TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

AUCTORITAS

Sistema de apoyo para el control de autoridades

Autores:

Dailién Calzadilla Reyes

Wilbert Alberto Ruano Alvarez

Tutor: Ing. Leandro Tabares Martín

Co – tutor: Ing. Luis Carlos Alvarez Fernández

Facultad 2

Universidad de las Ciencias Informáticas

La Habana, junio de 2015

Para ser exitoso no tienes que hacer cosas extraordinarias. Haz cosas ordinarias, extraordinariamente bien.

Jim Rohn

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos 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 _____.

Wilbert Alberto Ruano Alvarez	Dailién Calzadilla Reyes
Autor	Autor
Ing. Leandro Tabares Martín	Ing. Luis Carlos Álvarez Fernández
Tutor	Co - tutor

Datos de contacto

Autores:

Dailién Calzadilla Reyes (correo electrónico: dcreyes@estudiantes.uci.cu)

Wilbert Alberto Ruano Alvarez (correo electrónico: waruano@estudiantes.uci.cu)

Tutor:

Ing. Leandro Tabares Martín (correo electrónico: ltmartin@uci.cu)

Co – tutor:

Ing. Luis Carlos Álvarez Fernández (correo electrónico: lcalvarez@uci.cu)

Agradecimientos

Dailién:

A mis padres, quienes siempre me enseñaron que no hay sueños difíciles de alcanzar, ni montañas que no se puedan escalar, gracias sencillamente por ser los mejores padres del mundo, porque este título hoy es tanto mío como de ustedes. A mi hermana por ser mi confidente, mi madre en ocasiones y mi mejor amiga en otras, gracias a ella y a mi cuñado porque siempre fueron los hermanos más preocupados que he conocido en la vida. A toda mi familia que siempre ha estado presente en estos cinco años, gracias porque dicen que la familia no se puede escoger, pero aún si se pudiera yo los escogería a todos y cada uno de ustedes nuevamente. A mi novio quien ha sido capaz de sacar de mis labios una sonrisa y un te amo aun cuando vivía momentos de mucho estrés, gracias por apoyarme, mimarme y ser una de las más grandes alegrías que he vivido en estos años, te amo con el corazón. A Lisy quien además de ser mí amiga, ha sido mi hermana en esta universidad, gracias por los momentos que hemos vivido juntas. A mi tutor por todo su apoyo, por cuidarnos tanto que en ocasiones le decían que recordara que no éramos sus hijos, gracias porque desde el inicio de la tesis nos hizo sentir como tal, educándonos y enseñándonos como un padre a sus hijos. A todos mis compañeros y amigos por compartir los buenos y malos momentos. A todos los que de una u otra forma ayudaron a la realización de este trabajo. A mi compañero de tesis, porque aún en los momentos en que nos fajábamos no dejaste de ser el mejor compañero, gracias porque sin ti este resultado no hubiese sido posible.

Wilbert:

A mi compañera de tesis, por elegirme para hacer este trabajo con ella. A mis compañeros de aula por compartir estos inolvidables cinco años conmigo, especialmente a Lisy, una amiga sin igual. A mis padres por darme ánimo cada vez que me estresé y por confiar en mí. A nuestro tutor, por guiarnos durante todo el desarrollo del presente trabajo.

Dedicatoria

Dailién:

A mis maravillosos padres, por ser mi fuerza y mi mayor inspiración tanto en los buenos como en los malos momentos, los amo y no tengo la menor duda de que son los mejores padres del mundo.

A mi hermana porque aunque no esté físicamente conmigo en este momento, sé que su corazón está a mi lado, te amo mi tati bella.

A mis abuelos y demás familias por estar siempre presente en mi vida, los amo a todos.

A mi novio por apoyarme incondicionalmente y por su amor, que en muchas ocasiones fue el motor que me impulsó a seguir adelante.

Wilbert:

A mi padres, los mejores del mundo, por hacer realidad todos mis sueños.

A mi segunda madre, mi tía, por su amor incondicional.

A mis hermanos, a quienes quiero mucho con sus virtudes y sus defectos.

A toda mi familia, por poner todas sus esperanzas en mí, gracias por su apoyo y su confianza.

Resumen

El control de autoridades es un proceso complejo y costoso, llevado a cabo por sistemas de gestión bibliotecarios y para la gestión de repositorios digitales en todas partes del mundo. En el caso de los nombres de autores, el control de autoridades se refiere a generar los mismos de forma única bajo reglas definidas previamente. Este proceso requiere de una gran cantidad de información para ser llevado a término eficazmente; por lo que para facilitar el mismo, varias bibliotecas comparten libremente sus datos de control de autoridades.

La no existencia de una herramienta para el control de autoridades en proyectos del Centro de Informatización de la Gestión Documental (CIGED) de la Universidad de las Ciencias Informáticas (UCI), motivó el desarrollo de la presente investigación; la cual tuvo como objetivo desarrollar una herramienta informática que, haciendo uso de las fuentes de datos en forma de LOD que aportan diferentes instituciones, permita el control de autoridades realizado como parte del proceso de catalogación de materiales bibliográficos.

Como resultado de la misma se obtuvo la herramienta AUCTORITAS, capaz de proveer servicios web para la consulta de datos de autores personales y epígrafes de materia, mediante el aprovechamiento de las fuentes de datos en forma de Datos Enlazados Abiertos. El despliegue de la misma pretende prevenir la duplicidad de información y reducir el tiempo de búsqueda que lleva localizar una determinada publicación.

Palabras clave: control de autoridades, Datos Enlazados Abiertos, término autorizado

Índice de contenido

Índice de ilustraciones

Índice de tablas

Introducción

La Web ha cambiado profundamente las formas de comunicación, hacer negocios y realizar diversos trabajos. La comunicación prácticamente con todo el mundo, en cualquier momento y a bajo coste es posible hoy en día: se pueden realizar transacciones económicas a través de Internet; se tiene acceso a millones de recursos, independientemente de la situación geográfica e idioma. Todos estos factores han contribuido al éxito de la Web, sin embargo, al mismo tiempo, han originado sus principales problemas: sobrecarga de información y heterogeneidad de fuentes de información con el consiguiente problema de interoperabilidad[1] (1).

La necesidad de mejorar la interoperabilidad dentro de la *World Wide Web*[2] dio lugar a la creación de la Web Semántica, que a su vez condujo a la aparición de nuevos métodos para controlar y estandarizar la descripción de documentos, solucionar problemas de diversos sistemas de indexación y mejorar la interoperabilidad de registros. El control de autoridades que se lleva a cabo como parte del proceso de catalogación, constituye un problema global, que afecta no sólo a bibliotecas sino organizaciones de todo tipo. La publicación de datos de autoridad en Internet en una forma heterogénea o arbitraria produce ineficiencia en la recuperación de información y crea complicaciones al atribuirle la autoridad a un trabajo dado (2).

La comunidad bibliotecológica ha estado consciente por un largo tiempo de la necesidad del control de autoridades. Esto se puede evidenciar desde el siglo XIX, cuando las primeras regulaciones para la producción de catálogos, tituladas “Reglas para un Catálogo Impreso del Diccionario”, salieron a la luz en 1876 por Charles Cutter[3]. Fue seguido casi inmediatamente por las Reglas de Catalogación Angloamericanas (AACR), descritas en las

[1] El Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) define interoperabilidad como la habilidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada.
[2] La World Wide Web (o Red informática mundial) comúnmente conocida como la Web, es un sistema de distribución de documentos de hipertexto o hipermedios interconectados y accesibles vía Internet.
[3] Charles Ammi Cutter (14 de marzo de 1837 – 6 de septiembre de 1903) fue un bibliotecario estadounidense. Sus aportes a la Biblioteconomía lo sitúan como una figura clave del siglo XIX junto a Melvil Dewey. Es el creador de los principios teórico – prácticos que desembocarían en la creación de los listados de encabezamientos de materia de bibliotecas (o LEMB).

obras de Fumagalli[4] (3) y Tillett[5] (4). Ambos documentos son usados hoy en día como herramientas de control en bibliotecas y agencias de información de todo el mundo (2).

El control de autoridades, en el caso de los nombres de los autores, se refiere a la generación de forma única de los mismos, acorde a reglas previamente definidas. Esta ha sido un área de debate entre los bibliotecarios durante aproximadamente el último siglo y medio; numerosas son las opiniones sobre la forma en que deben registrarse los nombres de autores en los catálogos de las bibliotecas. Algunos han expuesto que esto no es necesario, que basta con registrar las entradas bibliográficas en el catálogo; otros siguen debatiendo el tema (5).

Lo cierto es que el gran volumen de información existente actualmente, debido al advenimiento de las Tecnologías de la Información y las Comunicaciones (TIC), hace que el control de autoridades sea vital a la hora de recuperar los registros. Abundantes son los ejemplos de redundancias en los cuales, el nombre de los autores es registrado de diversas formas, o sea, un mismo autor se muestra como si fueran diferentes personas (5). Esta situación motivó a que varias instituciones pusieran a disposición de la comunidad internacional, sus datos de autoridad en forma de Datos Enlazados Abiertos (LOD por sus siglas en inglés), posibilitando esto la reducción de costos y la agilización de los procesos de gestión bibliográfica, pues al reutilizar todos la información disponible, no se duplica el trabajo.

La realización de entrevistas a especialistas de los proyectos desarrollados en el Centro de Informatización de la Gestión Documental (CIGED), de la Universidad de las Ciencias Informáticas (UCI), arrojó que una de las principales dificultades que afrontan los mismos, es que no cuentan con un mecanismo para el control de autoridades, que aproveche las fuentes de datos en forma de LOD que aportan diversas instituciones. Esto provoca la existencia de primitivas formas para el control de autoridades e incluso su inexistencia (véase en Anexos: Entrevistas realizadas, las entrevistas #1 de cada centro). Estos problemas traen consigo duplicidad de información, lo que dificulta en muchos casos la localización de publicaciones y nombres de autores e incrementa el tiempo de búsqueda de los mismos.

[4] Giuseppe Fumagalli (27 de julio 1863 – 11 de mayo de 1939). Bibliógrafo, bibliólogo, bibliotecario, bibliotecónomo, polígrafo, nacido en Florencia, Italia.
[5] Barbara Tillet es una bibliotecaria famosa por sus trabajos en el control de autoridades y el modelado de datos bibliográficos.

Por las razones expuestas anteriormente se plantea el **problema a resolver**: ¿cómo facilitar el control de autoridades en el proceso de catalogación, mediante el aprovechamiento de las fuentes de datos en forma de LOD que aportan diferentes instituciones?

Dicho problema se enmarca en el **objeto de estudio** el proceso de catalogación de materiales bibliográficos, centrándose en el **campo de acción** el control de autoridades realizado en sistemas de gestión bibliotecarios[6] y para la gestión de repositorios digitales[7].

Para dar solución al problema planteado se estableció como **objetivo general** desarrollar una herramienta informática que, haciendo uso de las fuentes de datos en forma de LOD que aportan diferentes instituciones, permita el control de autoridades realizado como parte del proceso de catalogación de materiales bibliográficos.

Se definieron los siguientes **objetivos específicos**:

1) Definir los fundamentos teóricos y metodológicos relevantes del control de autoridades y tecnologías asociadas, para aplicarlos al sistema.
2) Analizar y diseñar de las herramientas de pre – procesamiento de datos y AUCTORITAS.
3) Implementar el *software* propuesto y herramientas auxiliares.
4) Probar y validar la solución obtenida para detectar posibles errores y corregirlos posteriormente.

Para dar cumplimiento a los objetivos planteados se determinaron las **tareas de investigación** que se enuncian a continuación:

1) Estudio de los procesos de negocio asociados al control de autoridades en el proceso de catalogación, realizado en sistemas de gestión bibliotecarios y repositorios digitales de acceso abierto.
2) Selección de las herramientas, tecnologías y metodología de desarrollo a utilizar.

[6] Sistema de Gestión Bibliotecaria: herramienta informática que permite automatizar los procesos inherentes a una biblioteca.
[7] Repositorio digital: sitio web centralizado donde se almacena y mantiene información digital, habitualmente bases de datos o archivos informáticos. Pueden contener los archivos en su servidor o referenciar desde su web al alojamiento originario. Pueden ser de acceso público, o pueden estar protegidos y necesitar de una autentificación previa. Los depósitos más conocidos son los de carácter académico e institucional y tienen por objetivo organizar, archivar, preservar y difundir la producción intelectual resultante de la actividad investigadora de la entidad.

3) Confección de las clases del diseño necesarias para la herramienta de filtrado de datos (de ahora en adelante: herramienta de pre - procesamiento) de la Biblioteca del Congreso de Estados Unidos (LOC por sus siglas en inglés).
4) Confección de las clases del diseño necesarias para la herramienta AUCTORITAS.
5) Elaboración del modelo de implementación de la herramienta propuesta.
6) Implementación de la herramienta de pre - procesamiento.
7) Implementación de la herramienta.
8) Realización de pruebas y validaciones a la propuesta de solución.

En correspondencia con el objetivo propuesto, se utilizaron los siguientes **métodos teóricos**:

- El análisis y síntesis al descomponer el problema de investigación en elementos por separado y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución propuesta.
- El método histórico – lógico para analizar la evolución del proceso de control de autoridades, su perfeccionamiento a través del desarrollo de las bibliotecas y su modernización dado el impacto de las innovaciones tecnológicas; realizando un estudio crítico de trabajos anteriores para utilizarlos como punto de referencia y comparación de los resultados alcanzados.

Se utilizó además el siguiente **método empírico**:

- La entrevista para recopilar información, mediante una conversación profesional con especialistas del CIGED, sobre las deficiencias de los mecanismo para el control de autoridades en los proyectos desarrollados en dicho centro; así como para obtener retroalimentación sobre la propuesta de solución, una vez culminado el proceso de desarrollo.

El presente documento está estructurado de la siguiente forma:

Capítulo I: Fundamentación teórica del control de autoridades. En este capítulo se presenta la fundamentación teórica en la cual se basa el desarrollo de la presente investigación, que respalda el desarrollo de la solución propuesta. Se realiza además un estudio del estado del arte sobre el control de autoridades y se fundamenta la utilización de los lenguajes, las herramientas, el marco de trabajo y la metodología seleccionados para el desarrollo de la aplicación.

Capítulo II: Propuesta de solución. Este capítulo describe brevemente la propuesta de solución, incluyendo los artefactos generados por la misma; así como la arquitectura y los patrones de diseño aplicados en el código fuente.

Capítulo III: Herramienta de pre – procesamiento y AUCTORITAS. En este capítulo se detallan y exponen porciones del código fuente de las herramientas de pre – procesamiento, necesarias para el correcto funcionamiento de la aplicación AUCTORITAS; así como se describen e ilustran las relaciones entre clases mediante esquemas.

Capítulo IV: Análisis de los resultados. Este capítulo comprende las pruebas realizadas a la herramienta propuesta, para evaluar la calidad de la misma. Contiene un esbozo de las entrevistas realizadas, así como una breve descripción de los conjuntos de datos utilizados, lo cual permitió validar que la propuesta de solución cumple los objetivos trazados.

Capítulo I: Fundamentación teórica del control de autoridades

1.1 Introducción

En el presente capítulo se realiza un estudio de los conceptos básicos asociados al control de autoridades y se valoran las principales herramientas diseñadas para este fin, existentes actualmente en el mundo. De igual forma, se describen las tecnologías y herramientas utilizadas en el diseño e implementación de la aplicación propuesta, así como la metodología de desarrollo aplicada.

1.2 Conceptos asociados

A continuación se exponen algunos de los conceptos fundamentales, que se hace necesario definir para una correcta comprensión de la investigación.

1.2.1. Catalogación

La **catalogación** es un proceso donde se desarrollan un conjunto de operaciones de identificación y otras analíticas, de ordenamiento y localización que tributan, por una parte, a la descripción bibliográfica de los documentos y, por otra, a la elección de los puntos de acceso para representar ambos en un documento secundario que facilita la recuperación y posterior difusión de la información contenida en ellos (6).

1.2.2. Control de autoridades

El término control de autoridades no es nuevo, aunque hace pocos años se ha comenzado a utilizar y se le ha dado mucha relevancia dentro de la automatización de los catálogos. Las primeras concepciones sobre el término control de autoridades datan del siglo XIX. Sin embargo, no es hasta principios de los años 80 del siglo XX que comienza a ser asumido y estudiado como una actividad fundamental. Desde entonces hasta la actualidad este ha sido definido de muy diversas maneras (7).

