Universidad de las Ciencias Informáticas

Facultad 6

Título: AUCTORITAS 2.0: Sistema de apoyo para el Control de Autoridades.

Centro: CIGED

Autores: Flavia González Barroso.
Darayne Pérez González.

Tutores: Ing. Leandro Tabares Martín
Ing. Dailién Calzadilla Reyes.

La Habana, julio de 2016

Nunca dejes para mañana

lo que puedes hacer hoy.

Benjamín Franklin

Declaración de autoría

Se declara que Darayne Pérez González y Flavia González Barroso son los únicos autores del trabajo de diploma AUCTORITAS 2.0: Sistema de apoyo al control de autoridades, y se le reconoce a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste se firma la presente a los ____ días del mes de _____ del año_____.

_____	_____
Firma del autor	Firma del autor
Darayne Pérez González	Flavia González Barroso
_____	_____
Firma del tutor	Firma del tutor
Ing. Leandro Tabares Martín	Ing. Dailién Calzadilla Reyes

Datos de contacto

Datos del Autor:
Nombre: Darayne Pérez González
Correo electrónico: dpgonzalez@estudiantes.uci.cu

Datos del Autor:
Nombre: Flavia González Barroso
Correo electrónico: fbarroso@estudiantes.uci.cu

Datos de Tutor:
Nombre: Ing. Leandro Tabares Martín
Ingeniero en Ciencias Informáticas.
Correo electrónico: ltmartin@uci.cu

Datos de Tutor:
Nombre: Ing. Dailién Calzadilla Reyes
Ingeniero en Ciencias Informáticas.
Correo electrónico: dcreyes@uci.cu

Agradecimientos de Darayne

Realmente ha sido duro llegar hasta este lugar, le doy gracias a Dios por ello porque sin él no habría sido posible pasar por tanto. Le doy infinitas gracias a mi hermana, gracias a mi mamá y a mi papá que me lo han dado todo para que yo pudiera estar aquí hoy, le doy gracias porque me han apoyado en cada una de las decisiones que he tomado en mi vida, sin importar lo difícil que ha sido afrontar cada una de ellas, pero creo que ha valido la pena; la verdad es que es muy duro estar alejado de la familia durante tanto tiempo. He llegado hasta este momento porque ustedes han confiado en que podía hacerlo y han formado una personalidad en mí que no se rinde hasta conseguir lo que quiere sin importar los límites. Le doy gracias a mi familia en general porque ellos han ayudado mucho mis tías, mis tíos, mis primos, todos me han ayudado incluso sin saberlo. No puedo dejar de agradecer a la familia de Flavia porque ellos me acogieron como si fuera otro hijo más desinteresadamente y a Flavia por soportarme todos estos años. Le doy muchas gracias a Nélida que ha estado desde mi niñez apoyándome y educándome muy dedicadamente; ella, Frank y su familia en general me han ayudado cada vez que los he necesitado, les estoy muy agradecido. Doy gracias también a los amigos y amigas del pre-universitario que hoy no están aquí pero que me ayudaron y me apoyaron mucho. Doy gracias a todas aquellas personas que han estado en mi vida y en algún momento han tomado mi mano para guiarme, gracias a esos que han hecho que me esfuerce más para demostrarme hasta donde soy capaz de hacer. Gracias a Julio, Iván y Yurián por ser nuestros compañeros de apartamento ustedes han demostrado ser amigos. Gracias a los tutores por su excelente trabajo, y gracias a Leandro por haber confiado en nosotros; cuando le dije a Flavia de hacer esta tesis le dije que iba a ser un reto para nosotros hacerla por su complejidad en general, pero no hubiera sido posible sin su trabajo y exigencia. Gracias al tribunal y la oponente que han hecho un buen trabajo también atentos a cada falta corrigiendo cada error.

Agradecimientos de Flavia

Le doy muchas gracias a Dios por tomarme de la mano y permitirme llegar hasta aquí.

Muchas gracias a mi familia por todo el apoyo que me dieron desde el comienzo y por confiar en mí, por confiar en qué podría lograrlo.

Gracias a mi esposo, que fue mi compañero durante todo este tiempo y lo seguirá siendo, además que pude ser también su compañera de tesis.

Muchas gracias a mis amigos, que siempre estaban para ayudarme de una forma u otra, gracias por los repasos usando las taquillas de pizarras y por el apoyo también.

Gracias también a los tutores que fueron un apoyo grande durante toda la realización de la tesis.

Gracias al tribunal y la oponente que también ayudaron a que todo saliera lo más correcto posible.

Gracias a todos los que estuvieron siempre para mí.

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. Actualmente en la Universidad de las Ciencias Informáticas (UCI) existe una herramienta con este propósito: Auctoritas 1.0, pero presenta algunas limitaciones que dificultan su funcionamiento.

Para resolver estas limitaciones se realizó la presente investigación. Se implementaron varios servicios web en el lenguaje JAVA y posee además una interfaz gráfica desarrollada en HTML, CSS y JavaScript. Como resultado se obtuvo la herramienta AUCTORITAS 2.0, capaz de proveer servicios web para la recuperación de información por medio de la utilización de un mecanismo de acceso a datos basado en ontologías (OBDA, por sus siglas en inglés). La información que proveen estos servicios está vinculada a los autores personales, autores corporativos, los vocabularios controlados existentes en la aplicación y términos autorizados; además provee una familia de servicios diseñados para un sistema de gestión de autores que se encuentra embebida dentro de Auctoritas 2.0. La aplicación fue probada exitosamente por medio de la realización de pruebas carga, de caja negra y caja blanca.

Palabras clave: control de autoridades, enfoque ontológico, término autorizado, vocabularios controlados.

Abstract

Authority control is a complex and expensive process, carried out by integrated library systems and for managing digital repositories. For author's names, authority control is referred to uniquely generate them according to predefined rules. Currently at the University of Informatics Science (UCI) there is a tool for this purpose: Auctoritas 1.0, but has some limitations that hinder its operation.

To address these limitations this research was conducted. Several web services were implemented by using Java programming language, also a web user interface was developed by using HTML, CSS and JavaScript technologies. As a result, AUCTORITAS 2.0, is capable of providing web services for the retrieval of information through the use of an Ontology-based data Access mechanism tool was obtained. The information provided by these services is linked to personal authors, corporate authors, controlled vocabularies existing in the application and authorized terms; also it provides a family of services designed for authors management system which is embedded within Auctoritas 2.0. The application was successfully tested through load testing, black box testing and white box testing.

Keywords: authority control, authorized term, controlled vocabularies, ontological approach.

Índice

Introducción 1

Capítulo 1: Fundamentos teóricos. 6

1.1 Introducción 6

1.2 Conceptos asociados 6

1.2.1 Catalogación 6

1.2.2 El control de autoridades 6

1.2.3 Epígrafes de Materia 7

1.2.4 Sistema de apoyo 7

1.2.5 Servicio Web 7

1.2.6 Datos Enlazados Abiertos 8

1.2.7 RDF 8

1.2.8 Ontología 8

1.3 Soluciones existentes 9

1.3.1 Panorama internacional 9

1.3.2 Panorama Nacional 11

1.4 Lenguajes de Programación y de Consulta 12

1.4.1 Java 12

1.4.2 SPARQL 13

1.4.3 SQL 13

1.4.4 JavaScript 13

1.4.5 HTML5 13

1.4.6 CSS3 13

1.4.7 OWL 2 14

1.5 Herramientas 14

1.5.1 Entorno de Desarrollo Integrado (IDE) 14

1.5.2 Sistema Gestor de Bases de Datos 15

1.5.3 Almacén de tripletas RDF 16

1.5.4 VIVO 1.8 16

1.5.5 Protégé 5.0.0 17
1.5.6 Tomcat 8.0.30 17
1.5.7 Maven 3.3.9 17
1.5.8 NodeJS 5.6.0 17
1.6 Marco de Trabajo (Framework) 18
1.6.1 Apache Jena 3.0.1 18
1.6.2 Spring framework 4.2.2 18
1.6.3 AngularJS 1.5.6 19
1.6.4 Bootstrap 3.2.0 19
1.7 Metodología de desarrollo 20
1.7.1 SCRUM 21
1.7.2 Agile Unified Process (AUP) 22
1.7.3 Programación Extrema (XP) 22
1.8 Conclusiones del capítulo 23
Capítulo 2: Descripción de la Herramienta AUCTORITAS 2.0 24
2.1 Introducción 24
2.2 Requisitos funcionales y no funcionales 24
2.3 Descripción de la propuesta de solución 25
2.3.1 Descripción Ontológica 26
2.3.2 Descripción de los requisitos 28
2.4 Patrón arquitectónico MVC 32
2.5 Fases del proceso de desarrollo 34
2.5.1 Fase de exploración 34
2.5.2 Fase de planificación 36
2.5.3 Fase de diseño 39
2.5.4 Fase de implementación 45
2.6 Conclusiones del capítulo 47
Capítulo 3: Pruebas de AUCTORITAS 2.0 48
3.1 Introducción 48

