponible en: https://www.w3.org/TR/2004/REC-rdf-primer-20040210/#intro.

MARTÍNEZ, N.S., 2014. *Método para la Transformación Automatizada de Modelos de Procesos de Negocio a Modelos de Componentes para Sistemas de Gestión Empresarial*. S.l.: Universidad de las Ciencias Informáticas.

MATEUS, S.P., RUIZ, M.A. y PLAZA, J.E.G., 2015. Lenguajes de recuperación de información sobre la web semántica. *REVISTA POLITÉCNICA*, vol. 5, no. 8, pp. 39–46.

NIELSEN, J., 2007. Breadcrumb navigation increasingly useful. *Jakob Nielsen's Alertbox*,

NOWACK, B., 2009. Paggr: Linked Data widgets and dashboards. *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 7, no. 4, pp. 272–277.

OREN, E., DELBRU, R. y DECKER, S., 2006. Extending faceted navigation for RDF data. *The Semantic Web-ISWC 2006* [en línea]. S.l.: Springer, pp. 559–572. [Consulta: 17 mayo 2016]. Disponible en: http://link.springer.com/chapter/10.1007/11926078_40.

PEÑA, C.N., 2003. Indización y clasificación: un problema conceptual y terminológico. *Documentación de las Ciencias de la Información*, no. 26, pp. 23–40.

PRESSMAN, R.S., 2005. *Ingeniería de Software, Sexta Edición, Ed*. S.l.: Mc Graw Hill, Mexico.

PRUD'HOMMEAUX, E., CAROTHERS, G., BECKETT, D. y BERNERS-LEE, T., 2013. Turtle–Terse RDF Triple Language. *Candidate Recommendation, W3C*,

RIZO, C.H. y GARCÍA, C.H.C., 2013. *Consumo de datos enlazados mediante búsqueda textual y facetada*. S.l.: Universidad de las Ciencias Informáticas.

RUIZ, J.H., ALMANZA, L.Á. y PONS, N.L., 2011. Comparación y tendencias entre metodologías ágiles y formales. Metodología utilizada en el Centro de Informatización para la Gestión de Entidades. *Serie Científica* [en línea], vol. 4, no. 10. [Consulta: 31 mayo 2016]. Disponible en: https://publicaciones.uci.cu/index.php/SC/article/view/484/0.

SCHREIBER, G. y RAIMOND, Y., 2014. RDF 1.1 Primer. [en línea]. [Consulta: 4 febrero 2016]. Disponible en: https://www.w3.org/TR/rdf11-primer/#section-Introduction.

SHANHONG, T., 2000. Gestión del conocimiento en las bibliotecas del siglo XXI. *66th IFLA Council and General Conference* [en línea]. S.l.: s.n., pp. 13–18. [Consulta: 4 febrero 2016]. Disponible en: http://www.hacienda.go.cr/centro/datos/Articulo/Gesti%C3%B3n%20del%20conocimiento%20en%20las%20Bibliotecas%20del%20siglo%20XXI.doc.

SICILIA, M.-A., 2014. METADATA RESEARCH: MAKING DIGITAL RESOURCES USEFUL AGAIN? *Handbook of metadata, semantics and ontologies* [en línea]. S.l.: World Scientific, [Consulta: 4 febrero 2016]. Disponible en: http://www.worldscientific.com/doi/pdf/10.1142/9789812836304_fmatter.

SINGH, T. y SHARMA, A., 2015. Research work and changing dimensions of digital library: A review. *Emerging Trends and Technologies in Libraries and Information Services (ETTLIS), 2015 4th International Symposium on* [en línea]. S.l.: IEEE, pp. 39–42. [Consulta: 4 febrero 2016]. Disponible en: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7048169.

STUDER, R., BENJAMINS, V.R. y FENSEL, D., 1998. Knowledge engineering: principles and methods. *Data & knowledge engineering*, vol. 25, no. 1, pp. 161–197.

SUÁREZ, A.R., NAVARRETE, R.B. y LEÓN, A.L.P., 2007. Indización en línea:?` capricho o necesidad? *Acimed*, vol. 15, no. 1, pp. 0–0.

SUDEEPTHI, G., ANURADHA, G. y BABU, M.S.P., 2012. A survey on semantic web search engine. *IJCSI International Journal of Computer Science Issues* [en línea], vol. 9, no. 2. [Consulta: 4 febrero 2016]. Disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.403.482&rep=rep1&type=pdf.