El **control de autoridades** es la operación mediante la cual se unifican en una forma normalizada los puntos de acceso de los catálogos y se muestran las relaciones entre ellos. (7) Un **punto de acceso** (o autoridad) es un nombre, término, código, etc., bajo el cual puede buscarse, encontrarse e identificarse un registro bibliográfico o de autoridad (8).

1.2.3. Epígrafes de materia

Los **epígrafes** (o encabezamientos) **de materia** sirven para designar la temática de los documentos. Estos encabezamientos no proceden del uso que un autor determinado hace de la terminología, sino del uso común y aceptado en la lengua de la agencia catalográfica y en la disciplina de la que proceden. Es por ello que el término aceptado como encabezamiento de materia puede no aparecer en la obra que se cataloga y, sin embargo, es el proceso de indización el que marca que debemos traducir los términos utilizados en el documento, procedentes del lenguaje natural, al lenguaje de indización, que, por su propia naturaleza, es siempre un lenguaje artificial (9).

Dado el carácter artificial de cualquier lenguaje de indización, los términos que lo componen deben ser únicos y consistentes, normalizados y no arbitrarios, controlados y no libres. Si existe más de una forma de designar una determinada materia (sinónimos) se deberá escoger una sola de ellas y referenciar el resto como fórmulas alternativas. Por otra parte, si un mismo término sirve para designar diferentes materias (homonimias) deberán ser diferenciados de alguna forma (9).

1.2.4. Web semántica

La web semántica, también conocida como Web de los Datos y web 3.0, es realmente una extensión de la tecnología web actual, ofreciéndole mucha más utilidad. El objetivo es convertir la web en un repositorio de datos, más que en una simple colección de sitios y páginas web (10).

La web semántica es un área pujante en la confluencia de la Inteligencia Artificial y las tecnologías web, que propone nuevas técnicas y paradigmas para la representación del conocimiento que faciliten la localización, compartición e integración de recursos a través de la WWW. Estas nuevas técnicas se basan en la introducción de conocimiento semántico explícito, que describa y/o estructure la información y servicios disponibles (11).

La web semántica mantiene los principios que han hecho un éxito de la web actual, como son la descentralización, compartición, compatibilidad, o la apertura al crecimiento y uso no previstos de antemano (11).

1.2.5. Datos Enlazados Abiertos (LOD)

El concepto de **Datos Enlazados Abiertos** (en idioma inglés "*Linked Open Data*", de ahí las siglas LOD) fue propuesto originalmente por Tim Berners – Lee[8] y el mismo se refiere a datos publicados en la web, de manera tal que pueden ser leídos por una computadora y cuyo significado está definido explícitamente; se encuentran enlazados a otros conjuntos de datos internos y a su vez pueden ser enlazados desde conjuntos externos. Conceptualmente, los LOD se refieren a un conjunto de buenas prácticas para publicar y conectar datos estructurados en la web (12). Esta filosofía permite reutilizar y compartir los datos de autoridad de forma masiva y estable; además, ayuda en la detección de duplicados, la desambiguación terminológica y el enriquecimiento de los datos de autoridad.

En 2006 Berners – Lee definió cuatro reglas para la publicación de LOD (13):

1) usar Identificadores Uniformes de Recursos (URI por las siglas en inglés del término *Uniform Resource Identifiers*) identificando los recursos de forma unívoca;
2) usar URIs HTTP[9] para acceder a la información del recurso;
3) ofrecer información sobre los recursos usando el marco de trabajo para la descripción de recursos (RDF);
4) incluir enlaces a otros URIs, facilitando el vínculo entre distintos datos distribuidos en la web.

Estos principios están definidos como reglas, pero en realidad son más bien recomendaciones o buenas prácticas para el desarrollo de la web semántica (13).

1.2.6. Marco de trabajo para la descripción de recursos (RDF)

RDF es el marco de descripción de recursos para metadatos en la Web elaborado por el *World Wide Web Consortium*[10] (W3C). Se basa en la idea de declarar recursos usando la expresión en la forma sujeto – predicado – objeto. Esta expresión es conocida en la

[8] Sir Timothy "Tim" John Berners-Lee (Londres, Reino Unido, 8 de junio de 1955) es un científico de la computación británico, conocido por ser el padre de la Web. En octubre de 1994 fundó el Consorcio de la World Wide Web (W3C) con sede en el Instituto Tecnológico de Massachusetts, para supervisar y estandarizar el desarrollo de las tecnologías sobre las que se fundamenta la Web y que permiten el funcionamiento de Internet.
[9] Abreviatura de la forma inglesa Hypertext Trasnfer Protocol (Protocolo de Transferencia de Hipertextos) que se utiliza en algunas direcciones de Internet.
[10] El *World Wide Web Consortium*, abreviado W3C, es un consorcio internacional que produce recomendaciones para la *World Wide Web*.

terminología RDF como triple o triplete. Un triplete RDF contiene tres componentes, todos con referencia en un URI (13):

- Sujeto: una referencia URI o nodo, es el ente al cual se hace referencia;
- Predicado: es la propiedad o relación que se desea establecer acerca del sujeto;
- Objeto: es el valor de la propiedad o del otro recurso con el que se establece la relación.

El hecho de utilizar URIs para enlazar los datos, convierte la web semántica en una especie de gran base de datos que permite que las personas y las máquinas, puedan explorar la información referenciada e interconectada entre sí en la Web, lo que al mismo tiempo fomenta su crecimiento (13).

Esta estructura de enlaces forma un grafo dirigido y etiquetado, donde los bordes representan el enlace nombrado entre dos recursos, representados por los nodos del grafo. Esta vista de grafo es el modelo mental más fácil posible para RDF y se utiliza a menudo en explicaciones visuales fáciles de entender (14).

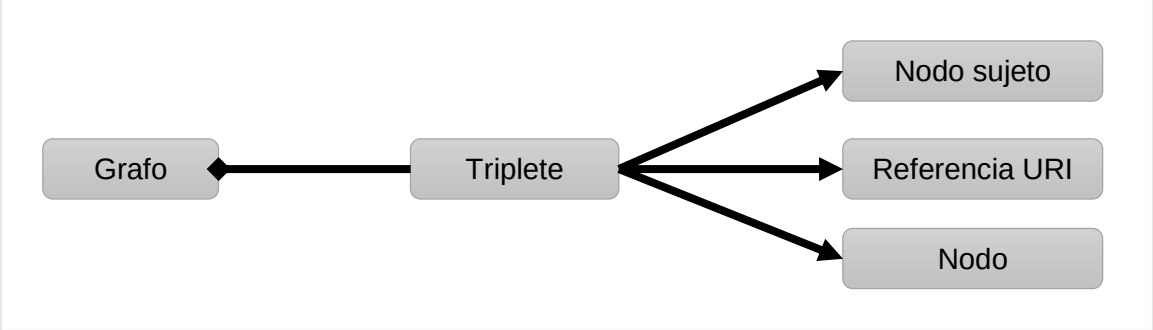

Ilustración 1 Esquema de grafo RDF

1.2.7. Servicios web

El consorcio W3C define los servicios web como sistemas de software, diseñados para soportar una interacción interoperable máquina a máquina sobre una red. Los servicios web suelen ser APIs[11] web, que pueden ser accedidas dentro de una red (principalmente Internet) y son ejecutados en el sistema que los aloja. La definición de servicios web propuesta alberga muchos tipos diferentes de sistemas, pero el caso común de uso se

[11] *Application Programming Interface*: (en español: interfaz de programación de aplicaciones) conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

refiere a clientes y servidores que se comunican mediante mensajes XML que siguen el estándar “Protocolo Simple de Acceso a Objetos”, más conocido por sus siglas SOAP (del inglés *Simple Object Access Protocol*). (15).

SOAP es un protocolo ligero para el intercambio de información en un entorno descentralizado y distribuido. Es un protocolo basado en XML que consta de tres partes: un envoltorio que define un marco de trabajo para describir lo que está en un mensaje y cómo procesarlo, un conjunto de reglas de codificación para expresar instancias de tipos de datos definidos por la aplicación, y una convención para la representación de llamadas y respuestas a procedimientos remotos. SOAP se puede utilizar en combinación con una variedad de otros protocolos (16).

En los últimos años se ha popularizado el estilo de arquitectura de software “Transferencia de Estado Representacional”, conocido como REST (del inglés *Representational State Transfer*). REST es un estilo de arquitectura para sistemas hipermedia distribuidos como la World Wide Web. El término fue introducido por Roy Fielding en su tesis doctoral y fue utilizado para guiar el desarrollo del protocolo HTTP (15). REST define un conjunto de principios de diseño, que incluyen: la funcionalidad y el estado de la aplicación son definidos como recursos; los recursos son únicamente direccionables; los recursos comparten una interfaz uniforme para la transferencia de estado entre el cliente y el recurso; el estilo de interacción cliente – servidor (17).

El concepto dominante de REST es la existencia de recursos, (fuentes de información específica) a las que se puede hacer referencia usando un identificador global. Los clientes pueden manipular estos recursos usando una interfaz estandarizada, mediante un protocolo estandarizado (como HTTP) e intercambiar representaciones de estos recursos (17).

1.3 Soluciones informáticas existentes

Las virtudes del control de autoridades han sido discutidas y reformuladas por décadas. Los catalogadores por al menos un siglo y medio, han documentado sus decisiones sobre cómo representar cada entidad bibliográfica en su catálogo, de manera única y autorizada (4).

1.3.1. Panorama internacional

Desde la década del 70 los estudiosos del tema han afirmado que el control de autoridades es la fase más costosa del proceso de catalogación y aún se buscan maneras de automatizar y simplificar el trabajo para reducir costos. Un paso agigantado en esta

dirección, lo constituye compartir el trabajo mediante un recurso en forma de archivo de autoridad entre muchas bibliotecas. Ejemplo de ello es el ahora famoso programa cooperativo *Name Authority Cooperative* (NACO). A través de este programa, los centros participantes aportan al Archivo de Nombres de Autoridad de la LOC, registros de autoridades con nombres personales, corporativos y jurisdiccionales y títulos (18).

Existen también numerosos ejemplos de archivos de autoridad compartidos nacional y regionalmente en el mundo, como el proyecto *Hong Kong Chinese Authority Name* (HKCAN). Este programa se estableció en 1999, liderado por las bibliotecas de la Universidad Lingnan y la Universidad China de Hong Kong; posibilitando la creación de una base de datos que provee acceso a registros de autoridades chinos. Esta base de datos cuenta con más de 180.000 registros de autoridad. Actualmente el grupo de trabajo de HKCAN tiene la misión de mejorar las operaciones de control de autoridades, haciéndolas más rápidas y baratas (19).

Otra iniciativa en este campo lo constituye AUTHORIS, una herramienta desarrollada en la Universidad de Granada, que aspira a facilitar el procesamiento de datos de autoridad de una manera estandarizada, basándose en los principios de los Datos Enlazados, centrada en el uso de reglas de aprendizaje automático y las posibilidades de los Datos Enlazados para operar registros de diversas organizaciones. Está basada en el sistema para la administración de contenidos Drupal y aprovecha sus facilidades para publicar datos en RDF (5).

AUTHORIS constituye un paso de avance con el fin de automatizar la generación de entradas de autoridad, sin embargo, se trata de una herramienta de consulta que no posibilita la interacción directa con un Sistema Integrado de Gestión Bibliotecario (SIGB), obligando al usuario a localizar la entrada de autoridad correspondiente y posteriormente trasladarla hacia su registro catalográfico (5).

La tendencia actual es realizar los proyectos de control de autoridades por medio de la cooperación entre varias bibliotecas, ya sea a nivel nacional o internacional. Este sistema reduce los costos y las tareas de control, porque todas las bibliotecas utilizan la información disponible y no se duplica el trabajo (7).

Un ejemplo concreto de esta tendencia es el Fichero Virtual Internacional de Autoridades (VIAF), un proyecto conjunto de varias bibliotecas, implementado y alojado por la *Online*

Computer Library Center[12] (OCLC) y apoyado por la Federación Internacional de Asociaciones de Bibliotecarios e Instituciones[13] (IFLA) y la LOC. Sus objetivos son disminuir el costo e incrementar la utilidad de los ficheros de autoridad de las bibliotecas, mediante la comparación y la correspondencia entre los ficheros de autoridades de las bibliotecas nacionales y poner esa información disponible en Internet. Este proyecto se basa en la filosofía de los LOD (7). El mismo ha representado grandes avances en la construcción y generación de entradas de autoridad, aunque no ha llegado a las principales instituciones de información a nivel mundial (2).

Actualmente la mayoría de los proyectos de control de autoridades que trabajan bajo la filosofía de datos abiertos enlazados, están implementados principalmente por las bibliotecas nacionales, quienes ponen a disposición de toda la comunidad en Internet sus datos de autoridades para que sean reutilizados (7).

1.3.2. Panorama nacional

La Biblioteca Nacional "José Martí" (BNJM), a partir de la implementación del formato UNIMARC[14], realiza proyectos encaminados a la organización de la información que respondan a las exigencias actuales. Como órgano cabecera del Sistema Nacional de Bibliotecas Públicas, cuenta con un grupo de trabajo encargado del control bibliográfico nacional.

Núñez[15] (20) define las autoridades en Cuba como *"*la forma normalizada de nombres personales, entidades corporativas, obra (títulos), marca, nombre de familia, tema como materia, acceso por lugar y formas genéricas o características físicas, que constituyen los puntos de acceso de un catálogo de biblioteca manual o automatizado y que se incluyan en

[12] El OCLC es un centro de bibliotecas por ordenador en línea que provee programas y servicios para ayudar a las bibliotecas a compartir y organizar su información.
[13] La IFLA o Federación Internacional de Asociaciones de Bibliotecarios e Instituciones (las siglas provienen de su denominación en inglés: *International Federation of Library Associations and Institutions*) es una organización mundial creada para proporcionar a bibliotecarios de todo el mundo un foro para intercambiar ideas, promoviendo la cooperación, la investigación y el desarrollo internacionales en todos los campos relacionados con la actividad bibliotecaria y la bibliotecología.
[14] En 1977 el Grupo de Trabajo sobre designadores de contenido de la Federación Internacional de Asociaciones de Bibliotecarios (IFLA), publicó el formato bibliográfico UNIMARC, que permite el intercambio de registros bibliográficos entre diferentes órganos bibliográficos nacionales, mediante un lenguaje común.
[15] Sonia Núñez Amaro: ingeniera en Geodesia y Cartografía. Master en Bibliotecología y Ciencias de la Información. Especialista del Departamento de Automatización de la Biblioteca Nacional José Martí. Investigadora agregada. Profesora asistente adjunta de la Facultad de Comunicación de la Universidad de La Habana.

el control bibliográfico nacional. Las formas que adopten estos puntos de acceso están en dependencia de las reglas aplicadas por la BNJM".

Según la autora, la aplicación del control de autoridades en Cuba puede implementarse a partir de la aplicación del formato UNIMARC/Autoridades, las directrices definidas por IFLA para la visualización y estructura de los registros: "Directrices para Registros de Autoridad y Referencias" (del inglés "*Guidelines for Authority Records and References"* – GARE) y "Guía para Autoridad de Materia y Entradas de Referencia" (del inglés "*Guideline for Subject Authority and Reference Entries*" – GSARE), y como órgano rector metodológico en el Sistema de Bibliotecas Públicas debe fungir la BNJM con su grupo coordinador de estudio de autoridades cubanas (7).

1.3.3. Consideraciones finales sobre las soluciones informáticas existentes

Al concluir el estudio sobre las soluciones informáticas existentes en el mundo para el control de autoridades, se constató que los sistemas actuales con este fin son básicos, la mayoría crean una base de datos interna de la cual consumen la información, lo cual no representa una alternativa factible pues se estaría limitando considerablemente el espacio de análisis. En el caso de Cuba, la aplicación del formato UNIMARC/Autoridades representa un costo enorme, ya que consta de normas muy complejas, que requieren demasiada información que no siempre está disponible.