3.2 Fase de pruebas 48
3.3 Herramientas para la realización de pruebas unitarias 48
3.2.1 Pruebas Unitarias 49
3.2.2 Pruebas de caja negra. 52
Resultados de las pruebas 54
3.2.3 Pruebas de Carga 54
3.2.3.1 Herramientas para la realización de pruebas de carga 54
3.3 Utilización de datos de prueba 57
3.3.1 Archivo de Nombres de Autoridad de la LOC 57
3.3.2 Vocabulario de la ACM para las Ciencias de la Computación 58
3.2.3 AGROVOC tesauro multilingüe de agricultura 58
3.4 Conclusiones del capítulo 58
Conclusiones generales 59
Recomendaciones 60
Referencias Bibliográficas 61
Anexos 66
Diferencia entre metodología ágiles y pesadas 66
Historias de Usuario 66
Tarjetas CRC 72
Tareas de Ingeniería 78
Casos de Prueba 90
Pruebas de carga 98
Control de Autoridades 100
Entrevista realizada al cliente 101
Representación de los datos modelados en la ontología 102

Índice de tablas

Tabla 1 Cinco factores de la matriz de Boehm – Turner 20
Tabla 2 Historia de Usuario Listar autores corporativos 35
Tabla 3 Historia de Usuario Insertar autor personal y corporativo al conjunto de datos local 35
Tabla 4 Estimación de esfuerzo por Historia de Usuario 36
Tabla 5 Plan de duración de las iteraciones 37
Tabla 6 Plan de Entregas 38
Tabla 7 Tarjeta CRC de la clase AuthorController 45
Tabla 8 Tarea de Ingeniería 46
Tabla 9 Caso de Prueba Insertar Autor Personal en el conjunto de datos local 52
Tabla 10 Descripción de las variables 53

Índice de figuras

Fig. 1 Matriz de Boehm y Turner aplicada a la investigación 21
Fig. 2 Ecosistema de AUTORITAS 2.0 (elaboración propia) 26
Fig. 3 Modelo de la Descripción Ontológica (T-Box) (elaboración propia) 28
Fig. 4 Patrón MVC 33
Fig. 5 Patrón MVC en Spring Framework 34
Fig. 6 Patrón Experto en Información 41
Fig. 7 Patrón Creador 41
Fig. 8 Patrón Controlador en la clase "controlledTermsService" 42
Fig. 9 Patrón Bajo Acoplamiento 42
Fig. 10 Patrón Alta cohesión 43
Fig. 11 Patrón Singleton 43
Fig. 12 Patrón Strategy 44
Fig. 13 Patrón *State* en la clase "DataSourceService" 44
Fig. 14 Patrón Facade 45
Fig. 15 Prueba de Servicios ofrecidos por Auctoritas 2.0 50
Fig. 16 Pruebas a los servicios que son consumidos a través de la aplicación basada en AngularJS .. 50
Fig. 17 Resultados no satisfactorios de pruebas unitarias 51
Fig. 18 Resultados satisfactorios de pruebas unitarias 51
Fig. 19 Iteraciones en la fase de pruebas 51
Fig. 20 Prueba de carga del servicio autor corporativo 57

Introducción

Los libros han sido desde hace siglos el recurso que buscó el hombre para plasmar sus ideas, su imaginación y el conocimiento que ha adquirido en su vida con el propósito que trascienda en el tiempo y que otras generaciones venideras puedan aprender de la experiencia de sus antecesores. A lo largo de los años se han venido generando grandes cantidades de ejemplares bibliográficos. Con el advenimiento de la imprenta de Gutenberg[1] se produjo una revolución en la producción de libros.

Al aumentar la cantidad de ejemplares impresos y los libros manuscritos existentes hasta la invención de la imprenta, surge la necesidad en las bibliotecas de catalogar cada ejemplar. Las reglas de catalogación han sido definidas para permitir la organización constante de diversos materiales de la biblioteca a través del tiempo. Cutter[2] afirmó, que el primer objetivo del catálogo es facilitar la localización de un libro del que se conoce su autor, título o materia. La catalogación, es el proceso de elaborar el catálogo. Es decir, el proceso de describir los elementos informativos que permiten identificar un documento y de establecer los puntos de acceso que van a permitir recuperarlo (Garrido Arilla 1996).

Como parte del proceso de catalogación se encuentra el control de autoridades que es el proceso de unificar, mediante la utilización de una forma normalizada, los puntos de acceso de los catálogos automatizados y mostrar además las relaciones entre los distintos puntos de acceso. Es decir, supone la normalización de los nombres de personas, entidades, títulos uniformes o materias, que pueden constituir el punto de acceso principal o los secundarios de un catálogo automatizado. Su finalidad es facilitar la identificación y la recuperación de los documentos almacenados, evitando las confusiones a que se pueden prestar los homónimos, sinónimos o la variedad de nombres con los que puede ser denominado una persona, entidad, obra, tema o concepto (Pascual 1999).
El conjunto de los registros o asientos de autoridad se denomina fichero de autoridades que resulta ser una lista de nombres (personas, entidades, congresos, familias y lugares geográficos), de títulos uniformes (individuales o colectivos) o de materias (palabras clave o descriptores). Un nombre, un título uniforme o una materia, establecido como punto de acceso autorizado, constituye una autoridad (Pascual 1999).

Con el surgimiento de la Internet y la influencia de las Tecnologías de la Información y las Comunicaciones (TIC) la generación de información se ha disparado grandemente. Debido a esto han surgido varios problemas como sobrecarga de información y heterogeneidad de fuentes de información. La Web Semántica ayuda a resolver estos dos importantes problemas. Gracias a la semántica en la

[1] Johannes Gutenberg: inventó la impresión tipográfica con tipos móviles metálicos, la que dio origen al libro moderno.

[2] CHARLES AMMI CUTTER fue una de las figuras fundadoras que establecieron la estructura fundamental del registro de catálogo en medio de sus muchas otras contribuciones valiosas al campo de la catalogación.(Kushiyama 1931)

Web, el software es capaz de procesar su contenido, razonar con este, combinarlo y realizar deducciones lógicas (W3C 2016d).

La Web Semántica es muy útil para los bibliotecarios en la prestación de servicios de biblioteca eficaces, además es una herramienta extraordinaria para las bibliotecas donde se protege la información propietaria y ayuda a compartir la riqueza de conocimientos (Yadagiri y Ramesh 2013). Dio lugar a la aparición de muchas nuevas formas de controlar y estandarizar la descripción de documentos, resolver los problemas que rodean diversos sistemas de indexación, y mejorar la interoperabilidad de los registros (Leiva Mederos et al. 2013).

Hoy día varias bibliotecas en el mundo han creado un proyecto para crear un conjunto de Datos Enlazados Abiertos (LOD por sus siglas en inglés) que funciona tanto dentro de las instituciones individuales y a través de las bibliotecas para capturar y aprovechar el valor intelectual que los bibliotecarios y otros expertos de la rama van añadiendo a los recursos de información de la misma. (Warner 2015) Estos LOD se encuentran a disposición de la comunidad internacional, reduciendo costos y agilizando los procesos de gestión bibliográfica; reutilizando la información disponible, no se duplica el trabajo y se elimina la redundancia en las entradas bibliográficas.

La implementación de portales de la biblioteca con los servicios de la Web Semántica contribuye a cumplir la visión de las Bibliotecas. Las grandes colecciones de recursos de aprendizaje son descritas semánticamente adoptando diversas tecnologías que faciliten el acceso del usuario al contenido de uno o más repositorios de información. Actualmente en la Universidad de las Ciencias Informáticas existe una herramienta que brinda apoyo al control de autoridades aprovechando las fuentes de datos en forma de LOD que aportan diversas instituciones, llamada AUCTORITAS en su primera versión, pero presenta algunas limitaciones que dificultan el proceso de catalogación de materiales bibliográficos.