SUERO, D.V., TERRAZAS, B.V. y PÉREZ, A.G., 2013. datos. bne. es: A library linked dataset. *Semantic Web*, vol. 4, no. 3, pp. 307–313.

SUMMERS, E., ISAAC, A., REDDING, C. y KRECH, D., 2008. LCSH, SKOS and linked data. *Universitätsverlag Göttingen*, pp. 25.

SUTHERLAND, J., 2015. *Scrum: El arte de hacer el doble de trabajo en la mitad de tiempo*. S.l.: Grupo Planeta (GBS).

TELLO, J.C., 2006. La Web Semántica y el lenguaje RDF. [en línea], [Consulta: 4 febrero 2016]. Disponible en: https://portal.uah.es/portal/page/portal/GP_EPD/PG-MA-PROF/OLD_PG-PROF-138886%202008-07-14%2010-07-31/TAB4348465/TAB4348469/TAB4348477/Articulo_WebSemantica_Jesus_Caceres_CISTI_06.pdf.

TKACZYK, D., SZOSTEK, P., DENDEK, P.J., FEDORYSZAK, M. y BOLIKOWSKI, L., 2014. Cermine–automatic extraction of metadata and references from scientific literature. *Document Analysis Systems (DAS), 2014 11th IAPR International Workshop on* [en línea]. S.l.: IEEE, pp. 217–221. [Consulta: 29 marzo 2016]. Disponible en: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6831001.

TRAN, T., HERZIG, D.M. y LADWIG, G., 2011. SemSearchPro–Using semantics throughout the search process. *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 9, no. 4, pp. 349–364.

TRAN, T. y MIKA, P., 2012. Semantic search-systems, concepts, methods and the communities behind it. *TKDE journal*, pp. 1-21.

TUMMARELLO, G., CYGANIAK, R., CATASTA, M., DANIELCZYK, S., DELBRU, R. y DECKER, S., 2010. Sig. ma: Live views on the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 8, no. 4, pp. 355–364.

VARGAS, B.C., 2012. El papel de las bibliotecas y la educación en la gestión del conocimiento de la sociedad contemporánea. *Alexandría: Revista de Ciencias de la Información*, vol. 4, no. 6, pp. 15-19. ISSN 1991-1653.

ANEXOS

Anexo 1. Interfaces del componente implementado.