La herramienta AUCTORITAS brindará varios servicios web, posibilitando la consulta de nombres de autores y entradas de materia, este último parámetro no es tomado en cuenta por las herramientas existentes actualmente. También se utilizarán URI para identificar los recursos a procesar, lo cual permite que diferentes tipos de identificadores (URI) de recursos puedan ser utilizados en el mismo contexto, incluso cuando difieran los mecanismos usados para acceder a esos recursos; del mismo modo, es posible la introducción de nuevos tipos de identificadores sin interferir con la manera en que los identificadores existentes son utilizados (21).

Para la clasificación de los epígrafes de materia se utilizarán vocabularios controlados específicos a cada rama de las ciencias, como el de la ACM[16] para las Ciencias de la Computación.

[16] ACM acrónimo de *Association for Computing Machinery* (Asociación de los Sistemas Informáticos). Fue fundada en 1947 como la primera sociedad científica y educativa acerca de la Computación.

1.4 Lenguajes de programación y de consulta

Un lenguaje de programación es un lenguaje artificial que puede ser usado para controlar el comportamiento de una máquina, especialmente una computadora. Estos se componen de un conjunto de reglas sintácticas y semánticas que permiten expresar instrucciones que luego serán interpretadas (22).

1.4.1. Java

Para el desarrollo de la aplicación se utilizó el lenguaje Java, el cual fue introducido como lenguaje de programación para computadoras en 1995. Según *Sun Microsystems*, (antigua compañía desarrolladora de este lenguaje), Java es un lenguaje dinámico, robusto y seguro, ya que se caracteriza por ser (23):

- Simple: incorpora de un modo estándar, sencillo y claro, muchos aspectos que en otros lenguajes son extensiones propiedad de empresas de software o fabricantes de ordenadores (hilos, ejecución remota, componentes, seguridad, acceso a bases de datos, etc.)
- De altas prestaciones: cualquier aplicación desarrollada en Java se apoya en un gran número de clases preexistentes. Algunas de ellas las ha podido hacer el propio usuario, otras pueden ser comerciales, pero siempre hay un número muy importante de clases que forman parte del propio lenguaje.
- Distribuido y multitarea: Java permite fácilmente el desarrollo de aplicaciones distribuidas, consistentes en crear aplicaciones capaces de conectarse a otros ordenadores y ejecutar tareas en varios ordenadores simultáneamente, repartiendo por lo tanto el trabajo. Aunque también otros lenguajes de programación permiten crear aplicaciones de este tipo, Java incorpora en su propio API[17] estas funcionalidades.
- Orientado a objetos: el lenguaje Java, basado en el paradigma de la programación orientada a objetos, permite utilizar objetos como elementos fundamentales en la

Publica varias revistas y periódicos científicos relacionados con la computación; patrocina conferencias en varias áreas del campo y otros eventos relacionados con las ciencias de la computación como por ejemplo el ACM *International Collegiate Programming Contest* (ACM – ICPC). Publica una extensiva Biblioteca digital y una referencia de la literatura de la computación.

[17] La interfaz de programación de aplicaciones, abreviada como API (del inglés: *Application Programming Interface*), es el conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas.

construcción de la solución. Un objeto es una abstracción de algún hecho o ente del mundo real, con atributos que representan sus características o propiedades, y métodos que emulan su comportamiento o actividad (23).

Además de las características expuestas anteriormente, se decide utilizar Java por las siguientes razones:

- Su arquitectura neutra, permitiendo que las aplicaciones desarrolladas en este lenguaje, no dependan del tipo de CPU[18] utilizado, sino que se ejecuten sobre la Máquina Virtual de Java, que es la encargada de interpretar el código neutro y convertirlo a código particular de la CPU utilizada.
- La posibilidad de añadirle numerosas bibliotecas (*libraries*) y drivers, que posibiliten su integración con otras tecnologías, tales como: jena-arq para la consulta de datos en formato RDF y postgresql para la conexión con el sistema de base de datos PostgreSQL.
- El dominio de los autores en el trabajo con el mismo.

1.4.2. SPARQL

SPARQL se puede utilizar para expresar consultas que permiten interrogar diversas fuentes de datos, si los datos se almacenan de forma nativa como RDF. SPARQL contiene las capacidades para la consulta de los patrones obligatorios y opcionales de grafo, junto con sus conjunciones y disyunciones. También soporta la ampliación o restricciones del ámbito de las consultas, indicando los grafos sobre los que se opera. Los resultados de las consultas SPARQL pueden ser conjuntos de resultados o grafos RDF (24). Este lenguaje, en su versión 1.1, fue utilizado para realizar consultas a los datos de vocabularios controlados almacenados en un servidor Virtuoso (véase 1.5.3.3 *Virtuoso Open Source*).

1.4.3. SQL

Structured Query Language (SQL) es el lenguaje estándar de definición, manipulación y control de bases de datos relacionales. Es un lenguaje declarativo y muy parecido al lenguaje natural; concretamente, se parece al inglés. Por estas razones, y como lenguaje

[18] Unidad de Procesamiento Central (más conocido por sus siglas en inglés CPU) es el hardware dentro de una computadora u otros dispositivos programables, que interpreta las instrucciones de un programa informático mediante la realización de las operaciones básicas aritméticas, lógicas y de entrada/salida del sistema.

estándar, el SQL es un lenguaje con el que se puede acceder a todos los sistemas relacionales comerciales (25). A continuación se listan algunas características proporcionadas por SQL (26):

- Comandos para inserción, borrado o modificación de datos.
- Capacidades aritméticas, es posible incluir operaciones aritméticas así como comparaciones.
- Asignación y comandos de impresión, es posible imprimir una relación construida por una consulta y asignar una relación calculada a un nombre de relación.
- Funciones agregadas, tales como promedio (average), suma (sum), máximo (max), etc., se pueden aplicar a las columnas de una relación para obtener una cantidad única.

Este lenguaje, fue utilizado para realizar consultas a los datos de autores personales almacenados en un servidor PostgreSQL. La elección del mismo se debe a las características mencionadas anteriormente, unido a que constituye un lenguaje estándar, utilizado por los principales sistemas gestores de bases de datos existentes actualmente, como el utilizado en la presente investigación: PostgreSQL.

1.5 Herramientas

1.5.1. Entorno de Desarrollo Integrado (IDE)

Un Entorno de Desarrollo Integrado (IDE por sus siglas en inglés) es una aplicación compuesta por un conjunto de herramientas útiles para un programador. Un IDE puede ser exclusivo para un lenguaje de programación o bien, poder utilizarse para varios. Suele consistir de un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

1.5.1.1. Eclipse

Es una plataforma de desarrollo, diseñada para ser extendida de forma indefinida a través de complementos. Fue concebida desde sus orígenes para convertirse en una plataforma de integración de herramientas de desarrollo. No tiene en mente un lenguaje específico, sino que es un IDE genérico, aunque goza de mucha popularidad entre la comunidad de desarrolladores del lenguaje Java usando el complemento *Java Development Tool* (JDT) que viene incluido en la distribución estándar del IDE. Eclipse es desarrollado por una

comunidad de código abierto. Además de proporcionar un IDE, Eclipse automatiza numerosas funciones que de otro modo, tendrían que ser programadas por los desarrolladores. Es barato de usar y hace que sea bastante fácil integrar sus propias herramientas con otras (27).

1.5.1.2. NetBeans

NetBeans es un entorno de desarrollo integrado de código abierto y gratuito, sin restricciones de uso. Proporciona a los desarrolladores múltiples herramientas que permiten crear aplicaciones empresariales, tanto para la Web como para dispositivos portátiles (28). Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación, además de que existe un número importante de módulos para extender su funcionamiento (29).

Se decide utilizar el IDE NetBeans dadas las siguientes razones:

- Es una herramienta de software libre muy completa y cuenta con amplia documentación.
- Es posible expandir sus funcionalidades básicas mediante la instalación de complementos (o *plugins*) y bibliotecas (*libraries*) para añadir funcionalidades a medida que se vayan necesitando, tales como los necesarios para realizar la conexión de dicho IDE con los gestores de bases de datos utilizados.
- Facilita la creación de servicios web REST, al generarlos de manera automática, dejando solo en manos de los desarrolladores la implementación de los métodos particulares de su proyecto.

1.5.2. Sistema Gestor de Bases de Datos (SGBD)

Un Sistema Gestor de Bases de Datos es una colección de programas que permiten a los usuarios crear y mantener una base de datos. También se define un SGBD como un sistema de propósito general, que facilita los procesos de definición, construcción y manipulación de la base de datos para distintas aplicaciones. La “definición” de la base de datos se refiere a especificar tipos de datos, estructuras y restricciones, la “construcción” consiste en almacenar datos y la “manipulación” comprende consultar, actualizar el diseño y generar informes (30).

1.5.2.1. SQL Server

Microsoft SQL Server es un Sistema de Gestión de Bases de Datos Relacionales (SGBDR) basado en el lenguaje Transact – SQL fabricado por Microsoft, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea. (31) Microsoft SQL Server ofrece las siguientes características:

- Facilidad de instalación, distribución y utilización.
- Posee una gran variedad de herramientas administrativas y de desarrollo, que permiten mejorar la capacidad de instalar, distribuir, administrar y utilizar SQL Server.
- Almacenamiento de datos.
- Incluye herramientas para extraer y analizar datos resumidos para el Proceso Analítico en Línea (OLAP). SQL Server incluye también herramientas para diseñar gráficamente las bases de datos y analizar los datos mediante preguntas en lenguaje normal.

Este sistema presenta las siguientes desventajas:

- Utiliza una gran cantidad de memoria RAM para la instalación y utilización del software.
- Solo permite alojar un máximo de 64 Gb.
- Requiere de un sistema operativo Windows.
- Es software privativo.

1.5.2.2. MySQL

MySQL es un Sistema de Gestión de Bases de Datos Relacionales. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. También tiene un amplio subconjunto del lenguaje SQL. Tiene como ventajas las siguientes (31):

- Conectividad segura.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Probado con un amplio rango de compiladores diferentes.
- Puede mezclar tablas de distintas bases de datos en la misma consulta.

A pesar de las excelentes características de MySQL, el mismo presenta las siguientes desventajas (31):

- Utiliza gran cantidad de memoria RAM para su instalación.
- Un gran porcentaje de sus utilidades no se encuentran documentadas.
- No es intuitivo como otros programas.

1.5.2.3. PostgreSQL

PostgreSQL es un Sistema de Gestión de Bases de Datos Objeto – Relacional, distribuido bajo licencia BSD[19] y con su código fuente disponible libremente. Tiene soporte total para transacciones, disparadores, vistas, procedimientos almacenados y almacenamiento de objetos de gran tamaño. Se destaca en ejecutar consultas complejas, consultas sobre vistas, subconsultas y uniones de gran tamaño. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Presenta un rendimiento elevado con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema (31).

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema; un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Algunas de las características de este sistema son (32):

- Es una base de datos 100% ACID[20].
- Posee integridad referencial.
- Realiza copias de seguridad en caliente.
- Es Unicode[21].
- Permite múltiples métodos de autentificación.
- Contiene APIs para programar en C/C++, Java, .Net, Perl, Phyton, Ruby, Tcl, ODBC, PHP, Lisp, Scheme, QT y muchos otros.
- Además de sus ofertas de soporte, cuenta con una importante comunidad de profesionales y entusiastas de PostgreSQL, de los que los centros de desarrollos pueden obtener beneficios y contribuir.

[19] Licencia Berkeley Software Distribution (licencia BSD) licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution). Es una licencia de software libre permisiva como la licencia de OpenSSL o la MIT License. Esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre.
[20] ACID es un acrónimo de los conceptos en inglés: atomicidad, consistencia, aislamiento y durabilidad en español.
[21] Unicode es un estándar de codificación de caracteres. Define la forma en que caracteres individuales se representan en archivos de texto, páginas web y otros tipos de documentos.

- Se encuentra disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows de 32 y 64 bit.

Este SGBD posee las siguientes desventajas:

- Es fácil de vulnerar sin protección adecuada.
- Reducida cantidad de tipos de datos.

Se decide utilizar PostgreSQL por las siguientes razones:

- Ser software libre, lo cual exime del pago de licencias para su uso.
- Su buen rendimiento al ejecutar consultas complejas y trabajar con grandes cantidades de datos, teniendo en cuenta que la herramienta desarrollada precisa la realización de consultas a conjuntos de datos voluminosos.
- Contar con una amplia documentación de cada una de sus funcionalidades, disponible lo mismo en repositorios de la Universidad, que en foros y sitios tanto nacionales como internacionales.
- Provee extensibilidad a la propuesta de solución, ya que no se limita solamente a los datos de la LOC, sino que es posible que una nueva herramienta de pre – procesamiento también inserte datos en dicha base de datos, incrementando la cantidad de registros ya presentas en esta.
- Dada la experiencia de los autores con el mismo.

1.5.3. Almacén de tripletas RDF

Un almacén de tripletas RDF ("*triple store*" en idioma inglés) es un Sistema Gestor de Bases de Datos diseñado para el almacenamiento y consulta de datos en formato RDF. Estos sistemas proveen un mecanismo para el almacenamiento persistente y el acceso a grafos RDF.

1.5.3.1. Apache Jena Fuseki

Apache Jena Fuseki es un servidor SPARQL, puede funcionar como un servicio del sistema operativo, como una aplicación web Java o como un servidor independiente. Provee una interfaz de usuario para el monitoreo y administración del servidor, así como proporciona los protocolos del lenguaje SPARQL 1.1 para la consulta y actualización y está estrechamente integrado con TDB, un componente de Jena para el almacenamiento y

consulta de RDF, lo cual proporciona una capa de almacenamiento transaccional persistente (33).

1.5.3.2. Sesame

Sesame está orientado al procesamiento de datos RDF, soporta almacenamiento basado en memoria y basado en disco. Su primera versión fue liberada en el año 2004 como una herramienta de código abierto, implementada sobre Java y multiplataforma. Soporta los lenguajes de programación Java, PHP y Phyton (34).

1.5.3.3. Virtuoso Open Source

Virtuoso es un servidor de datos multi – modelo de grado empresarial. Ofrece una plataforma para la administración, acceso e integración de los datos. La arquitectura híbrida única de Virtuoso, permite funciones tradicionales dentro de un mismo producto, lo cual le permite dominar las siguientes áreas (35):

- Administración de datos de tablas SQL relacional.
- Administración de datos de propiedades de grafos relacionales RDF.
- Administración de contenidos.
- Servicios de archivo para web y otros documentos.
- Servidor web de aplicaciones.

Se decide utilizar Virtuoso por las siguientes razones:

- Su liberación inicial se produjo en el año 1998, por lo que 17 años después de su lanzamiento, constituye una herramienta consolidada que posibilita un análisis robusto de datos.
- Ofrece una gran compatibilidad con diferentes sistemas operativos y lenguajes de programación, en comparación con los demás almacenes de tripletas RDF mencionados anteriormente (véase en Anexos: Tabla 7 Comparación de diferentes almacenes de tripletas RDF).
- Tener una distribución libre, lo cual garantiza su uso gratuito en sistemas de código abierto.

1.5.4. Herramientas para la realización de pruebas unitarias

1.5.4.1. JUnit

JUnit es una herramienta para Java, desarrollada por Erich Gamma[22] and Kent Beck[23], adoptada y apoyada por grupos partidarios de la programación extrema, la cual, entre otras cosas, sigue una política de primero probar y luego codificar. Esta y todas las herramientas descendientes de JUnit consisten de una serie de clases que auxilian en la preparación y codificación de casos de prueba y algunos mecanismos auxiliares, que en conjunto permiten ejecutar y verificar el cumplimiento de los casos de prueba. Además provee una interfaz que permite automatizar la ejecución de grupos de casos de prueba. JUnit ha tenido mucho éxito, por lo que está incorporado en varios IDEs, tales como Eclipse y NetBeans (36).

1.5.4.2. TestNG

TestNG es un *framework* creado por el Cédric Beust para probar software escrito en el lenguaje Java. El objetivo del mismo es cubrir una amplia gama de necesidades existentes en los diferentes tipos de pruebas, desde las unitarias hasta las de integración. Fue creada basándose en JUnit y NUnit (para .NET), pero introduciendo nuevas funcionalidades que la hacen más poderosa y fácil de usar (37).