A través de una entrevista, el cliente de AUCTORITAS 1.0 dió a conocer sus limitaciones:

- ✓ No se integra con la plataforma semántica VIVO (véase 1.5.4 VIVO 1.8). Esta plataforma brinda la posibilidad de hacer búsquedas en una extensa red de conocimiento entre instituciones, organizaciones y agencias, las mismas contribuyen para publicar los perfiles de sus investigadores en línea.
- ✓ Además, posee consultas dependientes de la estructura de la fuente de datos atadas al código de la aplicación el cual conlleva que en caso que la base de datos cambie, el código fuente de la aplicación tendría que sufrir grandes cambios.
- ✓ No gestiona autores corporativos, solo gestiona autores personales. Al solo gestionar este tipo de autores limita el control de autoridades.

- ✓ No posee una interfaz de usuario para incorporar autores al conjunto de datos local, por lo que en caso de que se desee añadir un nuevo autor no es posible insertarlo en la base de datos.
- ✓ Tampoco es capaz de listar las fuentes de información. Si el usuario necesitara conocer las fuentes de las cuales está consumiendo la información no podría hacerlo.
- ✓ El tiempo de respuesta de los servicios web es prolongado. Esto trae como consecuencia que este proceso sea lento.
- ✓ Soporta una sola fuente de datos de autores. Esto trae como consecuencia que no se pueden consultar más de una fuente de datos reduciéndose el espectro de búsqueda.
- ✓ La búsqueda sobre los vocabularios controlados solamente es sobre los términos alternativos. Si un usuario realiza una búsqueda que coincide con uno de los términos autorizados entonces la búsqueda no devuelve ningún resultado cuando realmente debería devolver el mismo término.
- ✓ Está implementada solamente para hacer consultas sobre el vocabulario controlado brindado por la ACM. Este vocabulario solo contiene información referente a las Ciencias de la Computación por tanto limita el proceso del control de autoridades.
- ✓ La búsqueda de los términos de los vocabularios controlados solamente se logra hacer en el idioma inglés. Si el usuario quisiera realizar una búsqueda de un término en cualquier otro idioma no podría hacerlo.

Por las razones expuestas anteriormente se plantea el **problema a resolver**: ¿Cómo contribuir al control de autoridades realizado por la herramienta AUCTORITAS?

Dicho problema se enmarca en el **objeto de estudio** el control de autoridades durante el proceso de catalogación de materiales bibliográficos centrándose en el **campo de acción** la herramienta de apoyo al control de autoridades denominada AUCTORITAS.

Para dar solución al problema planteado se estableció como **objetivo general** desarrollar la versión 2.0 de la herramienta AUCTORITAS como sistema de apoyo al control de autoridades.

Como **objetivos específicos** se identificaron:

1- Definir los fundamentos teóricos y metodológicos relevantes del control de autoridades y tecnologías asociadas, para aplicarlos al sistema.
2- Describir la versión 2.0 de la herramienta AUCTORITAS a partir de la elaboración de los artefactos definidos por la metodología seleccionada y la herramienta implementada.
3- Probar la herramienta AUCTORITAS 2.0.

Para dar cumplimiento al objetivo general se definieron las siguientes **Tareas de Investigación:**

1- Análisis de tecnologías, herramientas y metodologías utilizadas en AUCTORITAS 1.0 para obtener los referentes teóricos y metodológicos necesarios con vistas a garantizar el desarrollo de la solución propuesta.

2- Elaboración de los artefactos establecidos por la metodología para la descripción de la aplicación AUCTORITAS 2.0.

3- Implementación de la aplicación AUCTORITAS 2.0.

4- Realización de pruebas a la aplicación implementada para verificar su correcto funcionamiento.

Métodos teóricos de Investigación:

- ✓ 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.
- ✓ 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.
- ✓ Modelación se utiliza para describir el funcionamiento de la aplicación AUCTORITAS 2.0 a través de diagramas para una mejor comprensión de la misma.
- ✓ Inductivo-Deductivo se utiliza para adaptar a la aplicación desarrollada el conocimiento de otros desarrolladores con soluciones que utilizan mecanismos similares.
- ✓ Sistémico para la modelación de la aplicación de AUCTORITAS 2.0 mediante la determinación de sus componentes, así como las relaciones entre ellos.

Métodos empíricos de la investigación:

- ✓ Entrevista realizada para recopilar información sobre las limitaciones de AUCTORITAS 1.0, realizada al cliente en el comienzo de la investigación.

Resultados Esperados

- ✓ Obtención de una herramienta que facilite la localización de entradas de autoridad necesarias en el proceso de catalogación de materiales bibliográficos.
- ✓ Despliegue de una herramienta que gestione la publicación de perfiles de investigadores e instituciones, que facilite la realización de diversos estudios cienciométricos y a su vez sirva como fuente de datos para AUCTORITAS.
- ✓ Optimización del funcionamiento de la herramienta AUCTORITAS y su extensión para manejar el vocabulario especializado en Ciencias Agrícolas AGROVOC.
- ✓ Resolución de todas las limitantes de AUCTORITAS 1.0.

Estructuración por capítulos

El presente trabajo consta de introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas y anexos. El documento está estructurado de la siguiente forma:

En el ***Capítulo 1. Fundamentos teóricos*** se describen los términos más importantes asociados al control de autoridades, así como elementos esenciales de las aplicaciones existentes para realizar dicho proceso. Además, se realiza el estudio de los conceptos fundamentales relacionados con el tema en cuestión. Al final, se define la metodología para guiar el desarrollo de la investigación, así como las tecnologías y herramientas que serán utilizadas para la construcción de la solución.

En el ***Capítulo 2. Descripción de la Herramienta AUCTORITAS 2.0*** se hace una descripción general de la solución propuesta, así como la definición de los requisitos funcionales y no funcionales. También se presenta la arquitectura con la que se desarrolló el mismo, y se hace referencia a los patrones de diseño empleados en la solución. Además, se muestran los artefactos definidos por la metodología de desarrollo de software seleccionada y se describe la construcción de la aplicación, donde se explican los aspectos principales de la implementación.

En el ***Capítulo 3. Pruebas de AUCTORITAS 2.0*** se realiza el diseño y ejecución de pruebas sobre dicha aplicación en busca de errores relacionados con su funcionalidad. Además, se muestran los resultados obtenidos de dichas pruebas.

Capítulo 1: Fundamentos teóricos.

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 AUCTORITAS 2.0, 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

Según (Campos Herrera 2007) la catalogación es un proceso donde se desarrolla 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.

También (Wynar, Taylor y Osborn 1985) señala que es la operación que comporta la descripción de un documento, unido a su clasificación, según unas normas y al establecimiento de unos encabezamientos que sirvan para su ordenación y posterior recuperación.

Según (Atherton, Novoa y Others 1983) define la catalogación como el proceso mediante el cual se transfieren, conforme a determinadas reglas a un registro bibliográfico ciertos datos informativos de un documento.

Luego del análisis de estas definiciones, se tomó durante el transcurso de la investigación el concepto de (Campos Herrera 2007).

1.2.2 El 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 (Díaz Rodríguez 2012).

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 (Díaz Rodríguez 2012).

Se define como punto de acceso: Un nombre, término, código, etc., bajo el cual puede buscarse, encontrarse e identificarse un registro bibliográfico o de autoridad (Calvo Poyato, Hidalgo López y Blanco Martínez 2004).
Para más detalles acerca del control de autoridades ver Control de Autoridades en los Anexos.

1.2.3 Epígrafes de Materia

Los 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 se debe 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 (Departamento de Proceso Técnico de la Biblioteca Nacional de España 2016).
También otro autor define por encabezamiento de materia la práctica catalográfica, cuyo objeto es dar cuenta y hacer accesibles las publicaciones que posea la biblioteca, mediante la indicación en la ficha de las materias de esa publicación. Se puede decir que los encabezamientos de materia son los signos que representan la materia o asunto del que trata el libro (Bibliotecas Universidad de Salamanca 2016).
Durante el transcurso de la investigación se tomó en cuenta la definición ofrecida por la Biblioteca Nacional de España.

1.2.4 Sistema de apoyo

Se define como Sistema al conjunto de reglas o principios sobre una materia racionalmente enlazados entre sí (Real Academia Española 2016b).
Se entiende como apoyo a una persona, cosa o parte de ella sobre la que se apoya otra, algo que sirve para sostener. Protección, auxilio o favor (Real Academia Española 2016a).
Tras analizar estas definiciones se puede concluir que un Sistema de apoyo (en términos informáticos) no es más que un sistema que ayuda a realizar un procedimiento específico a través de hardware y software.