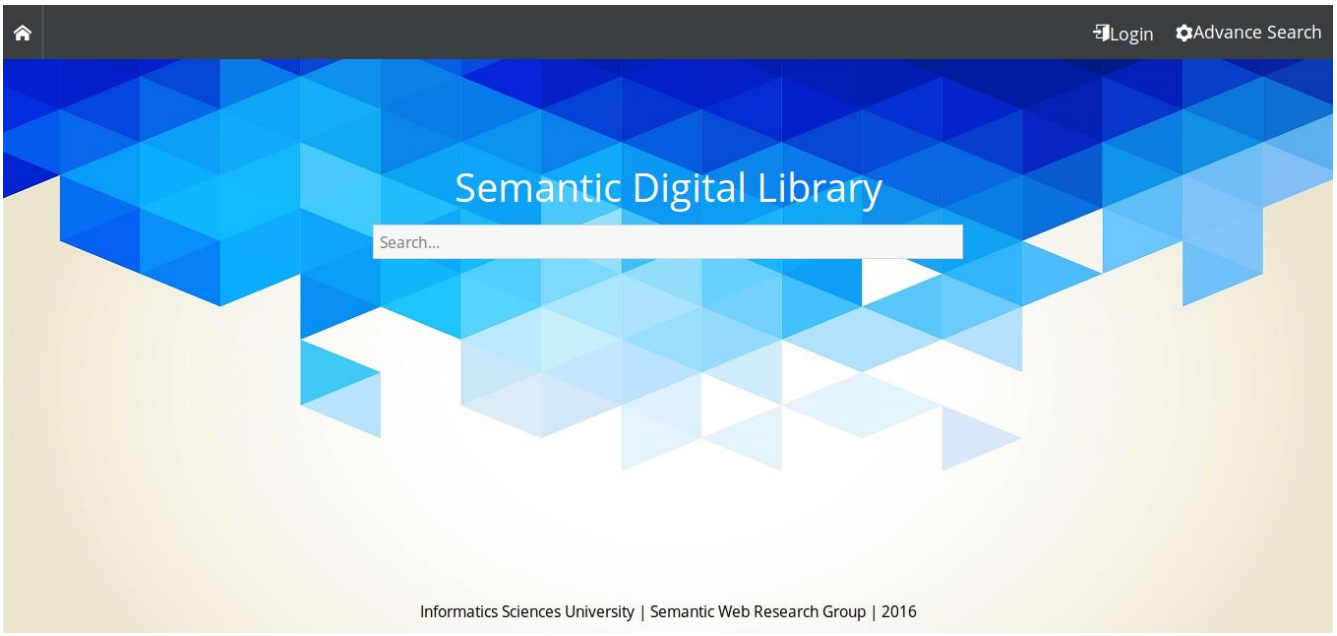


Figura 16 Vista principal de la Biblioteca Digital Semántica. (Fuente: elaboración propia).

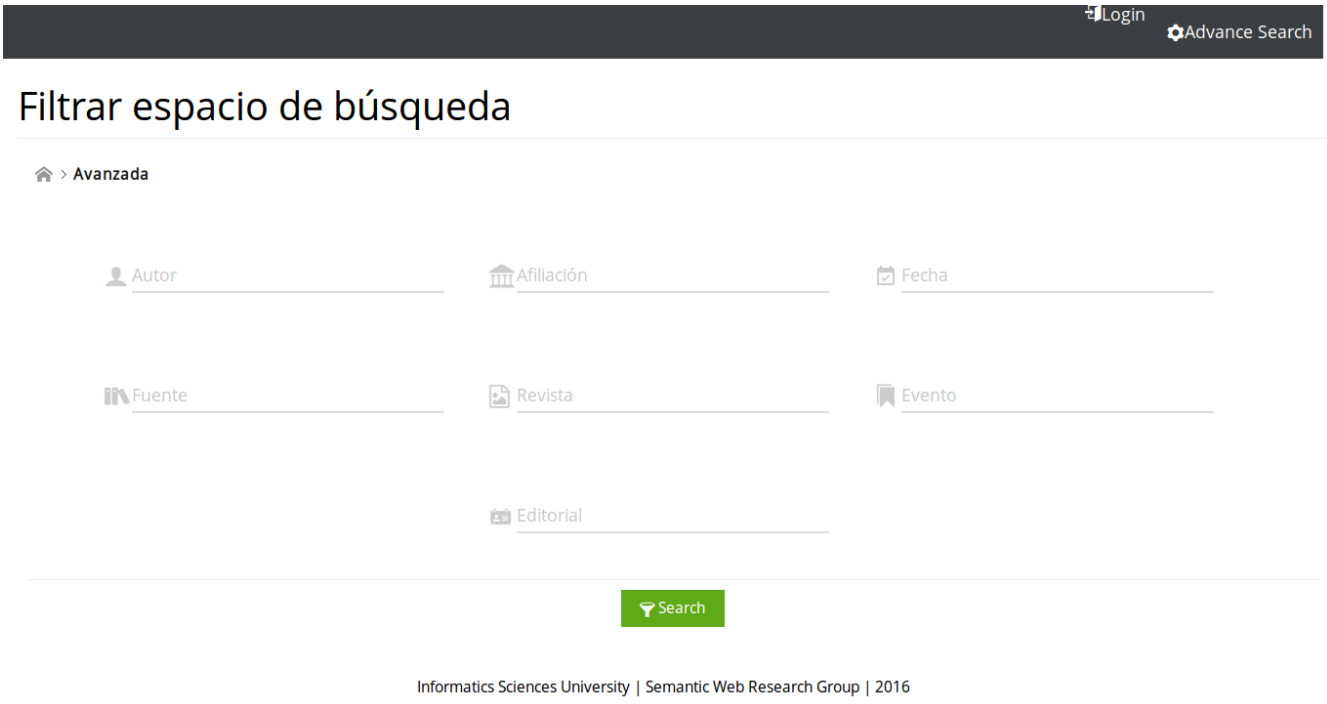


Figura 17 Vista para la búsqueda avanzada de la propuesta de solución implementada. (Fuente: elaboración propia).