Se decide utilizar JUnit por las siguientes razones:

- Permite definir los casos de prueba para ser ejecutados en cualquier momento, por lo cual es sencillo, luego de realizar modificaciones al programa, comprobar que los cambios no introdujeron nuevos errores.
- Es una herramienta bastante sencilla, integrada en el IDE seleccionado para el desarrollo de la propuesta de solución.

[22] Erich Gamma (Zúrich, 1961) es un informático suizo, actualmente empleado de Microsoft tras pasar por IBM Rational software y ser director del Object Technology International Zurich Lab en Zúrich, liderando el desarrollo de la plataforma Eclipse.
[23] Kent Beck es ingeniero de software estadounidense, uno de los creadores de la metodología de desarrollo de software de Programación Extrema (XP) y el desarrollo guiado por pruebas (*Test – Driven Development* o TDD), también llamados metodología ágil. Beck fue uno de los 17 firmantes originales del Manifiesto Ágil en 2001.

1.5.5. Herramientas para la realización de pruebas de carga

1.5.5.1. Rational Performance Tester

Rational Performance Tester es una solución de pruebas de rendimiento que valida la escalabilidad de aplicaciones web y de servidor. Permite identificar la presencia y la causa de los cuellos de botella de rendimiento del sistema, además de reducir la complejidad de las pruebas de carga, así como ejecutar de forma rápida pruebas de rendimiento que analizan el impacto de la carga sobre aplicaciones. Es un software propietario desarrollado por la corporación IBM (38).

1.5.5.2. LoadUI

LoadUI es un software de prueba de carga, destinado principalmente a servicios web. Entre sus características se encuentran la integración con la herramienta de pruebas funcionales SoapUI y la distribución de carga, al probar diferentes servicios web. Es posible agregarle funcionalidades en tiempo de ejecución utilizando *plugins* creados por desarrolladores de terceros (39). SoapUI es una aplicación de código abierto para la realización de pruebas a servicios web basados en la Arquitectura Orientada a Servicios (SOA) y basados en REST. Entre sus funcionalidades se encuentran la inspección, invocación, desarrollo y simulación de servicios web (40).

Se decide utilizar la herramienta LoadUI por las siguientes razones:

- Está orientado específicamente a la realización de pruebas a servicios web, los cuales constituyen la vía utilizada por la propuesta de solución, para el intercambio de información con las aplicaciones externas.
- Se integra con la herramienta SoapUI, orientada a la realización de pruebas a servicios web basados en REST, el cual es el estilo de arquitectura seleccionado, para la implementación de los servicios brindados por la propuesta de solución.

1.6 Marco de trabajo

Un marco de trabajo (o *framework*) es una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, se puede considerar como una aplicación genérica incompleta y configurable a la cual es posible añadirle las últimas piezas para construir una aplicación concreta. Los objetivos principales que persigue un *framework* son (41):

- Acelerar el proceso de desarrollo de software.
- Reutilizar código ya existente.
- Promover buenas prácticas de desarrollo como el uso de patrones.

1.6.1. Sesame

Sesame es un marco de trabajo para Java que permite el procesamiento y manejo de datos RDF, lo cual incluye crear, analizar gramaticalmente, almacenar, realizar inferencia y consultar dichos datos. Ofrece una API fácil de usar que puede ser conectada con los principales almacenes de tripletas RDF (42).

1.6.2. Apache Jena

Apache Jena (o simplemente Jena) es un marco de trabajo de Java, libre y de código abierto para la construcción de aplicaciones de web semántica y LOD. Se compone de diferentes APIs que interactúan entre sí para procesar los datos RDF. Las mismas permiten la creación de clases objetos para representar grafos, recursos, propiedades y literales. Los grafos son representaciones de los recursos modelados bajo el estándar RDF, los recursos (en el marco de la presente investigación) constituyen los datos bibliográficos que consume la herramienta propuesta como solución, las propiedades son características de los recursos y los literales se refieren al valor de cada propiedad (43). Provee un ambiente de programación para RDF, RDFS[24] y OWL[25], SPARQL e incluye un motor de inferencia basado en reglas (44).

Se decide utilizar Jena dado:

- Su facilidad de uso, que permite un aprendizaje avanzado en corto tiempo de las técnicas necesarias para manipular hábilmente grafos RDF.
- Su completa integración con el lenguaje Java, lo que permite que el procesamiento de datos se realice de manera fluida y rápida.

[24] RDFS o RDF Schema o Esquema RDF es una extensión semántica de RDF. Un lenguaje primitivo de ontologías que proporciona los elementos básicos para la descripción de vocabularios.
[25] OWL es el acrónimo del inglés *Web Ontology Language*, un lenguaje de marcado para publicar y compartir datos usando ontologías en la WWW. OWL tiene como objetivo facilitar un modelo de marcado construido sobre RDF y codificado en XML.

1.7 Metodología de desarrollo

Una metodología es un proceso formalizado o conjunto de buenas prácticas para crear software. Incluye: un conjunto de reglas a seguir, un conjunto de convenciones que la organización decide seguir y un acercamiento ingenieril y sistemático para organizar proyectos de software. (45)

En el ámbito de la Ingeniería de Software, la evolución de las metodologías de desarrollo de software ha llevado a la aparición de las denominadas metodologías ágiles, las cuales están destinadas a romper con la rigidez de las tradicionales, caracterizadas por la extensa documentación del proceso de desarrollo y por la inflexibilidad ante los cambios (46).

Existen diversas técnicas que, mediante la identificación y evaluación de las características del proyecto, permiten definir si se debe seguir un enfoque ágil o robusto para el desarrollo del mismo. Entre estas técnicas se encuentra la matriz de Boehm – Turner, la cual permite medir las características de un proyecto con el objetivo de determinar su idoneidad para un enfoque ágil. Estas características se evalúan en base a cinco factores críticos (descritos en Anexos: Tabla #2) para el desarrollo de un proyecto (según sus autores), los cuales se ubican en un gráfico radial (matriz). Las puntuaciones hacia el centro indican un buen ajuste para un enfoque ágil, mientras que las puntuaciones hacia el exterior sugieren un enfoque más tradicional (48).

En Anexos: Ilustración #2, se muestra la matriz resultante de aplicar los cinco factores descritos por Boehm y Turner al desarrollo de la presente investigación. En base al resultado observado, se decide considerar solamente metodologías ágiles para el desarrollo de la presente investigación.

1.7.1. SCRUM

Esta metodología centra su atención en las actividades de gerencia basándose principalmente en una planificación adaptativa y en el desarrollo incremental del software con entregas funcionales en breves períodos de tiempo. Frente a escenarios y requerimientos cambiantes, contar con una herramienta que permita simular la gestión de proyectos de desarrollo de software con SCRUM, representa una alternativa interesante para que los administradores puedan evaluar el impacto de sus decisiones sobre la gestión en el desarrollo del proyecto, sin influir o poner en riesgo el proyecto real y sus recursos. (49)

1.7.2. Agile Unified Process (AUP)

El AUP es un acercamiento aerodinámico (referente a la forma de las gráficas del esquema de trabajo de la metodología, mostrado en la Ilustración 1) al desarrollo del software basado en el Proceso Unificado Rational de IBM (RUP), basado en disciplinas y entregables incrementales con el tiempo. El ciclo de vida en proyectos grandes es serial mientras que en los pequeños es iterativo. Las disciplinas de AUP son: (50)

- Modelado
- Implementación
- Prueba
- Despliegue
- Administración de la configuración
- Administración o gerencia del Proyecto
- Entorno

En Anexos: Ilustración #3, se muestra el esquema de trabajo AUP.

1.7.3. Programación Extrema (XP)

La Programación Extrema (XP) reúne un conjunto de prácticas sencillas ya conocidas, pero que en este caso son llevadas a cabo conjuntamente y en forma extrema. Debido a esto, la gestión de proyectos de desarrollo de software se torna algo impredecible y compleja y es difícil anticipar por sus administradores, los efectos que tiene sobre la marcha del mismo, la aplicación de las diferentes prácticas de XP (46). Bajo esta metodología, cada programador define sus pruebas cuando escribe su código de producción. Las pruebas se acoplan en el proceso de integración continua y construcción lo que rinde una plataforma altamente estable para el desarrollo futuro.

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico (47).

Se decide utilizar la metodología XP ya que:

- No se contaba con mucho tiempo de desarrollo, el cual debía ser aprovechado al máximo posible en el desarrollo de la aplicación y no en la redacción y construcción de numerosos productos de trabajo.
- Se ajusta a las necesidades del proyecto, al proveer técnicas sencillas para la representación de la aplicación; como también se ajusta a escenarios con requisitos cambiantes. Los mismos cambiaron en dos ocasiones durante el desarrollo de la investigación.
- Sitúa la comprobación como el fundamento del desarrollo, permitiendo que las pruebas sean creadas en el propio IDE, con el *framework* de pruebas JUnit, justo después de escribir el código.
- Promueve la programación en dúos, lo cual conlleva ventajas implícitas como son: menor tasa de errores y mayor satisfacción de los programadores.

1.8 Conclusiones del capítulo

En este capítulo se plasmaron un conjunto de conceptos relevantes para la investigación, los cuales permitieron establecer la base teórica necesaria para el comienzo de la misma, así como, facilitaron la definición de la metodología y las herramientas utilizadas para el desarrollo de la propuesta de solución. Como resultado del análisis de diferentes vías y sistemas para realizar el control de autoridades en Cuba y el mundo, se constató que los mismos no presentan todas las características necesarias para ofrecer una solución completa al problema planteado.

Capítulo II: Propuesta de solución

2.1 Introducción

En el capítulo actual se describe la herramienta propuesta para dar solución a la situación problemática existente, se plasman los artefactos ingenieriles definidos por la metodología de desarrollo establecida y se detallan los patrones de diseño empleados.

2.2 Requisitos funcionales

La aplicación contará con los siguientes requisitos funcionales:

RF 1) Filtrar datos de autores personales.
RF 2) Ver detalles de autor personal.
RF 3) Buscar término autorizado.
RF 4) Listar los vocabularios controlados disponibles.

2.3 Breve descripción de la propuesta de solución

La propuesta de solución se nombra AUCTORITAS; consiste en una aplicación que proveerá servicios web a aplicaciones externas, para la realización de consultas a datos de vocabularios controlados y de autores personales. Los servicios web permitirán el intercambio de datos entre AUCTORITAS y las aplicaciones externas, independientemente de que hayan sido desarrolladas en un lenguaje de programación o plataforma diferente (véase 3.3.1 Servicios web ofrecidos por AUCTORITAS).

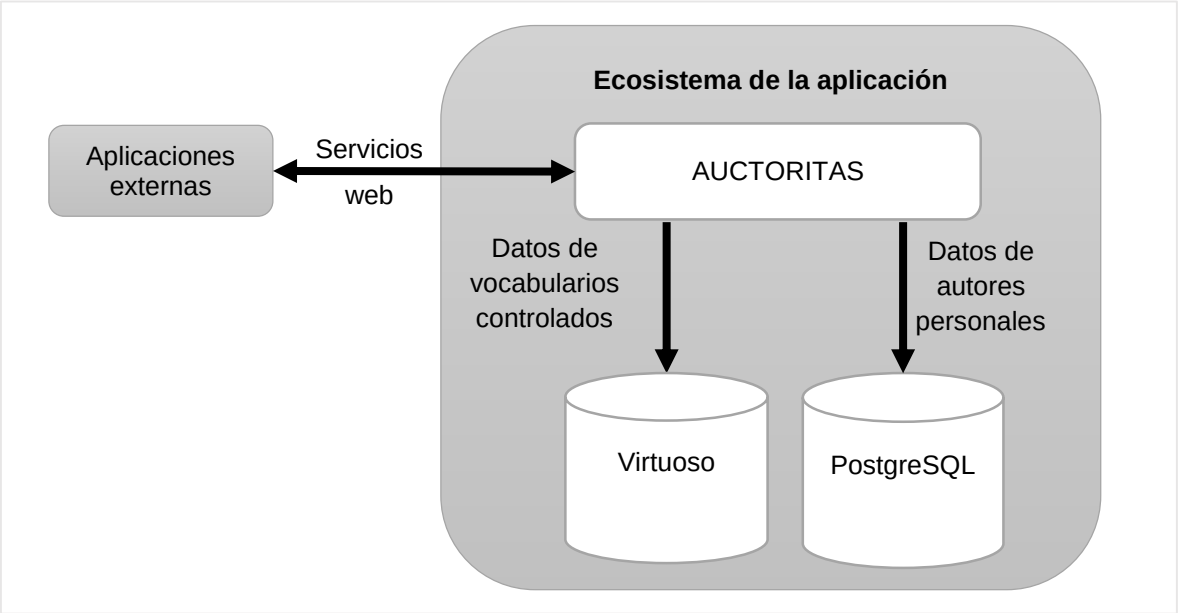

Ilustración 2 Flujo de información de AUCTORITAS

Los datos de vocabularios controlados están disponibles libremente en Internet en formato RDF; por lo que AUCTORITAS almacena los mismos en un servidor Virtuoso orientado a la persistencia y consulta de datos en este formato. Dicha consulta se hace posible mediante dos servicios web: “Listado de vocabularios controlados” y “Término autorizado”, este último tiene la intención de lograr la normalización terminológica de los materiales bibliográficos, para facilitar su registro y posterior recuperación. Para más información véanse los subepígrafes 3.3.1.1. Listado de vocabularios controlados y 3.3.1.2. Término autorizado.

El servicio web “Datos de autor personal” busca estandarizar el registro de la información bibliográfica de autores personales, lo cual viabiliza la posterior recuperación de dicha información y permite registrar nuevas obras y publicaciones de manera correcta (para más información véase 3.3.1.3 Datos de autor personal). Para lograr esto, se posibilita consultar datos de autores personales, los cuales son extraídos de ficheros de autoridad que en ocasiones no presentan la calidad suficiente para ser analizados correctamente (véase 3.2.2. Filtrado de las entradas bibliográficas). Por esta razón, se hace necesario depurar los mismos antes de ser almacenados en una base de datos PostgreSQL.

Con este propósito, AUCTORITAS se apoya en una herramienta capaz de filtrar (extraer los datos relevantes para la aplicación) y procesar (ordenar y almacenar los datos en una base de datos relacional) información bibliográfica, independientemente de su fuente de

origen, siempre que se encuentre en formato RDF. El funcionamiento de esta herramienta se detalla en el subepígrafe 3.2 Herramienta de pre – procesamiento.

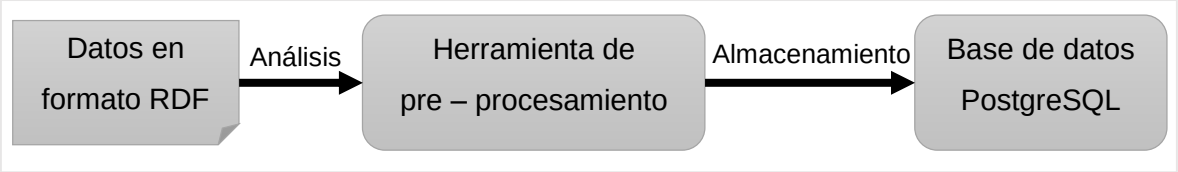

Ilustración 3 Flujo de información de la herramienta de pre - procesamiento

El despliegue de AUCTORITAS pretende prevenir el problema de la duplicidad de información, existente actualmente en diversos sistemas de gestión bibliotecarios y para la gestión de repositorios digitales de acceso abierto.

2.4 Patrón arquitectónico

Un patrón arquitectónico (también conocido como estilo arquitectural) se puede entender como un conjunto de principios que definen a alto nivel un aspecto de la aplicación. Un patrón arquitectónico viene definido por un conjunto de componentes, un conjunto de conexiones entre dichos componentes y un conjunto de restricciones sobre cómo se comunican dos componentes cualesquiera conectados. (51)

Un patrón de arquitectura es una plantilla para el diseño de aplicaciones. Estos especifican las propiedades generales a la estructura del sistema, y repercuten en la arquitectura de sus subsistemas. La selección de un patrón de arquitectura es por lo tanto una decisión fundamental al desarrollar un sistema de software (51).

La propuesta de solución presenta una arquitectura en dos capas, en la cual se detallan los principales subsistemas de software:

- **Servicios web:** constituye la interfaz de comunicación con las aplicaciones externas que soliciten información a la herramienta AUCTORITAS y a través de la cual AUCTORITAS devuelve la respuesta correspondiente.
- **Acceso a datos:** esta capa contiene la lógica de comunicación con otros sistemas que llevan a cabo tareas por la aplicación, como el almacenamiento persistente de información.