1.2.5 Servicio Web

Un servicio web es un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web. (W3C 2016c)

Estos servicios proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario. Para proporcionar interoperabilidad y extensibilidad entre estas aplicaciones, y que al mismo tiempo sea posible su combinación para realizar operaciones complejas, es necesaria una arquitectura de referencia estándar (W3C 2016c).

1.2.6 Datos Enlazados Abiertos

El concepto de Datos Enlazados Abiertos (en idioma inglés "*Linked Open Data*", de ahí las siglas LOD) fue propuesto originalmente por Tim Berners – Lee 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. 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 (Yu 2011).

En 2006 Berners-Lee definió cuatro principios básicos para la publicación de Datos Enlazados Abiertos (Berners-lee 2013):

1) Usar URIs (*Uniform Resource Identifiers)* identificando los recursos de forma unívoca.
2) Usar URIs HTTP para que la gente pueda acceder a la información del recurso.
3) Ofrecer información sobre los recursos usando RDF (véase 1.2.7 RDF).
4) Incluir enlaces a otros URIs, facilitando el vínculo entre distintos datos distribuidos en la web.

1.2.7 RDF

RDF es el marco de descripción de recursos para metadatos en la Web elaborado por el *World Wide Web Consortium* (W3C). Se basa en la idea de declarar recursos usando la expresión en la forma sujeto – predicado – objeto. Esta expresión es conocida en la terminología RDF como triple o triplete. Un triplete RDF contiene tres componentes, todos con referencia en un URI (Peset, Ferrer-Sapena y Subirats-Coll 2011):

- ✓ 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.

1.2.8 Ontología

Una ontología es una especificación explícita de una conceptualización. El término es tomado de la filosofía, donde una ontología es una explicación sistemática de la Existencia. Para los sistemas basados en el conocimiento, lo que "existe" es exactamente lo que se puede representar. Por lo tanto, se puede

describir la ontología de un programa mediante la definición de un conjunto representacional de términos. En tal ontología, las definiciones asocian los nombres de entidades (por ejemplo, clases, relaciones, funciones, u otros objetos) (Gruber 1993).
También es definida como una especificación formal de una conceptualización compartida (Borst 1997). Durante la investigación se tuvo en cuenta la definición de (Gruber 1993).

1.3 Soluciones existentes

En la siguiente sección se aborda una síntesis de las soluciones existentes tanto a nivel internacional como a nivel nacional especificando sus características.

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 programa cooperativo *Name Authority Cooperative* (NACO). A través de este programa, los centros participantes aportan al Archivo de Nombres de Autoridad de la LOC[3], registros de autoridades con nombres personales, corporativos y jurisdiccionales y títulos (Library of Congress 2016b).

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 (Joint University Librarians Advisory Committee 2016).

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 (Tabares Martín, Leiva Mederos y Fernández Peña 2014).

3 La Biblioteca del Congreso de Estados Unidos (United States Library of Congress en inglés), situada en Washington D. C. y distribuida en tres edificios (el *Edificio Thomas Jefferson*, el *Edificio John Adams*, y el *Edificio James Madison*), es una de las mayores bibliotecas del mundo, con más de 158 millones de documentos.

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 (Tabares Martín, Leiva Mederos y Fernández Peña 2014).

La tendencia 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 (Díaz Rodríguez 2012).

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* (OCLC) y apoyado por la Federación Internacional de Asociaciones de Bibliotecarios e Instituciones (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. 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 (Leiva Mederos et al. 2013).

En los últimos años 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 (Díaz Rodríguez 2012).

La Biblioteca Nacional de Finlandia (en inglés National Library of Finland) posee una herramienta web llamada Skosmos, que ofrece el servicio para acceder a vocabularios controlados (tesauros) los cuales son utilizados por los indizadores que describen los documentos y los investigadores que buscan palabras clave adecuadas. Estos vocabularios se pueden acceder a través de *endpoints* que facilitan el acceso a los mismos por medio del lenguaje de consultas SPARQL. Además de una interfaz de usuario web moderno, Skosmos proporciona una API estilo REST y acceso en forma de datos enlazados a la información de los vocabularios (W3C 2014). Aunque constituye un paso de avance hacia la reutilización

de vocabularios, tiene la limitación que dichos vocabularios tienen que necesariamente estar estructurados con SKOS[4].

1.3.2 Panorama Nacional

La aplicación del control de autoridades en Cuba puede implementarse a partir de la aplicación del formato UNIMARC/Autoridades, las directrices definidas por Federación Internacional de Asociaciones de Bibliotecarios e Instituciones 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 (Díaz Rodríguez 2012).

La herramienta AUCTORITAS 1.0 es 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 (Ruano Alvarez y Calzadilla Reyes 2015).

Consideraciones sobre las soluciones 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, ninguno de ellos realiza el control de autoridades de manera que unifique autores personales, autores corporativos y vocabularios controlados. 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. Además, no todos tienen la capacidad de integrarse con aplicaciones externas.

Tabla 1 Comparación entre las soluciones existentes

Nombre de la solución	**Autores Personales**	**Autores Corporativos**	**Vocabularios controlados**	**Servicios Web para la integración con aplicaciones externas**
Name Authority Cooperative	x	x	-	-
Hong Kong Chinese Authority Name	x	x	-	-
AUTHORIS	x	x	x	-

[4] 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.

Fichero Virtual Internacional de Autoridades	x	x	-	-
Skosmos	-	-	x	x
AUCTORITAS 1.0	x	-	x	x

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 (Alegsa 2010).

1.4.1 Java

Como lenguaje de programación para computadores, Java se introdujo a finales de 1995. Muchos expertos opinan que Java es el lenguaje ideal para aprender la informática moderna, porque incorpora conceptos de un modo estándar, mucho más sencillo y claro que otros lenguajes. Java es simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico(García de Jalón et al. 2000). Java es un lenguaje de programación y plataforma de computación primero lanzado por Sun Microsystems en 1995. Hay muchas aplicaciones y sitios web que no funcionan a menos que haya instalado Java, y más cada día se crean. Java es rápido, seguro y fiable. Desde ordenadores portátiles a los centros de datos, consolas de juegos a los superordenadores científicos, los teléfonos móviles a Internet, Java está en todas partes (ORACLE 2016).

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

- ✓ La versión 1.0 de la herramienta AUCTORITAS fue implementada sobre Java.
- ✓ Su arquitectura neutra, permite que las aplicaciones desarrolladas en este lenguaje no dependan del tipo de CPU 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 conectores (*drivers*), que posibiliten su integración con otras tecnologías, tales como: jena-arq para la consulta de datos en formato RDF y conectores JDBC[5] para la conexión con el sistema de base de datos PostgreSQL.

[5] *Java Database Connectivity*, más conocida por sus siglas JDBC, es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

1.4.2 SPARQL

SPARQL es la clave estándar para exponer los datos en la Web Semántica. Con la tecnología de consulta SPARQL, las personas pueden centrarse en lo que quieren conocer, sin tener en cuenta la tecnología de la base de datos o el formato utilizado para almacenar esos datos. Debido a que las consultas SPARQL expresan objetivos de alto nivel, es fácil extenderlos a orígenes de datos inesperados, o incluso transferirlos a nuevas aplicaciones. Pretender usar la Web Semántica sin SPARQL es como pretender usar una base de datos relacional sin SQL. SPARQL hace posible consultar información desde bases de datos y otros orígenes de datos en sus estados primitivos a través de la Web (W3C 2008).

1.4.3 SQL

SQL es un lenguaje de consulta diseñado específicamente para el acceso a Sistemas de Gestión de Bases de Datos Relacionales (SGBDR). Este lenguaje, fue utilizado para realizar consultas a los datos de autores personales almacenados en un servidor PostgreSQL.

Las principales ventajas que aporta SQL son (Quintana et al. 2008):

- ✓ Su enorme difusión pues es empleado en la gran mayoría de los sistemas.
- ✓ Su elevada expresividad. Por ejemplo, operaciones que cuestan semanas de duro esfuerzo en ser desarrolladas en un lenguaje de programación tradicional pueden ser realizadas con SQL en tan solo unos minutos.