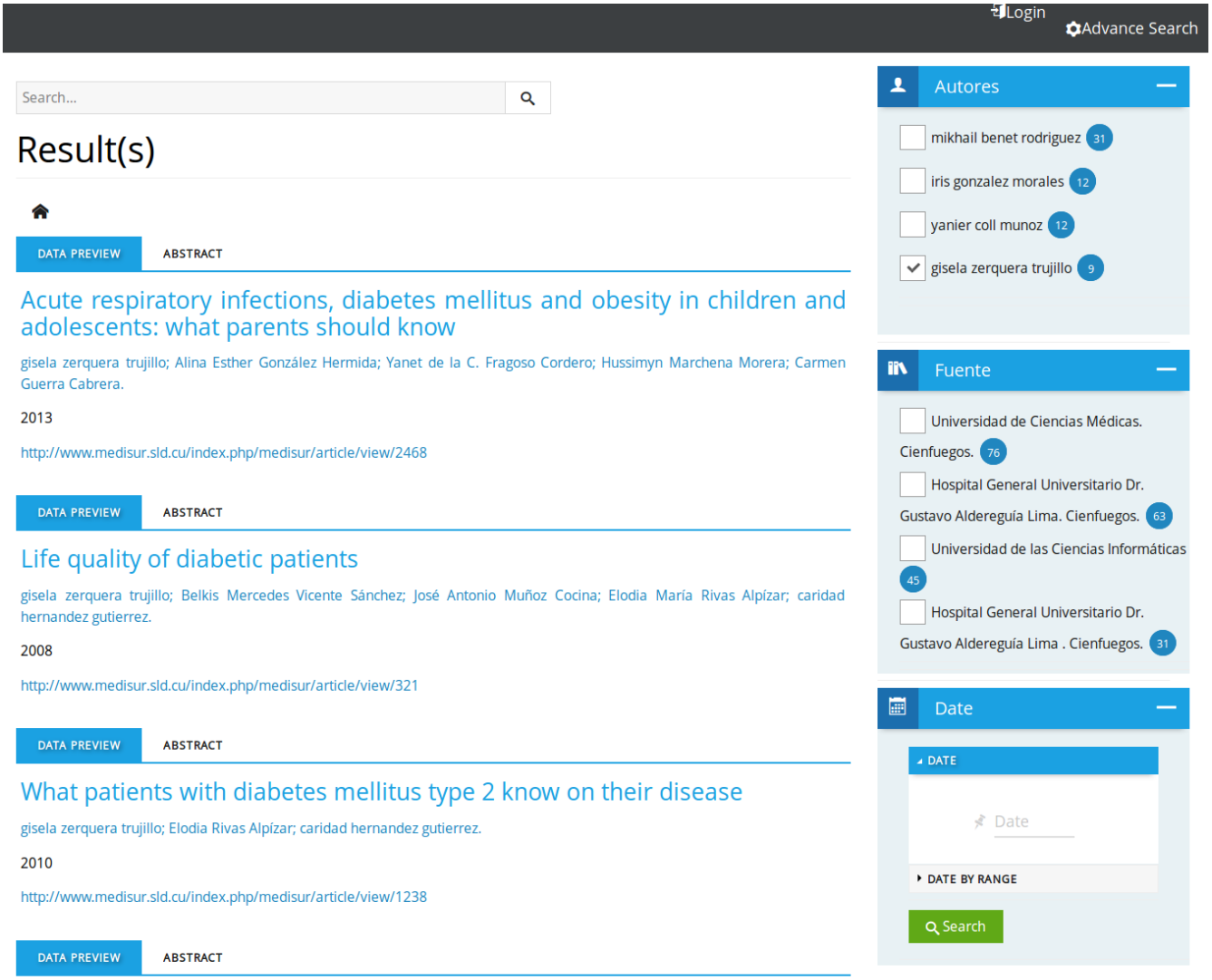


Figura 18 Vista para la búsqueda facetada y los resultados de búsqueda de la propuesta de solución implementada. (Fuente: elaboración propia).

Anexo 2. Estándares de código.

Nomenclatura según el tipo de clases

Clases controladoras: las clases que se encuentran dentro de la carpeta *controllers* después del nombre de la clase incorporan la palabra: “Controller”, siguiendo la nomenclatura UpperCamelCase.

Ejemplo: SearchController.

Clases de dominio: las clases que se encuentran dentro de la carpeta *domain* y al igual que las clases controladoras siguen la notación UpperCamelCase.