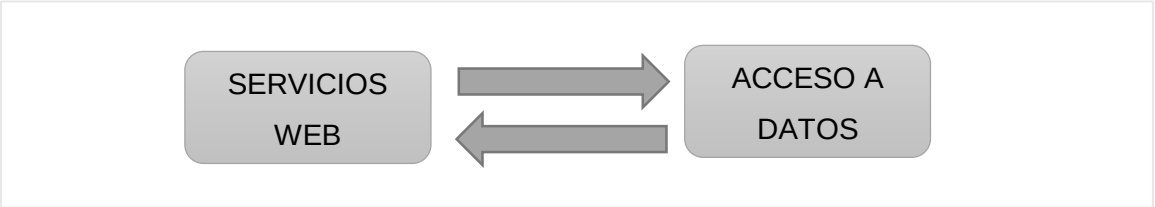

Ilustración 4 Arquitectura de la aplicación

Se escogió la arquitectura en dos capas pues posibilita considerar una capa como un todo, sin considerar las otras, minimizando las dependencias entre cada una de ellas y posibilitando estandarizar los servicios. Luego de construida una capa, puede ser utilizada por diversos servicios de mayor nivel.

2.5 Patrones de diseño

Un patrón de diseño describe las clases y objetos que se comunicarán entre sí de manera que puedan resolver un problema general de diseño en un contexto particular. (52) Un patrón nomina, abstrae e identifica los aspectos clave de una estructura de diseño común, lo que los hace útiles para crear un diseño orientado a objetos reutilizable. Identifica las clases e instancias participantes, sus roles y colaboraciones y la distribución de responsabilidades. (53)

Los patrones de diseño tienen como objetivo aumentar la eficiencia, disminuir el esfuerzo de mantenimiento y reducir el tiempo de desarrollo. (52)

2.5.1. Patrones GRASP

GRASP es un acrónimo de *General Responsibility Assignment Software Patterns* (Patrones Generales de Software para Asignar Responsabilidades). El nombre se eligió para sugerir la importancia de aprehender (*grasping* en inglés) estos principios para diseñar con éxito el software orientado a objetos. Estos patrones describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones. (54)

En la propuesta de solución constan los siguientes patrones GRASP:

2.5.1.1. Bajo acoplamiento

El patrón bajo acoplamiento impulsa la asignación de responsabilidades, de manera que su localización no incremente el acoplamiento, hasta un nivel que conlleve a los resultados negativos que puede producir un acoplamiento alto. El bajo acoplamiento soporta el diseño

de clases que son más independientes, lo que reduce el impacto del cambio. En otras palabras, debe haber poca dependencia entre las clases, de manera que se puedan extraer porciones de código de un modo independiente y reutilizarlas en otro proyecto (55).

Este patrón se evidencia en las clases que implementan los servicios web, pues tienen una responsabilidad concreta, que no se afecta si otra clase deja de existir.

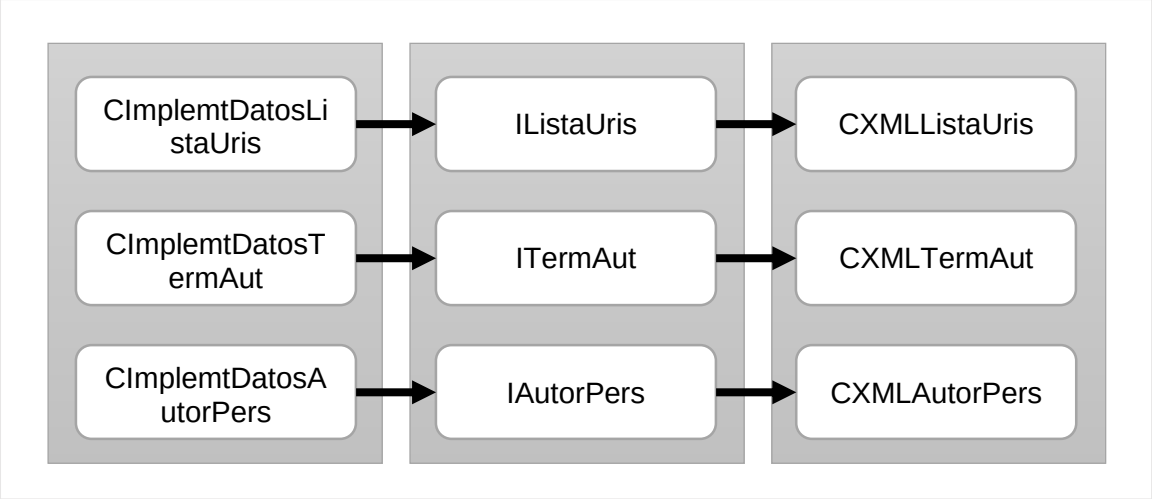

Ilustración 5 Bajo acoplamiento en clases que implementan los servicios web

2.5.1.2. Alta cohesión

La alta cohesión es un principio a tener en mente durante todas las decisiones de diseño. Este patrón incrementa la claridad y facilita la comprensión del diseño, simplificando el mantenimiento y las mejoras y a menudo soporta bajo acoplamiento (56).

Se evidencia entre las clases encargadas de crear los servicios web, acceder a los datos y ejecutar los mismos; la funcionalidad de cada una tributa a la correcta ejecución de cada servicio.

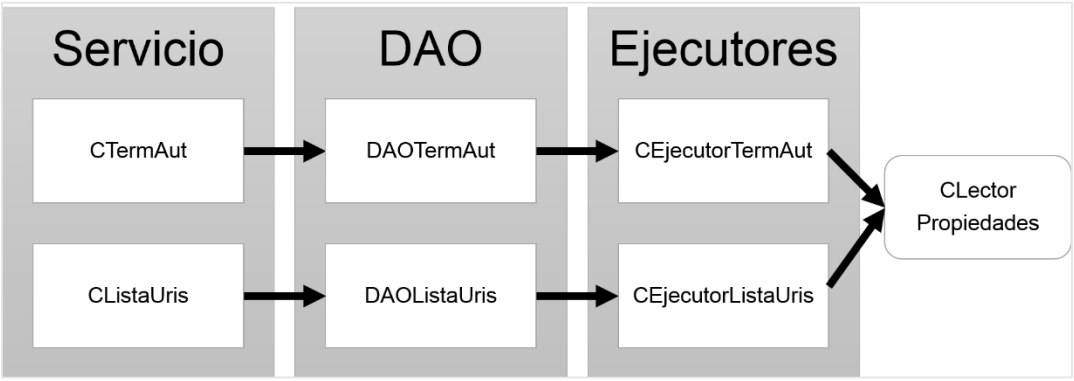

Ilustración 6 Alta cohesión evidenciada en clases que implementan los servicios web

2.5.1.3. Experto

El patrón experto se basa en asignar una responsabilidad al experto en información: la clase que tiene la información necesaria para realizar la responsabilidad. De esta manera se logra un sistema más fácil de entender, mantener y ampliar y existen más oportunidades para reutilizar componentes en futuras aplicaciones (56). Este patrón se observa en la clase `CConeccion`, que posee toda la información para establecer una conexión a la base de datos, dicha información no se encuentra en ninguna otra clase.

2.5.2. Patrones GoF

GoF es la sigla de *Gang of Four* (Banda de Cuatro). Estos patrones son considerados los fundadores de los demás patrones de diseño, su nombre se debe a que proceden del libro "*Design Patterns: Elements of Reusable Object – Oriented Software*", escrito por cuatro autores: Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Se clasifican según su propósito en tres grupos: creacionales, estructurales y de comportamiento. (54)

La aplicación presenta los siguientes patrones GoF:

2.5.2.1. Instancia única (Singleton)

Este patrón garantiza que exista una única instancia para una clase y define un mecanismo para acceder a dicha instancia (57). El mismo se aprecia en la clase `CLectorPropiedades` y `CProcesador`, en la primera se definen un conjunto de propiedades que son instanciadas una única vez en la segunda.

```
public class LectorPropiedades {

    private String sDirectorio;
    private String virtuoso;
    private String ubic;
    private String user;
    private String pass;
    private static LectorPropiedades instancia = null;

    private  LectorPropiedades() {
        leerArchivo();
    }

    public static LectorPropiedades getInstancia() {
        if (null == instancia)
            instancia = new LectorPropiedades();
        return instancia;
    }
```

Ilustración 7 Clase "LectorPropiedades"

2.5.2.2. Visitante (Visitor)

Permite definir nuevos métodos sobre una jerarquía de clases sin modificar las clases sobre las que opera (58). Este patrón se observa en las clases `CProcesador` y `CConeccion`; la primera controla el proceso de filtrado de datos y su inserción en una base de datos y la segunda contiene los parámetros de conexión a la base de datos; en ambas se pueden realizar modificaciones que no alteran el funcionamiento de las clases que dependen de ellas.

2.5.2.3. Cadena de responsabilidad (Chain of responsability)

A través de este patrón se establece la línea que deben llevar los mensajes para que los objetos realicen la tarea indicada (58). El mismo se aprecia desde que comienza el pre – procesamiento de los datos, cada clase realiza su función y transmite el resultado de su ejecución a la próxima clase.

2.6 Diseño de la propuesta de solución

La metodología XP prescinde de numerosos diagramas en lenguaje UML para la representación del sistema; en cambio define técnicas que consisten básicamente en

Historias de usuario, un plan de iteraciones y tarjetas CRC. Dichos artefactos son descritos a continuación.

2.6.1. Historias de usuario

Las **historias de usuarios** (HU) sustituyen a los documentos de especificación funcional y los **casos de uso** propuestos por otras metodologías. Estas “historias” son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar. La diferencia más importante entre estas historias y los tradicionales documentos de especificación funcional se encuentra en el nivel de detalle requerido. Las historias de usuario deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Cuando llegue el momento de la implementación, los desarrolladores dialogarán directamente con el cliente para obtener todos los detalles necesarios (59).

Las historias de usuarios deben poder ser programadas en un tiempo entre una y tres semanas. Si la estimación es superior a tres semanas, debe ser dividida en dos o más historias. Si es menos de una semana, se debe combinar con otra historia. (59) Cada historia se corresponde con un requisito funcional de la aplicación y las mismas contienen los elementos que se describen a continuación:

- **Número:** identificador de la HU.
- **Nombre:** nombre que identifica a la HU.
- **Usuario:** involucrados en la ejecución de la HU.
- **Iteración asignada:** iteración en que se implementará la HU.
- **Prioridad en el negocio:** prioridad de la HU respecto al resto de las HU, puede ser: alta, media o baja.
- **Riesgo en desarrollo:** riesgo en la implementación de la HU, puede ser: alto, medio o bajo.
- **Puntos estimados:** estima el esfuerzo asociado a la implementación de la HU. Un punto equivale a una semana ideal de programación, generalmente de 1 a 3 puntos.
- **Puntos reales:** resultado del esfuerzo asociado a la implementación de la HU. Un punto equivale a una semana ideal de programación, generalmente de 1 a 3 puntos.
- **Descripción:** descripción sintetizada de la HU.

Las HU se muestran en Anexos: Historias de usuarios.

2.6.2. Plan de iteraciones

Las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido. Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. Cada historia de usuario se traduce en tareas específicas de programación. Asimismo, para cada historia de usuario se establecen las pruebas de aceptación. Estas pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes, para verificar que subsiguientes iteraciones no han afectado a las anteriores. Las pruebas de aceptación que hayan fallado en el ciclo anterior son analizadas para evaluar su corrección, así como para prever que no vuelvan a ocurrir (59).

Tabla 1 Estimación del esfuerzo por Historias de Usuario

Historia de usuario	Puntos de estimación
Filtrar datos de autores personales	3
Ver detalles de autor personal	2
Buscar término autorizado	4
Listar vocabularios controlados	3

Tabla 2 Plan de duración de iteraciones

Iteración	Orden de las Historias de Usuario a implementar	Semanas
1	Filtrar datos de autores personales	3
2	Ver detalles de autor personal	2
3	Buscar término autorizado	4
4	Listar vocabularios controlados	3

Tabla 3 Plan de entrega

Iteración	Orden de las historias de usuario a implementar	Estimación (semana)	Fecha
1	Filtrar datos de autores personales	3	26/2 – 18/3
2	Ver detalles de autor personal	2	19/3 – 1/4
3	Buscar término autorizado	4	2/4 – 29/4

4	Listar vocabularios controlados	3	30/4 – 20/5

2.6.3. Tarjetas CRC

En las tarjetas Clase, Responsabilidad y Colaboración (CRC) se registran los nombres de las clases, sus responsabilidades y con qué otras clases colaboran; cada tarjeta representa una clase en el sistema. Las tarjetas CRC ayudan a identificar las clases que participan del diseño del sistema, obtener las responsabilidades que debe cumplir cada clase y establecer cómo colabora una clase con otras clases para cumplir con sus responsabilidades. (60)

Las restantes tarjetas CRC se muestran en los anexos: Tarjetas CRC.

2.6.4. Tarjetas de tarea

Las tarjetas de tarea (del inglés "*task cards*"), brindan información de las distintas tareas que componen a las historias de usuario. No existe una plantilla para la confección de las mismas, aunque se recomienda que contengan la siguiente información: el número de la tarea, la historia de usuario a la que hace referencia, el nombre de la tarea, el tipo de tarea (si es de desarrollo, de corrección, de mejora, o algún otro tipo de tarea específico), los puntos estimados, una fecha de inicio, una fecha de finalización, el programador responsable, y una breve descripción de la misma (en que consiste la tarea) (61).

A continuación se muestran se muestran las Tarjetas de tarea #1, #2 y #3, correspondientes a la Historia de Usuario #1.

Las Tarjetas de tarea se muestran en Anexos: Tarjetas de tarea.

2.7 Conclusiones del capítulo

La definición de XP como metodología ágil para el desarrollo de la aplicación, permitió fijar un tiempo relativamente corto de desarrollo y los artefactos necesarios para el diseño e implementación de la misma. El establecimiento de la arquitectura en capas posibilitó obtener una vista global de la estructura de la aplicación, así como un mayor entendimiento sobre cómo se relacionan cada una de sus partes; lo cual, unido a la identificación de los patrones de diseño, permitió reutilizar código y la implementación de una herramienta más robusta y segura.

Capítulo III: Herramienta de pre – procesamiento y AUCTORITAS

3.1 Introducción

En este capítulo se describen las herramientas implementadas para el análisis y procesamiento de los datos, así como su almacenamiento y consulta. Para una mejor comprensión, el capítulo se ha subdividido en dos subepígrafes: Herramienta de pre – procesamiento y AUCTORITAS; en el primero se detallan las fases correspondientes a la lectura, filtrado y almacenamiento de los datos bibliográficos; y en el segundo se describen los servicios web que brinda la aplicación.

3.2 Herramienta de pre – procesamiento

La herramienta de pre – procesamiento basa su funcionamiento en el análisis de archivos de autoridad. Para el desarrollo de la presente investigación se utilizó el Archivo de Nombres de Autoridad de la LOC (véase 4.3.1.1. Archivo de Nombres de Autoridad de la LOC). Este archivo fue desarrollado con la ontología MADS/RDF, siglas en idioma inglés del término “Esquema de Metadatos para la Descripción de Autoridades” en RDF; un modelo para datos de autoridad y vocabularios utilizados dentro de la comunidad de bibliotecas y ciencias de la información, la cual incluye museos, archivos y otras instituciones culturales (62). Este formato facilitó el desarrollo de las distintas fases del pre – procesamiento de datos, permitiendo establecer una correspondencia entre las etiquetas bajo las cuales se identifican los datos en dicho archivo y la información necesaria para la aplicación AUCTORITAS, de la forma en que se muestra a continuación:

Etiqueta	Información obtenida de la misma
ns0:authoritativeLabel	Autoridad, de la cual se extrae el nombre y los apellidos del autor
rdf:about	URI de la autoridad

3.2.1. Lectura de archivos RDF

El formato RDF es un modelo simple para representar y estructurar datos, que se ha convertido en un método general para el modelado de la información que se implementa en los recursos web (véase Marco de trabajo para la descripción de recursos). Por esta razón,

amplias son las instituciones a nivel mundial que lo han elegido para publicar sus datos de autoridad y consecuentemente, es el formato reconocido por la herramienta propuesta.