1.4.4 JavaScript

JavaScript es un lenguaje interpretado que se utiliza para múltiples propósitos, pero solo considerado como un complemento hasta ahora. Una de las innovaciones que ayudó a cambiar el modo en que se observa JavaScript fue el desarrollo de nuevos motores de interpretación, creados para acelerar el procesamiento de código. Esta mejorada capacidad permitió superar viejas limitaciones de rendimiento y confirmar el lenguaje JavaScript como la mejor opción para la web. (Gauchat 2012)

1.4.5 HTML5

HTML es el lenguaje de marcado estándar utilizado para crear páginas web y sus elementos forman los componentes básicos de todos los sitios web(W3C 2016e). HTML5 no es una nueva versión del antiguo lenguaje de etiquetas, ni siquiera una mejora de esta ya antigua tecnología, sino un nuevo concepto para la construcción de sitios web y aplicaciones en una era que combina dispositivos móviles, computación en la nube y trabajos en red(Gauchat 2012).

1.4.6 CSS3

Hojas de Estilo en Cascada (Cascading Style Sheets), es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada

la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos. CSS se utiliza para dar estilo a documentos HTML y XML, separando el contenido de la presentación. Los estilos definen la forma de mostrar los elementos HTML y XML. CSS permite a los desarrolladores web controlar el estilo y el formato de múltiples páginas web al mismo tiempo (W3C 2016b).

1.4.7 OWL 2

El W3C[6] *Web Ontology Language* (OWL) es un lenguaje de Web Semántica diseñado para representar el conocimiento abundante y complejo de las cosas, los grupos de cosas, y las relaciones entre las cosas. Los archivos OWL son conocidos como ontologías, pueden ser publicados en la *World Wide Web* y referirse a otras ontologías o ser referidos por una de ellas (W3C 2012).

1.5 Herramientas

1.5.1 Entorno de Desarrollo Integrado (IDE)

Un entorno de desarrollo integrado o IDE (acrónimo en inglés de *Integrated Development Enviroment*), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien, puede utilizarse para varios. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes (Fuentes y Troya 2009).

1.5.1.1 InteliJ Idea 2016

InteliJ Idea provee un conocimiento profundo del código. Analiza el código, en busca de conexiones entre los símbolos a través de todos los archivos y los idiomas del proyecto. Con esta información se proporciona asistencia en profundidad de codificación, la navegación rápida, análisis de errores inteligente, y refactorizaciones. Contiene un editor de código centrado en el desarrollo, posee una interfaz de usuario ergonómica. Posee varios marcos de trabajo para realizar pruebas como JUnit, TestNG, Spock, etc. Tiene varias herramientas que posibilitan el trabajo con bases de datos desde ejecución y edición de consultas de bases de datos hasta editar los esquemas en una interfaz visual (JetBrains 2016b). Se integra con marcos de trabajo como Java EE, GWT/Vaadin, JBoss, Play, Grails,

[6] El Consorcio World Wide Web (W3C) es un consorcio internacional donde las Organizaciones Miembro, el personal a tiempo completo y el público en general, trabajan conjuntamente para desarrollar estándares Web (W3C 2008).

App Servers/Clouds, entre otros. Se integra además con tecnologías web como JavaScript, HTML/CSS, AngularJS, React y Node.js (JetBrains 2016a).

1.5.2 Sistema Gestor de Bases de Datos

Los Sistemas Gestores de Base de Datos (SGBD) fueron diseñados para gestionar grandes volúmenes de información, tanto la definición de estructuras para el almacenamiento como los mecanismos para la gestión de los datos. Estos permiten a los usuarios definir, crear y mantener la base de datos y proporcionar un acceso controlado en todo momento. También es el software que permite la utilización y la actualización de los datos almacenados en una o varias bases de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez. (Torres Torres et al. 2015)

1.5.2.1 PostgreSQL 9.4

PostgreSQL es un Sistema de Gestión de Bases de Datos Objeto – Relacional, distribuido bajo licencia BSD[7] 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 (Lozano 2016).

PostgreSQL utiliza una arquitectura 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 (PostgreSQL 2013):

- ✓ Es una base de datos 100% ACID[8].
- ✓ Posee integridad referencial.
- ✓ Realiza copias de seguridad en caliente.
- ✓ Es Unicode.
- ✓ Permite múltiples métodos de autentificación.
- ✓ Además de sus ofertas de soporte, cuenta con una importante comunidad de profesionales de PostgreSQL, de los que los centros de desarrollos pueden obtener beneficios y contribuir.
- ✓ Acceso encriptado vía SSL.

[7] La licencia BSD es la licencia de software otorgada principalmente para los sistemas BSD (*Berkeley Software Distribution*), un tipo del sistema operativo *Unix-like*. Es una licencia de software libre permisiva. Esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público.

[8] ACID, conformado por las siglas provenientes de Atomicity, Consistency, Isolation y Durability. En español, Atomicidad, Consistencia, Aislamiento y Durabilidad, son un conjunto de propiedades necesarias para que un conjunto de instrucciones, sean consideradas como una transacción en un sistema de gestión de bases de datos.

- ✓ Completa documentación.
- ✓ Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows 32/64bit.

PostgreSQL posee las siguientes ventajas:

- ✓ 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.

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 Virtuoso Open Source 7.2.2.4

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 (Virtuoso Open Source 2016):

- ✓ 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.

1.5.4 VIVO 1.8

VIVO es una plataforma semántica de acceso abierto que permite descubrir la investigación y el saber técnico en las múltiples disciplinas y extremos administrativos, por medio de perfiles profesionales vinculados e información relacionada. Asimismo, su adopción facilita la colaboración entre personas no solo en el ámbito interno de las organizaciones, sino entre los diferentes sectores. VIVO y otras aplicaciones compatibles producen una extensa red de conocimiento entre instituciones, organizaciones y agencias, las que en sus búsquedas contribuyen al trabajo colaborativo, las sinergias y a la apertura del conocimiento (Wiki Duraspace 2016).

1.5.5 Protégé 5.0.0

Protégé es una plataforma libre, de código abierto que proporciona una comunidad de usuarios cada vez mayor con un conjunto de herramientas para la construcción de modelos de dominio y las aplicaciones basadas en el conocimiento con las ontologías. La arquitectura *plug-in* de Protégé se puede adaptar para crear aplicaciones simples y complejas basadas en ontologías (Stanford Center for Biomedical Informatics Research 2016).

1.5.6 Tomcat 8.0.30

El software Apache Tomcat® es una implementación de código abierto de Java Servlets[9], Java Server Pages, tecnologías Java WebSocket y Java Expression Language. El proyecto Apache Tomcat pretende ser una colaboración de los mejores desarrolladores de su categoría en todo el mundo. Es una herramienta poderosa que impulsa a aplicaciones de gran escala en todo el mundo a través de una amplia gama de industrias y organizaciones (The Apache Software Foundation 2016).

1.5.7 Maven 3.3.9

Maven es una herramienta de gestión de proyectos de software. Basado en el concepto de un modelo de objeto de proyecto (POM), Maven puede gestionar la compilación, generación de informes y documentación de un proyecto a partir de una pieza central de la información (Apache Maven Project 2016a). El principal objetivo de Maven es permitir a un desarrollador comprender el estado completo de desarrollo en el menor período de tiempo. Para alcanzar este objetivo hay varias áreas de preocupación que Maven intenta abordar (Apache Maven Project 2016b):

- ✓ Haciendo que el proceso de construcción fácil.
- ✓ Proporcionar un sistema de construcción uniforme.
- ✓ Proporcionar la información del proyecto de calidad.
- ✓ Proporcionar directrices para las mejores prácticas de desarrollo.
- ✓ Permitiendo la migración transparente a nuevas versiones.

1.5.8 NodeJS 5.6.0

Node.js es un entorno de ejecución de JavaScript incorporado en el motor JavaScript V8 de Chrome. Utiliza un modelo de no- bloqueo orientado a eventos de Entrada/ Salida (por sus siglas en inglés “I/O”) que hace que sea ligero y eficiente. El ecosistema de Node.js, npm, es el mayor ecosistema de librerías de código abierto en el mundo (Node.js Foundation 2016).

[9] El **servlet** es una clase en el lenguaje de programación Java, utilizada para ampliar las capacidades de un servidor.