Ejemplo: Sincronization.

Nomenclatura de las funcionalidades y atributos

El nombre a emplear para las funciones y los atributos se escriben con la inicial minúscula, en caso de que sea un nombre compuesto se empleará la notación CamelCase.

Ejemplo de función: sincronize(). El nombre de este método está compuesto por una sola palabra, debido a esto es que se escribe con minúscula, si fuera un nombre compuesto por más de una palabra se procede a aplicar la nomenclatura antes mencionada.

Ejemplo de atributo: startTime. El nombre del atributo está compuesto por dos palabras, la primera en minúscula y la segunda iniciando con letra mayúscula.

Nomenclatura de los comentarios

Los comentarios deben ser lo suficientemente claros y precisos de forma tal que se entienda el propósito de lo que se está desarrollando. Su uso ayuda a una mejor comprensión del código por parte de terceros para futuras adaptaciones.

Anexo 3. Casos de prueba de caja blanca.

Tabla 17 Caso de prueba del camino básico No. 3.

Código: CPCB-03	
Descripción:	Los datos de entrada cumplirán el siguiente formato: La dirección de un SPARQL que termine en “query” o “sparql”
Condición de ejecución:	- La URI del artículo en cuestión es distinta a la URI de artículo antes analizado. - El contador no es un número par. - El valor comparado (i+1) es distinto al tamaño de la lista de autores
Entrada:	URI (dirección del SPARQL): http://localhost:3030/backyard/query
Resultados esperados:	Indizar los metadatos bibliográficos correspondientes.
Evaluación de la prueba: Satisfactoria	

Tabla 18 Caso de prueba del camino básico No. 4.

Código: CPCB-04	
Descripción:	Los datos de entrada cumplirán el siguiente formato: La dirección de un SPARQL que termine en “query” o “sparql”
Condición de ejecución:	- La URI del artículo en cuestión es distinta a la URI de artículo antes analizado. - El contador es un número par. - El valor comparado (i+1) es igual al tamaño de la lista de autores
Entrada:	URI (dirección del SPARQL): http://localhost:3030/backyard/query
Resultados esperados:	Indizar los metadatos bibliográficos correspondientes.
Evaluación de la prueba: Satisfactoria	

Tabla 19 Caso de prueba del camino básico No. 5.

Código: CPCB-05	
Descripción:	Los datos de entrada cumplirán el siguiente formato: La dirección de un SPARQL que termine en “query” o “sparql”
Condición de ejecución:	- La URI del artículo en cuestión es distinta a la URI de artículo antes analizado. - El contador es un número par. - El valor comparado (i+1) es distinto al tamaño de la lista de autores
Entrada:	URI (dirección del SPARQL): http://localhost:3030/backyard/query

Resultados esperados:	Indizar los metadatos bibliográficos correspondientes.
Evaluación de la prueba: Satisfactoria	

Tabla 20 Caso de prueba del camino básico No. 6.

Código: CPCB-06	
Descripción:	Los datos de entrada cumplirán el siguiente formato: La dirección de un SPARQL que termine en “query” o “sparql”
Condición de ejecución:	No existen datos en el SPARQL.
Entrada:	URI (dirección del SPARQL): http://localhost:3030/backyard/query
Resultados esperados:	Mensaje: “Sincronización incompleta”
Evaluación de la prueba: Satisfactoria	

Tabla 21 Caso de prueba del camino básico No. 7.

Código: CPCB-07	
Descripción:	Los datos de entrada cumplirán el siguiente formato: La dirección de un SPARQL que termine en “query” o “sparql”
Condición de ejecución:	La URI del artículo en cuestión es igual a la URI de artículo antes analizado.
Entrada:	URI (dirección del SPARQL): http://localhost:3030/backyard/query
Resultados esperados:	Se actualizan la lista de autores y de afiliaciones.
Evaluación de la prueba: Satisfactoria	

Tabla 22 Caso de prueba del camino básico No. 8.

Código: CPCB-08	
Descripción:	Los datos de entrada cumplirán el siguiente formato: La dirección de un SPARQL que termine en “query” o “sparql”
Condición de ejecución:	La lista de autores es vacía.
Entrada:	URI (dirección del SPARQL): http://localhost:3030/backyard/query
Resultados esperados:	No se adicionan metadatos al Elasticsearch.
Evaluación de la prueba: Satisfactoria	