Con este fin se implementaron las clases `CAnalizador` e `CItem`, que permiten la lectura de los archivos RDF contenedores de la información bibliográfica. La clase `CItem` constituye la definición formal de una entrada bibliográfica y la clase `CAnalizador` contiene el algoritmo para analizar sintácticamente un archivo RDF y formar las tripletas con la información de un autor determinado, basado en la estructura definida por la clase `CItem`.

3.2.2. Filtrado de las entradas bibliográficas

Las fuentes de datos bibliográficos pueden ser bastante diversas, conteniendo gran cantidad de información personal o institucional en diversos idiomas. La propuesta de solución sólo abarca autores personales, cuyos nombres estén formados por caracteres del alfabeto latino, debido a que las fuentes de datos consumidas no ofrecen patrones para la identificación de otros tipos de autores y no se dominan caracteres pertenecientes a lenguas extranjeras.

Con este propósito los autores de la presente investigación diseñaron, dentro de la clase `CProcesador`, una expresión regular que excluye los datos irrelevantes para AUCTORITAS (véase Anexos: Ilustración #10). Esta clase incluye además las consultas en lenguaje SQL para insertar estos datos en una base de datos PostgreSQL.

3.2.3. Almacenamiento de los nombres de autores personales en una base de datos relacional

El almacenamiento de información en una base de datos ofrece flexibilidad y rapidez al obtener los datos, así como garantiza la seguridad e independencia de los mismos, entre otras ventajas (véase también 1.5.2. Sistema Gestor de Bases de Datos). Por estas y otras razones, se decide almacenar los datos de autores personales en una base de datos relacional utilizando el SGBD PostgreSQL.

La clase `CConeccion` se encarga de la conexión a la base de datos PostgreSQL a través de una piscina de conexiones, el cual mantiene las conexiones en cache posibilitando su reutilización; de esta forma se mejora el rendimiento al ejecutar comandos en una base de datos y se reduce el tiempo que un usuario tiene que esperar para establecer una conexión

a la base de datos. Dicha conexión se realiza mediante programación multihilo, con el objetivo de paralelizar el procesamiento de los registros; esto permite mayor rapidez de procesamiento al ejecutar la aplicación en una computadora con múltiples CPU. Para facilitar su consulta los datos se dispusieron en una única tabla con los campos: nombre, apellidos y autoridad[26].

3.3 AUCTORITAS

AUCTORITAS está desarrollada en lenguaje Java con el IDE NetBeans, cuenta con dos módulos: Servicios y GeneradorXML. El módulo Servicios contiene las clases necesarias para la implementación de los servicios web que brinda la aplicación, los cuales son descritos en el subepígrafe siguiente de este documento; GeneradorXML se encarga de generar la respuesta dada por los servicios web (véase 3.3.2 Formato de respuesta dada por AUCTORITAS).

3.3.1. Servicios web ofrecidos por AUCTORITAS

La utilización de servicios web para el intercambio de datos aporta numerosas ventajas desde el mismo momento en que son creados, pues son aplicaciones de implementación sencilla que no requieren interfaces de usuario complejas, pueden ser desarrollados en una gran variedad de lenguajes para ser implementados sobre muchos tipos de redes de computadoras. Aportan interoperabilidad entre aplicaciones mediante la adopción de estándares y protocolos abiertos, los cuales facilitan acceder a su contenido y entender su funcionamiento. Igualmente, permiten que servicios de empresas u organizaciones distantes geográficamente, puedan interactuar para proveer servicios integrados (63).

Los servicios web provistos por AUCTORITAS están basados en REST. Este estilo posibilita que usando el mismo protocolo, se pueda acceder a todos los recursos, ya sean una cola de mensajes o una base de datos, reemplazar un recurso solo implica cambiar su nombre. El contrato (mensaje) entre el consumidor y el productor tiene que estar estandarizado, pero el estándar no contiene ninguna información sobre el contenido del mensaje; esto aporta sencillez a la ejecución del servicio propiamente dicho, pues es responsabilidad de ambas partes validar y procesar el mensaje (17).

[26] Autoridad se refiere a toda la información del autor, que incluye nombre, apellidos, años de nacimiento y muerte, entre otros datos.

Los servicios web brindados por la aplicación son: Listado de vocabularios controlados, Término autorizado y Datos de autor personal, los cuales son detallados a continuación. En el caso de los dos primeros servicios, estos basan su funcionamiento en la realización de consultas a vocabularios controlados, los cuales se encuentran en formato RDF. Este formato permitió establecer una correspondencia entre las etiquetas bajo las cuales se identifican los datos en dicho archivo y la información a consumir por los servicios web, de la forma en que se muestra a continuación:

Etiqueta	Información obtenida de la misma
rdf:about	URI del término autorizado
skos:prefLabel	Término autorizado
skos:altLabel	Términos alternativos

3.3.1.1. Listado de vocabularios controlados

Este servicio web posibilita la visualización de las URIs correspondientes a cada uno de los vocabularios controlados previamente insertados en el servidor Virtuoso. Su objetivo es proveer un listado que permita al usuario conocer los vocabularios en los cuales puede realizar búsquedas de términos autorizados; constituye la base del servicio "Término autorizado", descrito en el próximo subepígrafe.

Este servicio se define en la clase C`ListaUris`, encargada de recuperar las URIs de los vocabularios controlados disponibles en el servidor Virtuoso; y se apoya en las clases: `CDaoListaUris` y `CEjecutorListaUris`. Un esquema de este servicio se muestra en Anexos: Ilustración #11.

3.3.1.2. Término autorizado

Al registrar una publicación, esta se identifica con palabras clave (o términos) que ofrecen una vía más para recuperar la misma y maximizan su visibilidad en todo tipo de buscadores. Para optimizar este proceso se han creado vocabularios controlados para ramas específicas de las ciencias, que dado un término determinado, permiten recuperar el término autorizado con el cual debe ser registrada la publicación en cuestión.

Haciendo uso de esta lógica, el servicio web Término autorizado recibe un término y el vocabulario controlado en el cual se quiere realizar la búsqueda, y devuelve el término

autorizado correspondiente. El servicio se encuentra implementado en la clase `CTermAut`, la cual hace uso de las clases `CEjecutorTermAut` y `CDaoTermAut`. Un esquema de este servicio se muestra en Anexos: Ilustración #12.

3.3.1.3. Datos de autor personal

El servicio web Datos de autor personal recibe datos de un autor determinado y devuelve la forma normalizada de registrar el mismo. Este servicio se concreta en las clases `CAutorPers` y `CDaoAutorPers`, las cuales en conjunto llevan a cabo una búsqueda en la base de datos que contiene los datos bibliográficos, previamente analizados por las herramientas de pre – procesamiento. Un esquema de este servicio se muestra en Anexos: Ilustración #13.

3.3.2. Formato de respuesta dada por AUCTORITAS

Cualquier usuario puede acceder a servicios web utilizando XML, independientemente de la plataforma, lenguaje o modelo de objetos que utilice. Basado en este principio, XML es el formato de la respuesta dada por los servicios web que propone la aplicación en cuestión. XML se corresponde con las siglas en idioma inglés del término “Lenguaje de Marcado Extensible”. El mismo fue desarrollado por el W3C y es utilizado para almacenar datos en forma legible. Resulta útil para la comunicación o integración de varias aplicaciones entre sí (64), como es el caso de AUCTORITAS y las aplicaciones externas que hagan uso de sus servicios.

3.4 Conclusiones del capítulo

En este capítulo se describió el funcionamiento de la herramienta de pre – procesamiento y de AUCTORITAS, lo que permitió detallar el funcionamiento de las mismas y las fases por las que transitan los datos, desde su filtrado e inserción en una base de datos, hasta su consulta por otras aplicaciones. De igual manera se analizaron tecnologías tales como servicios web basados en REST y XML como formato de respuesta, permitiendo justificar el uso de las mismas para el desarrollo de la aplicación.

Capítulo IV: Análisis de los resultados

4.1 Introducción

El presente capítulo constituye una recopilación de diversas pruebas realizadas a la aplicación, con el objetivo de asegurar el cumplimiento de los requisitos funcionales establecidos para la misma. Para ello, se plasman las pruebas definidas por la metodología seleccionada, haciendo uso de un *framework* diseñado con este fin.

4.2 Pruebas realizadas

Una de las características de la metodología XP son las pruebas unitarias continuas, frecuentemente repetidas y automatizadas. Se aconseja escribir el código de la prueba antes de la codificación. (50) XP pone la comprobación como el fundamento del desarrollo, con cada programador escribiendo pruebas cuando escriben su código de producción. Las pruebas se integran en el proceso de integración continua y construcción, lo que proporciona una plataforma altamente estable para el desarrollo futuro (65).

4.2.1. Pruebas unitarias

XP propone la realización de pruebas unitarias, realizadas a nivel de las clases y los métodos construidos para la aplicación. Estas pruebas son implementadas por los mismos programadores en el lenguaje de programación, utilizando clases de prueba encargadas de verificar el comportamiento correcto de los métodos en una clase (66). Para ello se hizo uso del marco de trabajo JUnit. Las funcionalidades escogidas para ser probadas coinciden con los métodos más complejos de la aplicación, localizados en clases que influyen directamente en la respuesta dada por el sistema.

El empleo de JUnit conlleva a que sean creados paquetes de pruebas que invocan a las clases y métodos a probar, permitiendo comparar sus resultados con los esperados. Los resultados de las pruebas se almacenan en una lista. Luego se comprueba que cada uno de los resultados obtenidos coincide con los resultados esperados y se muestran en una ventana. Cuando estos son satisfactorios para todas las pruebas realizadas, se muestra una línea verde, en caso contrario aparece una línea roja.

La prueba final realizada a la herramienta de pre – procesamiento, arrojó los resultados que se muestran en la ilustración siguiente. Los resultados de las pruebas se muestran en Anexos: Pruebas unitarias.

En la primera iteración de la fase de pruebas, se obtuvieron dos incorrectas de un total de 7 pruebas realizadas, correspondientes al módulo de pre – procesamiento y a los métodos encargados de generar los archivos XML. En una segunda fase, se agregaron al paquete de pruebas inicial los restantes métodos de la aplicación, obteniéndose 3 pruebas insatisfactorias, las cuales fueron corregidas en la última iteración, donde todos los métodos retornaron los resultados esperados.

4.2.2. Pruebas de carga

Las pruebas de carga son aquellas que permiten verificar el comportamiento de la aplicación en condiciones de carga normales y máximas. Se llevan a cabo para comprobar si la aplicación cumple con los objetivos de rendimiento deseados; así como para medir los tiempos de respuesta, tasas de rendimiento y los niveles de recursos de utilización, y para identificar el punto de ruptura de la aplicación, bajo el supuesto de que el mismo se produce por debajo de la condición de carga máxima (67).

Se realizaron pruebas de carga a los tres servicios web ofrecidos por la aplicación, la cual se ejecutó en una computadora con las siguientes prestaciones: procesador Intel Core i3 de segunda generación, 4 gigabytes de memoria RAM y sistema operativo Windows 8.1

El diseño y los resultados de las pruebas se muestran en Anexos: Pruebas de carga.

4.3 Validación de la herramienta

A continuación se enuncian las vías utilizadas para validar la propuesta de solución.

4.3.1. Utilización de datos de prueba

4.3.1.1. Archivo de Nombres de Autoridad de la LOC

Para la validación de la herramienta de pre – procesamiento, cuyos datos de salida tributan a la correcta ejecución del servicio web “Datos de autor personal”, se utilizó el Archivo de Nombres de Autoridad (NAF por sus siglas en inglés) de la LOC. El NAF provee datos de autoridad para nombres de personas, organizaciones, eventos, lugares y títulos. Su propósito es la identificación de estas entidades y, a través del uso de dichos vocabularios

controlados, proveer acceso uniforme a recursos bibliográficos. Actualmente contiene más de ocho millones de descripciones, creadas durante varias décadas de acuerdo a diferentes políticas de catalogación. El NAF constituye un esfuerzo cooperativo en el cual los participantes siguen un conjunto común de estándares y directrices (68).

Se escogió este fichero por las siguientes razones:

- Ser considerado uno de los más completos a nivel internacional, conteniendo un enorme cúmulo de información de datos bibliográficos de autores personales e institucionales de todo el mundo.
- El prestigio alcanzado por la LOC en el campo del control de autoridades, siendo una de las organizaciones punteras a nivel mundial.

Este fichero al momento de su descarga presentaba un peso de 50 gigabytes, el cual no podía ser analizado con el poder de cómputo presente en la UCI. Por esta razón fue subdivido en 195 ficheros, conteniendo cada uno cuatro millones de tripletas RDF aproximadamente, lo cual facilitó su lectura, filtrado y almacenamiento en la BD. La información contenida en dicho archivo, fue utilizada para realizar pruebas a la herramienta de pre – procesamiento y al servicio web “Datos de autor personal”.

4.3.1.2. Vocabulario de la ACM para las Ciencias de la Computación

Los servicios web referentes a los vocabularios controlados, fueron validados mediante un fichero en formato RDF que contiene el vocabulario de la ACM para las Ciencias de la Computación, más conocido por sus siglas CCS (del inglés *Computing Classification System*). Su versión del 2012 (que fue la utilizada para la pruebas), fue desarrollado como una ontología polijerárquica que puede ser utilizada en aplicaciones de la web semántica. Depende de un vocabulario semántico como la única fuente de categorías y conceptos que reflejan el estado del arte de las ciencias de la computación. Este vocabulario se encuentra disponible libremente para propósitos educacionales y de investigación en los formatos: SKOS[27], Word y HTML (69). Este archivo fue almacenado en un servidor Virtuoso, para la realización de consultas al mismo en lenguaje SPARQL desde la aplicación AUCTORITAS,

[27] SKOS: siglas de *Simple Knowledge Organization System,* es una iniciativa del W3C en forma de aplicación de RDF que proporciona un modelo para representar la estructura básica y el contenido de esquemas conceptuales como listas encabezamientos de materia, taxonomías, esquemas de clasificación, tesauros y cualquier tipo de vocabulario controlado.

lo cual permitió probar los servicios web “Listado de vocabularios controlados” y “Término autorizado”.

4.3.2. Entrevistas realizadas

Una vez concluida la frase de pruebas de la herramienta propuesta, la misma fue mostrada a especialistas del CIGED, previo a la realización de entrevistas a los mismos. Dicho encuentro se realizó con el objetivo de obtener retroalimentación sobre la propuesta de solución y constatar si la misma, podría representar una propuesta viable como mecanismo de control de autoridades en sus respectivos proyectos. De los tres especialistas entrevistados, la totalidad de los mismos afirmó que la integración de AUCTORITAS a sus proyectos representa una necesidad, que ayudará a realizar el registro y recuperación de publicaciones por los campos más utilizados para esto: autor y materia.

Las entrevistas realizadas se pueden observar en Anexos: Entrevistas realizadas, identificadas con el #2.

4.4 Conclusiones del capítulo

En el capítulo se plasmaron los aspectos principales de la fase de pruebas y validación de ambas herramientas. La realización de las mismas posibilitó comprobar que el funcionamiento de cada uno de los métodos responde con el comportamiento esperado. La utilización del *framework* de pruebas Junit, el cual se encuentra incorporado al IDE con el cual se desarrolló la aplicación, permitió la creación y ejecución de pruebas automatizadas. Los conjuntos de datos provenientes de la LOC y los de vocabularios controlados, permitieron validar la solución y garantizar que la herramienta provee la respuesta esperada para cada servicio.

Conclusiones generales

Al término de la presente investigación, se arribó a las siguientes conclusiones:

- El establecimiento de los fundamentos teóricos y metodológicos relevantes del control de autoridades y tecnologías asociadas, permitieron sentar las bases del proceso investigativo y conocer la lógica del negocio, tomando como punto de partida, soluciones informáticas existentes diseñadas con el mismo fin.
- El análisis y diseño de las herramientas de pre – procesamiento de datos y AUCTORITAS, posibilitaron formalizar la estructura ingenieril de la propuesta de solución y definir un plan de desarrollo acorde a la misma.
- La implementación del software propuesto y la herramienta auxiliar, dio lugar a una aplicación funcional que permite consultar datos bibliográficos, mediante el consumo de servicios web.
- Las pruebas y validaciones realizadas a la solución obtenida, posibilitaron comprobar la calidad de la misma, garantizando la entrega de un producto de software libre de errores y listo para su uso.