1.6 Marco de Trabajo (Framework)

Marco de Trabajo (*del inglés "Framework"*) se define como "un conjunto de componentes físicos y lógicos estructurados de tal forma que permiten ser reutilizados en el diseño y desarrollo de nuevos sistemas de información". En una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado, a partir de una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. Los marcos de trabajo contienen patrones y buenas prácticas que apoyan el desarrollo de un producto y un proceso con calidad. Lo anterior permite vislumbrar la importancia de adoptar un Marco de Trabajo y cómo su selección debe ser una de las actividades relevantes al inicio de todo proceso de desarrollo software (Guerrero 2014).

1.6.1 Apache Jena 3.0.1

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. Provee un ambiente de programación para RDF, RDFS y OWL, SPARQL e incluye un motor de inferencia basado en reglas (W3C 2016a).

1.6.2 Spring framework 4.2.2

Spring ayuda a los equipos de desarrollo de todo el mundo a construir de manera simple, portátil, sistemas y aplicaciones de forma rápida y flexible basadas en JVM[10]. Permite escribir código limpio, que puede ser probado con los componentes de la infraestructura de su elección y llevar a cabo cualquier tarea. Spring proporciona un modelo de programación abierta integral, coherente, ampliamente entendido y bien soportado. Una de sus principales características es la inyección de dependencias (Pivotal Software 2016a).

1.6.2.1 Spring Security 4.1.1

Spring Security es un marco de trabajo que se centra en proporcionar la autenticación y autorización para aplicaciones Java. Como todos los proyectos Spring, el poder real de Spring Security se encuentra

10 JVM: máquina virtual Java es una máquina virtual de proceso nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el bytecode Java), el cual es generado por el compilador del lenguaje Java.

en la facilidad con que se puede ampliar para satisfacer los requerimientos del cliente Alguna de sus características:

- ✓ Apoyo integral y extensible tanto para autenticación y autorización.
- ✓ Protección contra ataques como la fijación de sesión, el clickjacking, etc.
- ✓ Integración API servlet.
- ✓ Integración opcional con Spring Web MVC.

1.6.2.2 Spring Boot 1.3.5

Spring Boot hace que la creación de aplicaciones con calidad sea más fácil. Este framework puede crear aplicaciones con el mínimo esfuerzo. La mayoría de las aplicaciones basadas en Spring Boot no necesitan mucha configuración. Alguna de sus características (Pivotal Software 2016b):

- ✓ Crea aplicaciones autónomas de Spring.
- ✓ Contiene un servidor Tomcat Embebido para correr las aplicaciones sin tener que compilar un archivo WAR.
- ✓ Proporcionar características listas para la producción, tales como las métricas, comprobaciones de estado y la configuración externalizada.

1.6.2.3 Spring Data

La misión de Spring Data es proporcionar un modelo de programación basado en Spring familiar y consistente para el acceso de datos al tiempo que conserva las características especiales del almacén de datos subyacente. Esto hace que sea fácil de usar tecnologías de acceso a datos, bases de datos relacionales y no relacionales, y servicios de datos basados en la nube. Este es un proyecto global que contiene muchos sub-proyectos que son específicas de una base de datos determinada. Los proyectos se desarrollan mediante el trabajo conjunto de muchas compañías y desarrolladores (Pivotal Software 2016c).

1.6.3 AngularJS 1.5.6

AngularJS es un framework MVC JavaScript creado por Google para construir adecuadamente la arquitectura y la aplicación web fácil de mantener. AngularJS se basa en la filosofía de que el código declarativo es mejor que el código imperativo para la construcción de interfaces de usuario y unir los diferentes componentes de las aplicaciones web. Permite decorar el HTML con etiquetas especiales que se sincroniza con el JavaScript (Jain, Mangal y Mehta 2014).

1.6.4 Bootstrap 3.2.0

Bootstrap es el marco de trabajo HTML, CSS, y JavaScript más popular en la web. Bootstrap hace que el desarrollo front-end web más rápido y más fácil. Está hecho para personas de todos los niveles,

dispositivos de todas las formas, y los proyectos de todos los tamaños. Bootstrap muestra excelentes resultados en la personalización de los front-end (Bootstrap team 2016).

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 (Zacchiroli 2011).

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 (Kasiak y Godoy 2012).

En los Anexos se muestra una comparación entre las metodologías ágiles y robustas según (Letelier y Penadés 2006). (Véase Diferencia entre metodología ágiles y pesadas)

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, El método de Boehm y Turner plantea 5 criterios fundamentales mediante los que se estará valorando el proyecto; estos son: tamaño del equipo, criticidad del producto, dinamismo de los cambios, cultura del equipo y personal con que se cuenta. Cada uno de esos criterios tiene elementos que lo discriminan y por tanto se tienen en cuenta a la hora de seleccionar uno u otro enfoque, 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 (Boeras Velázquez et al. 2012). A continuación, se muestra en una tabla, la descripción de los factores de la matriz de Boehm – Turner, con su descripción y los resultados de aplicarlos a la presente investigación.

Tabla 2 Cinco factores de la matriz de Boehm – Turner

Factores	Descripción	Valores resultantes
Tamaño (cantidad de integrantes en el equipo de desarrollo)	Los métodos ágiles son más fáciles de introducir, ejecutar y gestionar en los equipos pequeños. Los equipos de menos de 10 se desempeñan mejor con un enfoque ágil dado, el cual: ✓ Facilita la co – localización física de los demás miembros. ✓ Permite la comunicación a través de los debates cara a cara, que pueden apoyar el conocimiento no escrito (tácito) por conversaciones. A medida que crece el tamaño del equipo, si se sigue el principio ágil, se requiere de técnicas adicionales para escalar con eficacia, lo cual demanda más trabajo y habilidad.	2 integrantes

Criticidad (pérdidas por concepto de falla en el sistema)	Se refiere a la consecuencia de un fallo en el sistema. El enfoque ágil es más adecuado para aplicaciones triviales, donde el fallo del sistema resulta en una pérdida de conveniencia (como la pérdida de tiempo si un juego se bloquea o trabajo si un procesador de textos falla); pero no es recomendable para aplicaciones críticas para una misión o para la vida.	Utilidad
Dinamismo (% de probabilidad de cambios)	Responde a la interrogante: ¿cuán dinámico (cambiante) es el proyecto?, ¿qué porcentaje de los requisitos son propensos a cambiar durante el proyecto? Si es probable que cambien como mínimo el 50% de los requerimientos, las puntuaciones indican una metodología ágil.	42%
Personal (habilidades del equipo de desarrollo)	Para que un proyecto ágil se desarrolle sin problemas, es recomendable una baja proporción de los desarrolladores principiantes (nivel 1) y una alta proporción de intermedios (nivel 2) y expertos (nivel 3). Si el equipo tiene un mayor porcentaje de principiantes (y por tanto, un menor porcentaje de personal con más experiencia) entonces el enfoque robusto es el más apropiado.	Proporción de 50% principiantes
Cultura (% de prosperidad)	Si el proyecto tiende a prosperar en una cultura donde el equipo se sienta cómodo y motivado al tener muchos grados de libertad, se recomienda una metodología ágil. En cambio, es más adecuada una robusta si prospera en una cultura donde los desarrolladores prefieren tener sus roles definidos por políticas y procedimientos claros.	60%

En la imagen debajo de estas líneas 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.

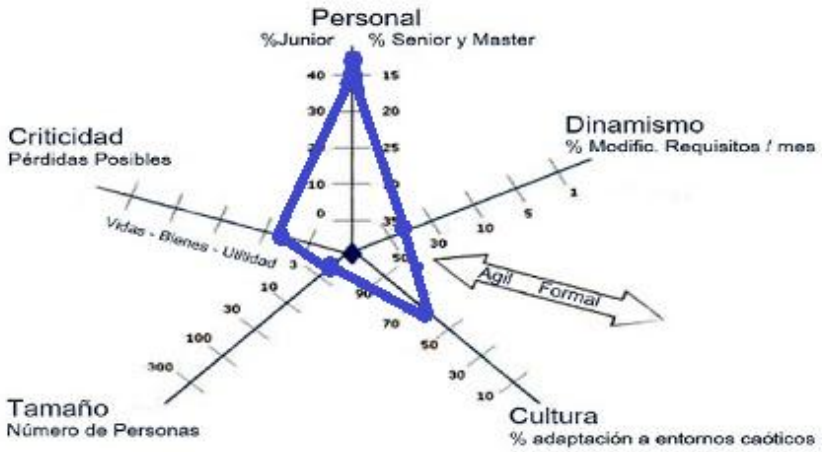

Fig. 1 Matriz de Boehm y Turner aplicada a la investigación

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 (Godoy et al. 2014).

1.7.2 Agile Unified Process (AUP)

El AUP es un acercamiento 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 (Figueroa, Solís y Cabrera 2011):

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

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 (Kasiak y Godoy 2012). XP propone 5 fases para el proceso de desarrollo de software, exploración, planificación, diseño, implementación y pruebas. 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. XP propone (Letelier y Penadés 2006).

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 tres 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.
- ✓ Da lugar a una programación organizada.
- ✓ Facilita los cambios.
- ✓ El cliente tiene el control sobre las prioridades.

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. A partir de este estudio se define la metodología de desarrollo ágil XP para regir el proceso de desarrollo de software, los diferentes lenguajes de programación y las herramientas utilizadas para el desarrollo de la aplicación AUCTORITAS 2.0. Como resultado del análisis de diferentes vías y sistemas para realizar el control de autoridades, se constató que las soluciones existentes no presentan todas las características necesarias para ofrecer una solución completa al problema planteado.

Capítulo 2: Descripción de la Herramienta AUCTORITAS 2.0

2.1 Introducción

En el presente capítulo 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 y no funcionales

La herramienta contará con los siguientes requisitos funcionales:

RF 1) Listar autores personales.
RF 2) Listar autores corporativos.
RF 3) Listar vocabularios controlados.
RF 4) Listar término autorizado.
RF 5) Listar Fuentes de Datos (*Datasources*) locales por Autores Personales.
RF 6) Listar Fuentes de Datos (*Datasources*) locales por Autores Corporativos.
RF 7) Insertar autor personal al conjunto de datos local.
RF 8) Buscar autor personal en el conjunto de datos local.
RF 9) Eliminar autor personal del conjunto de datos local.
RF 10) Modificar autor personal en el conjunto de datos local.
RF 11) Insertar autor corporativo al conjunto de datos local.
RF 12) Buscar autor corporativo en el conjunto de datos local.
RF 13) Eliminar autor corporativo del conjunto de datos local.
RF 14) Modificar autor corporativo en el conjunto de datos local.
RF 15) Insertar usuario.
RF 16) Modificar usuario.
RF 17) Buscar usuario.
RF 18) Eliminar usuario.
RF 19) Autenticar usuario.
RF 20) Mostrar las Fuentes de Datos (*Datasources*) de Autores Personales.
RF 21) Mostrar las Fuentes de Datos (*Datasources*) de Autores Corporativos.
RF 22) Mostrar Insertar Autor Personal en el conjunto de datos local.
RF 23) Mostrar Buscar Autor Personal en el conjunto de datos local.
RF 24) Mostrar Modificar Autor Personal en el conjunto de datos local.
RF 25) Mostrar Eliminar Autor Personal dado un identificador en el conjunto de datos local.
RF 26) Mostrar Insertar Autor Corporativo en el conjunto de datos local.

RF 27) Mostrar Buscar Autor Corporativo en el conjunto de datos local.
RF 28) Mostrar Modificar Autor Corporativo en el conjunto de datos local.
RF 29) Mostrar Eliminar Autor Corporativo dado un identificador en el conjunto de datos local.
RF 30) Mostrar Insertar usuario.
RF 31) Mostrar Modificar usuario dado un identificador.
RF 32) Mostrar Buscar usuario.
RF 33) Mostrar Eliminar usuario dado un identificador.

La herramienta contará con los siguientes requisitos no funcionales:

Requisitos de apariencia

- ✓ El idioma que se utilizará será inglés.

Requisitos de usabilidad

- ✓ El usuario debe tener dominio básico del idioma inglés.

Requisitos de eficiencia

- ✓ Tiempo de respuesta: la aplicación AUCTORITAS 2.0 debe tener un tiempo de respuesta menor a tres segundos.
- ✓ La aplicación debe ser capaz de soportar hasta 20 peticiones concurrentes por segundo en un margen de 10 segundos.

Requisitos de Hardware

- ✓ El servidor debe contar al menos con el hardware: 8gb de memoria RAM y un procesador core-i5 de segunda generación.

Requisitos de software

- ✓ Para que la aplicación funcione debe estar instalado en el servidor la máquina virtual de java 8 y un Sistema Gestor de Bases de Datos.
- ✓ Del lado del cliente es necesario tener instalado un navegador web, se recomienda Mozilla Firefox 46 o Google Chrome 46.

2.3 Descripción de la propuesta de solución

La propuesta de solución se nombra AUCTORITAS 2.0; consiste en la versión siguiente de AUCTORITAS 1.0, la solución dispone de servicios web a aplicaciones externas para la realización de consultas a datos de vocabularios controlados, de autores personales y de autores corporativos; teniendo como ventaja que se pueden añadir diversas fuentes de datos independientemente de su tipo (bases de datos relacionales, APIs Sparql o servicios web REST). Las consultas a estas fuentes de datos no están sujetas al código de la aplicación gracias al uso de una descripción ontológica para gestionar estas fuentes de datos; además se pueden añadir mediante una aplicación web autores personales al conjunto de datos local. Véase Fig.2.

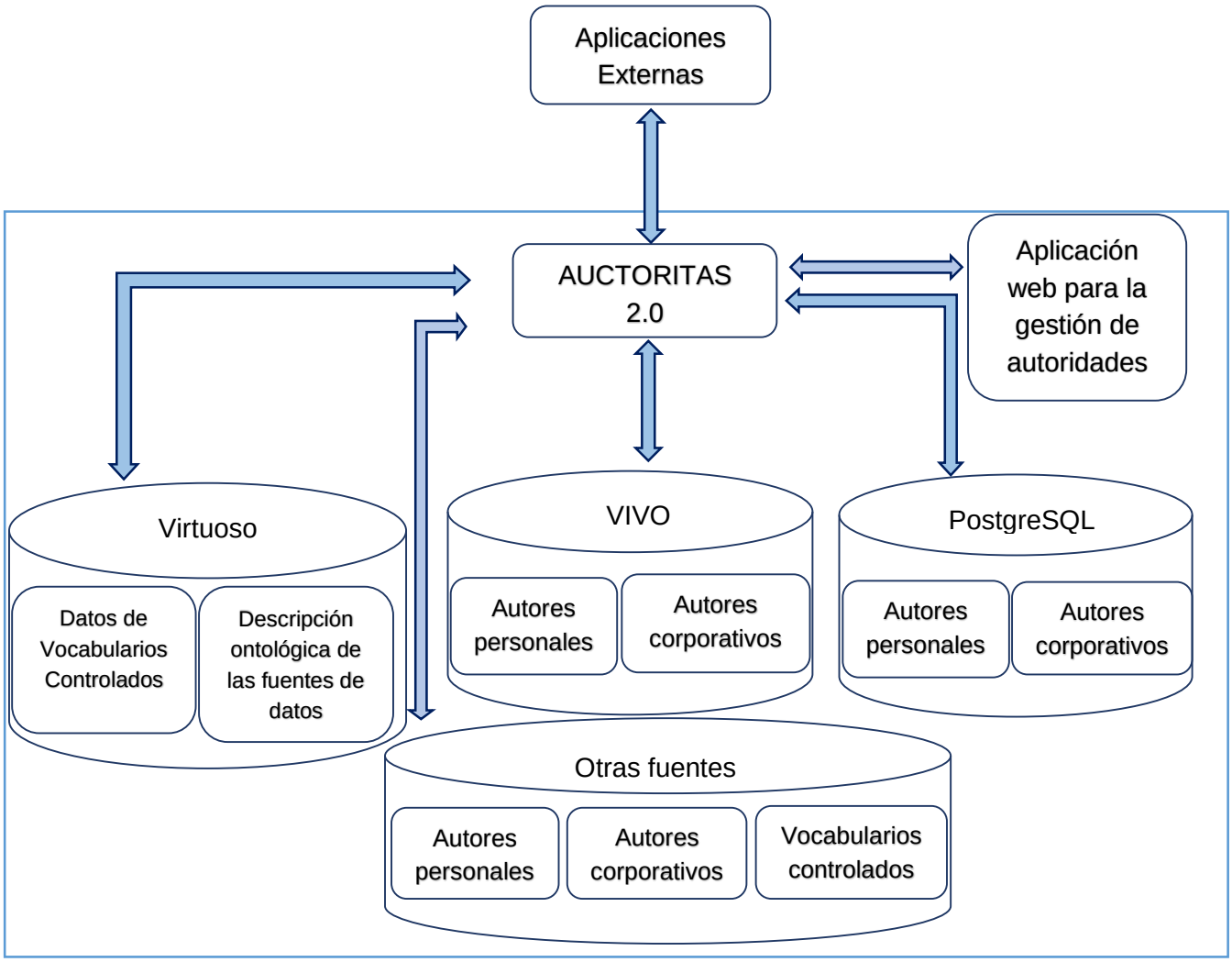

Fig. 2 Ecosistema de AUTORITAS 2.0 (elaboración propia)

Estos servicios web pueden ser consumidos por cualquier aplicación independientemente del lenguaje en que hayan sido desarrolladas. La aplicación cuenta con un sistema de seguridad que garantiza la integridad y la fiabilidad de la información acerca del control de autoridades.