Recomendaciones

Se recomienda para una futura versión de la aplicación integrar AUCTORITAS con VIVO, una aplicación desarrollada en la Universidad de Cornell (Estados Unidos), que permite el manejo de datos bibliográficos de forma automatizada. La misma se describe como una herramienta basada en la web semántica, para el descubrimiento de investigadores y publicaciones científicas. Provee información multidisciplinaria, visible públicamente para conectar especialistas, comunidades de investigadores, campus universitarios y el mundo usando los LOD.

Durante el desarrollo de la presente investigación no se utilizó dicha aplicación, pues la misma se encuentra en fase de pruebas.

Referencias bibliográficas

1. W3C. Guía Breve de Web Semántica. *World Wide Web Consortium (W3C) - España.* [En línea] 2015. [Citado el: 12 de marzo de 2015.] http://www.w3c.es/Divulgacion/GuiasBreves/WebSemantica.

2. Leiva-Mederos, Amed, et al. *AUTHORIS: a tool for authority control in the semantic web.* Departamento de Información y Comunicación, Universidad de Granada. Granada, España : s.n., 2013. p. 2.

3. Fumagalli, Giuseppe. *Cataloghi di biblioteche e indici bibliografici.* Sansoni, Florence : s.n., 1887.

4. *Authority Control: State of the Art and New Theoretical Perspectives.* B. Tillet, PhD. Barbara. Londres : Routledge, 2009, Cataloging & Classification Quarterly.

5. Tabares Martín, Leandro, Leiva Mederos, Amed Abel y Fernández Peña, Félix Oscar. *Diseño de un sistema para la generación de entradas de autoridad desde la perspectiva de los datos enlazados.* La Habana : s.n., 2014. Publicado en el XIII Congreso Internacional de Información.

6. Campos Herrera, Lic. Airelys. Las reglas angloamericanas de catalogación y la norma ISO 690 - 1 y 2. *SciELO.* [En línea] 2007. http://scielo.sld.cu/scielo.php?pid=S1024-94352007000800010&script=sci_arttext.

7. *Control de autoridades de nombres personales de autores cubanos en ciencias de la salud.* Díaz Rodríguez, Lic. Yoselyn. 1, La Habana : s.n., 23 de enero de 2012, Revista Cubana de Información en Ciencias de la Salud [versión electrónica], Vol. 23. ISSN: 2307-2113.

8. Subdirección General de Información y Publicaciones, Ministerio de Cultura, España. *Directrices para Registros de Autoridad y Referencias.* s.l. : Secretaría General Técnica, 2004. 84-369-3838-0.

9. Biblioteca Nacional de España. *Manual de Autoridades.* Departamento de Proceso Técnico, Biblioteca Nacional de España. 2010. pág. 235.

10. *The semantic web.* Berners - Lee, Tim, Handler, J. and Lassila, Ora. 3, 2006, Database and Network journal, Vol. 36.

11. Castells, Pablo. *Aplicación de técnicas de la web semántica.* Escuela Politécnica Superior, Universidad Autónoma de Madrid. Madrid : s.n., 2002.

12. Yu, Liyang. *A Developer's Guide to the Semantic Web.* Atlanta : Springer- Verlag Berlin Heidelberg, 2011. pp. 409-410. 978-3-642-15969-5.

13. *OPEN DATA Y LINKED OPEN DATA: SU IMPACTO EN EL ÁREA DE BIBLIOTECAS Y DOCUMENTACIÓN.* Peset, Fernanda, Ferrer - Sapena, Antonia y Subirats - Coll, Imma. 2, marzo - abril de 2011, El profesional de la información, Vol. 20.

14. Semantic Web Standards. Resource Description Framework (RDF). [Online] febrero 25, 2014. [Cited: marzo 18, 2015.] http://www.w3.org/2001/sw/wiki/RDF.

15. Navarro Marset, Rafael. *REST vs. Web Services.* 2007. págs. 2-19.

16. World Wide Web Consortium. Simple Object Access Protocol (SOAP) 1.1 . *W3C Note.* [Online] World Wide Web Consortium, mayo 8, 2000. [Cited: mayo 15, 2015.] http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

17. Wesenberg, Harald and Landre, Einar. *REST versus SOAP as Architectural Style for Web Services.* StatoilHydro ASA.

18. Library of Congress. NACO - Name Authority Cooperative Program. *Library of Congress.* [En línea] [Citado el: 2 de marzo de 2015.] http://www.loc.gov/aba/pcc/naco/index.html.

19. Joint University Librarians Advisory Committee. Hong Kong Chinese Authority Name Project (HKCAN). *JULAC Joint University Librarians Advisory Committee.* [Online] 2015. [Cited: marzo 2, 2015.] http://www.julac.org/?page_id=2137.

20. Núñez, Sonia. El control de autoridades y el formato UNIMARC, evolución y desarrollo: Biblioteca Nacional "José Martí". [En línea] 2007. http://www.bpvillena.ohc.cu/index.php?option=com_content&view=article&id=272:ponencia-el-control-de-autoridades-y%20-el-formato-unimarc-evoluci-desarrollo&catid=49:evento&Itemid=47.

21. Masinter, Larry, Berners - Lee, Tim and Fielding, Roy T. Uniform resource identifier (URI): Generic syntax. [Online] 2005. [Cited: mayo 7, 2015.] https://tools.ietf.org/html/rfc3986#page-4.

22. Alegsa, Leandro. Definición de Lenguaje de programación. *DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA .* [En línea] [Citado el: 4 de marzo de 2015.] http://www.alegsa.com.ar/Dic/lenguaje%20de%20programacion.php.

23. García de Jalón, Javier, y otros. *Aprenda Java como si estuviera en primero.* San Sebastián : s.n., 2000. págs. 1-2.

24. World Wide Web Consortium. SPARQL Lenguaje de consulta para RDF. *W3C.* [En línea] 15 de enero de 2008. [Citado el: 11 de mayo de 2015.] http://travesia.mcu.es/portalnb/jspui/bitstream/10421/6401/1/sparql_es.pdf.

25. Martín Escofet, Carme. *El lenguaje SQL.*

26. El lenguaje SQL. *Tutorial de PostgreSQL - SQL.* [En línea] [Citado el: 27 de mayo de 2015.] (http://es.tldp.org/Postgresql-es/web/navegable/tutorial/sql-language.html.

27. *Eclipse becomes the dominant Java IDE.* Geer, David. 7, julio 11, 2005, Computer, Vol. 38.

28. Domínguez Jiménes, J. J., y otros. *El reto de los servicios Web para el software libre.* Departamento de Lenguajes y Sistemas Informáticos, Universidad de Cádiz. 2007. pág. 117. Este libro se distribuye bajo licencia Creative Commons Reconocimiento CompartirIgual 2.5 Espa.

29. Oracle. NetBeans IDE - The Smarter and Faster Way to Code. *NetBeans.* [Online] Oracle Corporation, 2015. [Cited: marzo 23, 2015.] https://netbeans.org/features/index.html.

30. *Sistemas gestores de bases de datos.* Garzón Pérez, María Teresa. 30, Granada : s.n., mayo de 2010, Revista digital "Innovación y experiencias educativas", pág. 5. ISSN 1988-6047.

31. Lozano, Carlos. SGBD CARACTERISTICAS VENTAJAS DESVENTAJAS REQUERIMIENTOS . *Academia.edu.* [En línea] 2015. [Citado el: 9 de marzo de 2015.] http://www.academia.edu/8199329/SGBD_CARACTERISTICAS_VENTAJAS_DESVENTAJAS_REQUERIMIENTOS.

32. PostgreSQL Global Development Team. Sobre PostgreSQL. *PostgreSQL [sitio oficial].* [En línea] [Citado el: 4 de marzo de 2015.] http://www.postgresql.org.es/sobre_postgresql.

33. The Apache Software Foundation. Apache Jena Fuseki. *Apaceh Jena.* [Online] 2015. [Cited: mayo 12, 2015.] http://jena.apache.org/documentation/fuseki2/.

34. System Properties Comparison Jena vs. Sesame vs. Virtuoso. *DB-Engines Knowledge Base of Relational and NoSQL Database Management Systems.* [Online] 2015. http://db-engines.com/en/systems.

35. OpenLink Software. Virtuoso Universal Server Home Page. *OpenLink Software.* [Online] OpenLink Software, 2015. [Cited: marzo 10, 2015.] http://virtuoso.openlinksw.com/.

36. Fernández Peña, Juan Manuel. *Pruebas de unidad utilizando JUnit.* 2005. págs. 1-10.

37. *Pruebas Unitarias en Java JUnit y TestNG.* Ávila, Adriana, y otros. Milveinticuatro, págs. 47-48.

38. IBM Corporation. IBM - Rational Performance Tester. *IBM.* [En línea] IBM Corporation. [Citado el: 28 de mayo de 2015.] http://www-03.ibm.com/software/products/es/performance.

39. Rapid API Load Testing with LoadUI NG Pro | SmartBear Software. *SmartBear.* [Online] SmartBear Software. [Cited: mayo 27, 2015.] http://smartbear.com/product/ready-api/loadui/overview/.

40. SmartBear Software. What is soapUI? | About SoapUI. *SmartBear Software.* [Online] SmartBear Software, 2015. [Cited: mayo 27, 2015.] http://www.soapui.org/about-soapui/what-is-soapui-.html.

41. J. Gutiérrez, Javier. *¿Qué es un framework web?* Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla. Sevilla : s.n.

42. Sesame. Sesame. [Online] octubre 15, 2014. [Cited: mayo 14, 2015.] http://rdf4j.org/about.docbook?view.

43. The Apache Software Foundation. Getting started with Apache Jena. *Apache Jena.* [Online] The Apache Software Foundation, 2015. [Cited: mayo 15, 2015.] The Apache Software Foundation.

44. World Wide Web Consortium. Jena, a Java RDF API and toolkit. *Semantic Web Standards.* [En línea] World Wide Web Consortium, 15 de noviembre de 2014. [Citado el: 10 de marzo de 2015.] http://www.w3.org/2001/sw/wiki/Apache_Jena.

45. Zacchiroli, Stefano. *Génie Logiciel Avancé Cours 6 — Extreme Programming.* Laboratoire PPS, Université Paris Diderot - Paris 7. Paris : s.n., 2012.

46. Kasiak, Tamara y Godoy, Diego Alberto . *Simulación de Proyectos de Software desarrollados con XP: Subsistema de Desarrollo de Tareas.* Centro de Investigación en Tecnologías de la Información y Comunicaciones (C.I.T.I.C.), Universidad Gastón Dachary. 2012. pág. 5.

47. H. Canós, José, Letelier, Patricio y Penadés, Mª Carmen. *Métodologías Ágiles en el Desarrollo de Software.* Universidad Politécnica de Valencia. Valencia : s.n.

48. Griffiths, Mike. Agile Suitability Filters. *Leading Answers.* [Online] 2007. [Cited: mayo 27, 2015.] www.leadinganswers.com.

49. Godoy, Diego Alberto, y otros. *Simulando Proyectos de Desarrollo de Software Administrados con Scrum.* Departamento de Ingeniería y Ciencias de la Producción , Centro de Investigación en Tecnologías de la Información y Comunicaciones (C.I.T.I.C.), Universidad Gastón Dachary. 2014. pág. 5.

50. G. Figueroa, Roberth, J. Solís, Camilo y A. Cabrera, Armando. *METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES.* Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación. 2011. pág. 9.

51. de la Torre Llorente, César, y otros. *Guía de Arquitectura N-Capas orientada al Dominio con .NET 4.0.* Microsoft Ibérica S.R.L. s.l. : Krasis Consulting, S. L., 2010. ISBN: 978-84-936696-3-8.

52. *Patrones de diseño: ejemplo de aplicación en los Generative Learning Object.* Cáceres Tello, Jesús. Murcia : Sistema de Información Científica, Red de Revistas Científicas de América Latina, el Caribe, España y Portugal, noviembre de 2009, RED. Revista de Educación a Distancia, Vol. X, pág. 14.

53. Damián Campo, Gustavo. *Patrones de Diseño, Refactorización y Antipatrones. Ventajas y Desventajas de su Utilización en el Software Orientado a Objetos.* Universidad Católica de Salta. Buenos Aires : s.n., 2009.

54. Larman, Craig. *UML y Patrones Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* 2ª. s.l. : Prentice Hall.

55. Canales Mora, Roberto. Patrones de GRASP. *Adictos al trabajo.* [En línea] 22 de diciembre de 2003. [Citado el: 14 de mayo de 2015.] http://www.adictosaltrabajo.com/tutoriales/grasp/.

56. *Curso de Patrones y Anti patrones de Diseño.* Universidad de las Ciencias Informáticas. La Habana : s.n.

57. Microsoft Corporation. El Patrón Singleton . *Microsoft Development Network.* [En línea] 2015. [Citado el: 14 de mayo de 2015.] https://msdn.microsoft.com/es-es/library/bb972272.aspx.

58. Addison Wesley. *Design Patterns CD Elements of Reusable Object - Oriented Software.* [Multimedia] Professional Computing Series.

59. Joskowicz, Ing. José. *Reglas y Prácticas en eXtreme Programming.* Universidad de Vigo. 2008. pág. 22.

60. Vallespir, Ing. Diego. *CRC y un Taller.* Facultad de Ingeniería - Universidad de la República. Montevideo, Uruguay : s.n., 2002. pág. 10.

61. Cuccaro Goggi, Lucila Ana. *Adecuación de la metodología de desarrollo Extreme Programming a proyectos llevados a cabo en la materia Laboratorio III de la Facultad de Ingeniería de la Universidad Austral.* Facultad de Ingeniería, Universidad Austral. págs. 23-24.

62. Library of Congress. MADS/RDF Primer. *Library of Congress.* [Online] Library of Congress, 2015. [Cited: mayo 27, 2015.] http://www.loc.gov/standards/mads/rdf/.

63. Morales Machuca, Carlos Andrés. *Estado del Arte: Servicios Web.* Universidad Nacional de Colombia. Bogotá : s.n. págs. 1-2.

64. Bray, Tim, et al. *Extensible Markup Language (XML).* World Wide Web Consortium. 2006.

65. Fowler, Martin. *La Nueva Metodología.* ThoughtWorks. 2003. Texto original: The New Methodology.

66. Marcelo Hernán, Schenone. *Diseño de una Metodología Ágil de Desarrollo de Software.* Universidad de Buenos Aires, Facultad de Ingeniería. Buenos Aires : s.n., 2004. pág. 2004, Tesis de Grado en Ingeniería en Informática.

67. D. Meier, J., et al. Chapter 2 – Types of Performance Testing. *Microsoft Developer Network.* [Online] Microsoft Corporation, septiembre 2007. [Cited: mayo 28, 2015.] https://msdn.microsoft.com/en-us/library/bb924357.aspx.

68. Library of Congress. Library of Congress Names. *Library of Congress.* [Online] Library of Congress, 2015. [Cited: mayo 27, 2015.] http://id.loc.gov/authorities/names.html.

69. Association for Computing Machinery. The 2012 ACM Computing Classification System. *Association for Computing Machinery.* [Online] ACM, 2015. [Cited: mayo 25, 2015.] https://www.acm.org/about/class/2012.

Bibliografía

¿Está muriendo la biblioteca? Hacia la e-evolución. Vicente-De-Billion, Chloé y Oyarce-Gatica, Alejandro. 2010, El profesional de la información, págs. 70-76.

Addison Wesley. *Design Patterns CD Elements of Reusable Object - Oriented Software.* [Multimedia] Professional Computing Series.

Alegsa, Leandro. Definición de Lenguaje de programación. *DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA .* [En línea] [Citado el: 4 de marzo de 2015.] http://www.alegsa.com.ar/Dic/lenguaje%20de%20programacion.php.

Alexander Zuluaga, Carlos. *Enterprise Architect y UML Basic.* 2008. pág. 21.

Apache Software Foundation. An Introduction to RDF and the Jena RDF API. *Apache Jena.* [Online] 2011-2013. [Cited: marzo 11, 2015.] http://jena.apache.org/tutorials/rdf_api.html.

Association for Computing Machinery. The 2012 ACM Computing Classification System. *Association for Computing Machinery.* [Online] ACM, 2015. [Cited: mayo 25, 2015.] https://www.acm.org/about/class/2012.

Authority Control: State of the Art and New Theoretical Perspectives. B. Tillet, PhD. Barbara. Londres : Routledge, 2009, Cataloging & Classification Quarterly.

Biblioteca Nacional de España. *Manual de Autoridades.* Departamento de Proceso Técnico, Biblioteca Nacional de España. 2010. pág. 235.

Bravo Santos, Crescencio y Redondo Duque, Miguel Ángel. *Sistemas interactivos y colaborativos en la web.* 2004. pág. 279.

Bray, Tim, et al. *Extensible Markup Language (XML).* World Wide Web Consortium. 2006.

Campos Herrera, Lic. Airelys. Las reglas angloamericanas de catalogación y la norma ISO 690 - 1 y 2. *SciELO.* [En línea] 2007. http://scielo.sld.cu/scielo.php?pid=S1024-94352007000800010&script=sci_arttext.

Canales Mora, Roberto. Patrones de GRASP. *Adictos al trabajo.* [En línea] 22 de diciembre de 2003. [Citado el: 14 de mayo de 2015.] http://www.adictosaltrabajo.com/tutoriales/grasp/.

Castells, Pablo y Saíz, Francisco. *Ontologías para la Web semántica. La Web semántica: tecnologías y aplicaciones.* Madrid : Escuela Politécnica Superior, Universidad Autónoma de Madrid, 2003. pág. 16.

Castells, Pablo. *Aplicación de técnicas de la web semántica.* Escuela Politécnica Superior, Universidad Autónoma de Madrid. Madrid : s.n., 2002.

Control de autoridades de nombres personales de autores cubanos en ciencias de la salud. Díaz Rodríguez, Lic. Yoselyn. 1, La Habana : s.n., 23 de enero de 2012, Revista Cubana de Información en Ciencias de la Salud [versión electrónica], Vol. 23. ISSN: 2307-2113.

Cuccaro Goggi, Lucila Ana. *Adecuación de la metodología de desarrollo Extreme Programming a proyectos llevados a cabo en la materia Laboratorio III de la Facultad de Ingeniería de la Universidad Austral.* Facultad de Ingeniería, Universidad Austral. págs. 23-24.

Curso de Patrones y Anti patrones de Diseño. Universidad de las Ciencias Informáticas. La Habana : s.n.

D. Meier, J., et al. Chapter 2 – Types of Performance Testing. *Microsoft Developer Network.* [Online] Microsoft Corporation, septiembre 2007. [Cited: mayo 28, 2015.] https://msdn.microsoft.com/en-us/library/bb924357.aspx.

Damián Campo, Gustavo. *Patrones de Diseño, Refactorización y Antipatrones. Ventajas y Desventajas de su Utilización en el Software Orientado a Objetos.* Universidad Católica de Salta. Buenos Aires : s.n., 2009.

de la Torre Llorente, César, y otros. *Guía de Arquitectura N-Capas orientada al Dominio con .NET 4.0.* Microsoft Ibérica S.R.L. s.l. : Krasis Consulting, S. L., 2010. ISBN: 978-84-936696-3-8.

Domínguez Jiménes, J. J., y otros. *El reto de los servicios Web para el software libre.* Departamento de Lenguajes y Sistemas Informáticos, Universidad de Cádiz. 2007. pág. 117. Este libro se distribuye bajo licencia Creative Commons Reconocimiento CompartirIgual 2.5 Espa.

Eclipse becomes the dominant Java IDE. Geer, David. 7, julio 11, 2005, Computer, Vol. 38.

El lenguaje SQL. *Tutorial de PostgreSQL - SQL.* [En línea] [Citado el: 27 de mayo de 2015.] (http://es.tldp.org/Postgresql-es/web/navegable/tutorial/sql-language.html.

Escalona Cuaresma, Dra. María José y Gutiérrez Rodríguez, D. Javier Jesús. *Introducción a Enterprise Architecture.* Universidad de Sevilla. Sevilla : s.n., 2007. pág. 15.

Evolución histórica de la función social de las bibliotecas públicas. Fernández Abad, Francisco Javier. 2, 2006, Revista General de Información y Documentación.

Fernández Peña, Juan Manuel. *Pruebas de unidad utilizando JUnit.* 2005. págs. 1-10.

Fowler, Martin. *La Nueva Metodología.* ThoughtWorks. 2003. Texto original: The New Methodology.

Fuentes, Lidia y Vallecillo, Antonio. *Una introducción a los perfiles UML.* Dpto. de Lenguajes y Ciencias de la Computación, Universidad de Málaga, Campus de Teatino. Málaga : s.n. pág. 13. E29071.

Fumagalli, Giuseppe. *Cataloghi di biblioteche e indici bibliografici.* Sansoni, Florence : s.n., 1887.

G. Figueroa, Roberth, J. Solís, Camilo y A. Cabrera, Armando. *METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES.* Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación. 2011. pág. 9.

García de Jalón, Javier, y otros. *Aprenda Java como si estuviera en primero.* San Sebastián : s.n., 2000. págs. 1-2.

Godoy, Diego Alberto, y otros. *Simulando Proyectos de Desarrollo de Software Administrados con Scrum.* Departamento de Ingeniería y Ciencias de la Producción , Centro de Investigación en Tecnologías de la Información y Comunicaciones (C.I.T.I.C.), Universidad Gastón Dachary. 2014. pág. 5.

Griffiths, Mike. Agile Suitability Filters. *Leading Answers.* [Online] 2007. [Cited: mayo 27, 2015.] www.leadinganswers.com.

Grupo Knowledge Reuse Group. *Knowledge Reuse Group.* [En línea] 2013-2014. [Citado el: 9 de marzo de 2015.] http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf.

H. Canós, José, Letelier, Patricio y Penadés, Mª Carmen. *Métodologías Ágiles en el Desarrollo de Software.* Universidad Politécnica de Valencia. Valencia : s.n.

IBM Corporation. IBM - Rational Performance Tester. *IBM.* [En línea] IBM Corporation. [Citado el: 28 de mayo de 2015.] http://www-03.ibm.com/software/products/es/performance.

J. Gutiérrez, J., y otros. *PRUEBAS DEL SISTEMA EN PROGRAMACIÓN EXTREMA.* Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla.

J. Gutiérrez, Javier. *¿Qué es un framework web?* Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla. Sevilla : s.n.

Javier Lasso de la Vega y los principios de la catalogación bibliográfica. López Guillamón, Ignacio. Madrid : Universidad Complutense de Madrid, 22 de julio de 2012, Revista General de Información y Documentación [versión electrónica], Vol. 22. ISSN 1132-1873.

Joint University Librarians Advisory Committee. Hong Kong Chinese Authority Name Project (HKCAN). *JULAC Joint University Librarians Advisory Committee.* [Online] 2015. [Cited: marzo 2, 2015.] http://www.julac.org/?page_id=2137.

Joskowicz, Ing. José. *Reglas y Prácticas en eXtreme Programming.* Universidad de Vigo. 2008. pág. 22.

Junit / Frequently Asked Questions. *Junit .* [Online] 12 04, 2014. [Cited: mayo 21, 2015.] http://junit.org/faq.html#overview_1.

Kasiak, Tamara y Godoy, Diego Alberto . *Simulación de Proyectos de Software desarrollados con XP: Subsistema de Desarrollo de Tareas.* Centro de Investigación en Tecnologías de la Información y Comunicaciones (C.I.T.I.C.), Universidad Gastón Dachary. 2012. pág. 5.

Larman, Craig. *UML y Patrones Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* 2ª. s.l. : Prentice Hall.

Leiva-Mederos, Amed, et al. *AUTHORIS: a tool for authority control in the semantic web.* Departamento de Información y Comunicación, Universidad de Granada. Granada, España : s.n., 2013. p. 2.

Library of Congress. Library of Congress Names. *Library of Congress.* [Online] Library of Congress, 2015. [Cited: mayo 27, 2015.] http://id.loc.gov/authorities/names.html.

Library of Congress. MADS/RDF Primer. *Library of Congress.* [Online] Library of Congress, 2015. [Cited: mayo 27, 2015.] http://www.loc.gov/standards/mads/rdf/.

Library of Congress. NACO - Name Authority Cooperative Program. *Library of Congress.* [En línea] [Citado el: 2 de marzo de 2015.] http://www.loc.gov/aba/pcc/naco/index.html.

Lozano, Carlos. SGBD CARACTERISTICAS VENTAJAS DESVENTAJAS REQUERIMIENTOS . *Academia.edu.* [En línea] 2015. [Citado el: 9 de marzo de 2015.] http://www.academia.edu/8199329/SGBD_CARACTERISTICAS_VENTAJAS_DESVENTAJAS_REQUERIMIENTOS.

Marcelo Hernán, Schenone. *Diseño de una Metodología Ágil de Desarrollo de Software.* Universidad de Buenos Aires, Facultad de Ingeniería. Buenos Aires : s.n., 2004. pág. 2004, Tesis de Grado en Ingeniería en Informática.

Martín Escofet, Carme. *El lenguaje SQL.*

Masinter, Larry, Berners - Lee, Tim and Fielding, Roy T. Uniform resource identifier (URI): Generic syntax. [Online] 2005. [Cited: mayo 7, 2015.] https://tools.ietf.org/html/rfc3986#page-4.

Microsoft Corporation. El Patrón Singleton . *Microsoft Development Network.* [En línea] 2015. [Citado el: 14 de mayo de 2015.] https://msdn.microsoft.com/es-es/library/bb972272.aspx.

Morales Machuca, Carlos Andrés. *Estado del Arte: Servicios Web.* Universidad Nacional de Colombia. Bogotá : s.n. págs. 1-2.

Navarro Marset, Rafael. *REST vs. Web Services.* 2007. págs. 2-19.

Núñez, Sonia. El control de autoridades y el formato UNIMARC, evolución y desarrollo: Biblioteca Nacional "José Martí". [En línea] 2007. http://www.bpvillena.ohc.cu/index.php?option=com_content&view=article&id=272:ponencia-el-control-de-autoridades-y%20-el-formato-unimarc-evoluci-desarrollo&catid=49:evento&Itemid=47.

OPEN DATA Y LINKED OPEN DATA: SU IMPACTO EN EL ÁREA DE BIBLIOTECAS Y DOCUMENTACIÓN. Peset, Fernanda, Ferrer - Sapena, Antonia y Subirats - Coll, Imma. 2, marzo - abril de 2011, El profesional de la información, Vol. 20.

OpenLink Software. Virtuoso Universal Server Home Page. *OpenLink Software.* [Online] OpenLink Software, 2015. [Cited: marzo 10, 2015.] http://virtuoso.openlinksw.com/.

Oracle. NetBeans IDE - The Smarter and Faster Way to Code. *NetBeans.* [Online] Oracle Corporation, 2015. [Cited: marzo 23, 2015.] https://netbeans.org/features/index.html.

Pastor Sánchez, Juan Antonio y Díaz Ortuño, Pedro Manuel. W3C. *SPARQL Lenguaje de consulta para RDF.* [En línea] W3C, 15 de enero de 2008. [Citado el: 30 de abril de 2015.] http://skos.um.es/TR/rdf-sparql-query/#GroupPatterns.

Patrones de diseño: ejemplo de aplicación en los Generative Learning Object. Cáceres Tello, Jesús. Murcia : Sistema de Información Científica, Red de Revistas Científicas de América Latina, el Caribe, España y Portugal, noviembre de 2009, RED. Revista de Educación a Distancia, Vol. X, pág. 14.

PostgreSQL Global Development Team. Sobre PostgreSQL. *PostgreSQL [sitio oficial].* [En línea] [Citado el: 4 de marzo de 2015.] http://www.postgresql.org.es/sobre_postgresql.

Pruebas Unitarias en Java JUnit y TestNG. Ávila, Adriana, y otros. Milveinticuatro, págs. 47-48.

Rapid API Load Testing with LoadUI NG Pro | SmartBear Software. *SmartBear.* [Online] SmartBear Software. [Cited: mayo 27, 2015.] http://smartbear.com/product/ready-api/loadui/overview/.

Semantic Web Standards. Resource Description Framework (RDF). [Online] febrero 25, 2014. [Cited: marzo 18, 2015.] http://www.w3.org/2001/sw/wiki/RDF.

Sesame. About. *Sesame.* [Online] Callimachus, marzo 4, 2015. [Cited: marzo 11, 2015.] http://rdf4j.org/about.docbook?view.

Sesame. Sesame. [Online] octubre 15, 2014. [Cited: mayo 14, 2015.] http://rdf4j.org/about.docbook?view.

Setién Quesada, Emilio y Pérez Matos, Nuria Esther. *Bibliotecología y Ciencia de la Información: enfoque interdisciplinario.*

Sistemas gestores de bases de datos. Garzón Pérez, María Teresa. 30, Granada : s.n., mayo de 2010, Revista digital "Innovación y experiencias educativas", pág. 5. ISSN 1988-6047.

SmartBear Software. What is soapUI? | About SoapUI. *SmartBear Software.* [Online] SmartBear Software, 2015. [Cited: mayo 27, 2015.] http://www.soapui.org/about-soapui/what-is-soapui-.html.

Sub - Jefatura de Informática, Dirección Técnica de Desarrollo Informático, Instituto Nacional de Estadística e Informática. *Herramientas Case.* 1999. 875-99-OI-OTDETI-INEI.

Subdirección General de Información y Publicaciones, Ministerio de Cultura, España. *Directrices para Registros de Autoridad y Referencias.* s.l. : Secretaría General Técnica, 2004. 84-369-3838-0.

System Properties Comparison Jena vs. Sesame vs. Virtuoso. *DB-Engines Knowledge Base of Relational and NoSQL Database Management Systems.* [Online] 2015. http://db-engines.com/en/systems.

Tabares Martín, Leandro, Leiva Mederos, Amed Abel y Fernández Peña, Félix Oscar. *Diseño de un sistema para la generación de entradas de autoridad desde la perspectiva de los datos enlazados.* La Habana : s.n., 2014. Publicado en el XIII Congreso Internacional de Información.

The Apache Software Foundation. Apache Jena Fuseki. *Apaceh Jena.* [Online] 2015. [Cited: mayo 12, 2015.] http://jena.apache.org/documentation/fuseki2/.

The Apache Software Foundation. Getting started with Apache Jena. *Apache Jena.* [Online] The Apache Software Foundation, 2015. [Cited: mayo 15, 2015.] The Apache Software Foundation.

The semantic web. Berners - Lee, Tim, Handler, J. and Lassila, Ora. 3, 2006, Database and Network journal, Vol. 36.

Vallespir, Ing. Diego. *CRC y un Taller.* Facultad de Ingeniería - Universidad de la República. Montevideo, Uruguay : s.n., 2002. pág. 10.

W3C. Guía Breve de Web Semántica. *World Wide Web Consortium (W3C) - España.* [En línea] 2015. [Citado el: 12 de marzo de 2015.] http://www.w3c.es/Divulgacion/GuiasBreves/WebSemantica.

Wesenberg, Harald and Landre, Einar. *REST versus SOAP as Architectural Style for Web Services.* StatoilHydro ASA.

World Wide Web Consortium. Jena, a Java RDF API and toolkit. *Semantic Web Standards.* [En línea] World Wide Web Consortium, 15 de noviembre de 2014. [Citado el: 10 de marzo de 2015.] http://www.w3.org/2001/sw/wiki/Apache_Jena.

World Wide Web Consortium. Simple Object Access Protocol (SOAP) 1.1 . *W3C Note.* [Online] World Wide Web Consortium, mayo 8, 2000. [Cited: mayo 15, 2015.] http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

World Wide Web Consortium. SPARQL Lenguaje de consulta para RDF. *W3C.* [En línea] 15 de enero de 2008. [Citado el: 11 de mayo de 2015.] http://travesia.mcu.es/portalnb/jspui/bitstream/10421/6401/1/sparql_es.pdf.

Yu, Liyang. *A Developer's Guide to the Semantic Web.* Atlanta : Springer- Verlag Berlin Heidelberg, 2011. pp. 409-410. 978-3-642-15969-5.

Zacchiroli, Stefano. *Génie Logiciel Avancé Cours 6 — Extreme Programming.* Laboratoire PPS, Université Paris Diderot - Paris 7. Paris : s.n., 2012.