2.3.1 Descripción Ontológica

El "acceso a datos basado en ontologías" (en inglés "*Ontology-Based Data Access*") (OBDA), es un enfoque importante para el usuario final orientado al acceso directo de los datos. OBDA provee un acceso semántico a bases de datos a través de una ontología, mientras mantiene los datos originales en los almacenes. Una virtud de las ontologías es que permite a los expertos de dominio solicitar la información que necesitan en sus propios términos sin tener en cuenta la manera en que se almacena en la fuente (Kharlamov et al. 2015). El enfoque OBDA resulta ser muy útil en todos los escenarios en los que se dificulta el acceso a datos de una manera unificada y coherente. Esto puede ocurrir por varias

razones. Por ejemplo, las bases de datos pueden haber sido objeto de diversas manipulaciones durante los años, a menudo para optimizar el uso de aplicaciones, y pueden haber perdido su diseño original. Pueden haber sido distribuidas o reproducido sin un diseño coherente, por lo que la información resulta estar dispersos en varias (quizá heterogéneos) fuentes de datos independientes, y los datos de origen tienden a ser redundante y mutuamente inconsistentes(Calvanese et al. 2011). Este mecanismo proporciona al usuario un punto de acceso claro y una vista conceptual unificada sin ambigüedades (Calvanese et al. 2016).

La descripción ontológica que se utiliza está formada por tres *Concepts* que son: *DataSource*, *Connection*, *Concept* (véase Fig. 3).

Cada instancia de *DataSource* posee un identificador (URI) que representa una fuente de datos consumida por AUCTORITAS. La instancia de *DataSource* está relacionada con la instancia *Connection* a través de la propiedad (*Object Property*) *has*. La instancia de la clase *Connection* Posee las siguientes propiedades (*data properties*):

- ✓ Endpoint: una cadena de texto que contiene la ruta de la fuente de datos que atiende las consultas.
- ✓ User: una cadena de texto que contiene el nombre se usuario para aquellas fuentes de datos que necesiten autenticación.
- ✓ Password: una cadena de texto que contiene la contraseña para aquellas fuentes de datos que necesiten autenticación.

Cada instancia *DataSource* está relacionado con una o varias instancias *Concept* a través de la propiedad (*objectProperty*) *composedBy*. Las instancias *Concept* son representaciones abstractas de la información almacenada en las fuentes de datos. Cada instancia de *Concept* posee las siguientes propiedades (*data properties*):

- ✓ type: es una cadena de texto que contiene un discriminador acerca de la información que el concepto hace referencia.
- ✓ name: una cadena de texto que contiene el nombre del Concept en lenguaje natural.
- ✓ mappedTo: es una cadena de texto que contiene una consulta sintáctica en un lenguaje de consulta. Teniendo en cuenta una regla para el consumo de los datos que consiste en identificar en que sección de la consulta se encuentra(n) el/los parámetro(s) a requerir y sustituirlo(s) por los términos: “param1” y “param2”. Además, para un funcionamiento correcto de la aplicación se debe especificar el “from” en cada propiedad mappedTo.

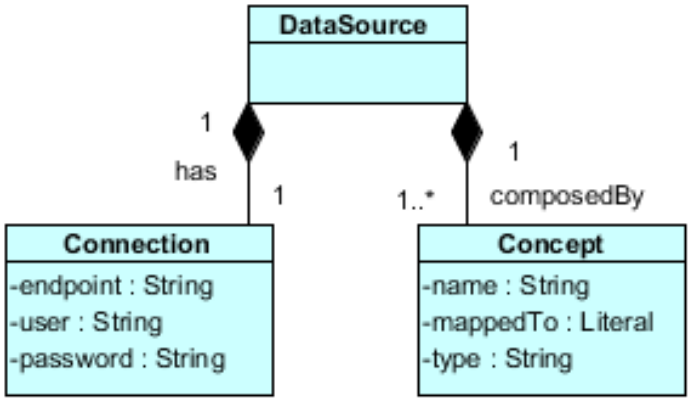

Fig. 3 Modelo de la Descripción Ontológica (T-Box[11]) (elaboración propia)

2.3.2 Descripción de los requisitos

Listar Autores Personales

Este servicio web posibilita la visualización de una lista de Autores Personales de los cuales se muestra:

- ✓ Nombre.
- ✓ Apellidos
- ✓ Autoridad.
- ✓ URI.

Recibiendo los datos especificados por el usuario que son nombre (*name*) o apellido (*lastname*). Ninguno de estos parámetros es obligatorio, pero como mínimo uno de ellos debe ser especificado. Además, debe especificar un parámetro inicio (*start*) que especifica a partir de qué valor el usuario desea ver los resultados y un parámetro que define la cantidad de respuestas que desea ver (*limit*). Este servicio devuelve un archivo XML.

Listar Autores Corporativos

Este servicio web posibilita la visualización de una lista de Autores Corporativos de los cuales se muestra:

- ✓ Nombre.
- ✓ Autoridad.
- ✓ URI.

Recibiendo los datos especificados por el usuario que son nombre (*name*) o una etiqueta (*label*) (si existe). Ninguno de estos parámetros es obligatorio, pero como mínimo uno de ellos debe ser

[11] Los términos ABox y TBox se utilizan para describir dos tipos diferentes de declaraciones en ontologías. Las declaraciones TBox describen una conceptualización, de un conjunto de conceptos, propiedades y relación entre estos conceptos. ABox son instancias de las clases TBox; el ABox contiene los elementos descritos conceptualmente en el TBox.

.

especificado. Además, debe especificar un parámetro inicio (*start*) que especifica a partir de qué valor el usuario desea ver los resultados y un parámetro que define la cantidad de respuestas que desea ver (*limit*). Este servicio devuelve un archivo XML.

Listar Fuentes de Datos (Datasources) locales por Autores Personales y Listar Fuentes de Datos (Datasources) locales por Autores Corporativos.

Este es un servicio web que devuelve un archivo en formato JSON con una lista correspondiente a las diferentes fuentes de datos contenidas en la descripción ontológica, donde esta fuente de se corresponde con las fuentes de datos locales. Además, estas fuentes contienen autores Personales o Corporativos respectivamente.

Insertar autor personal al conjunto de datos local e Insertar autor corporativo al conjunto de datos local.

Estos servicios permiten al cliente insertar un autor en la fuente de datos que él seleccionó. También el usuario tiene que proporcionar el nombre del autor (*name*) que va a insertar, apellido (*lastname*) en el caso de los autores personales, URI (*uri*), los datos de autoridad (*authority*) y un número que representa una posición de una fuente de datos (este número es equivalente a la posición de las fuentes de datos en la respuesta ofrecida por el servicio “listar fuentes de datos”). El sistema devuelve un mensaje de notificación en caso de que se haya modificado correctamente y en caso contrario devuelve un mensaje con el error ocurrido.

Buscar autor personal en el conjunto de datos local y Buscar autor corporativo en el conjunto de datos local.

Estos servicios permiten al cliente buscar un autor en la fuente de datos que él seleccionó. También el usuario debe proporcionar el nombre del autor (*name*) que va a buscar, apellido (*lastname*) en el caso de los autores personales, URI (*uri*), los datos de autoridad (*authority*), ninguno de estos atributos es requerido, pero al menos uno debe ser proporcionado por el cliente y además un número que representa una posición de una fuente de datos (este número es equivalente a la posición de las fuentes de datos en la respuesta ofrecida por el servicio “listar fuentes de datos”). Además, debe especificar un parámetro inicio (*start*) que especifica a partir de qué valor el usuario desea ver los resultados y un parámetro que define la cantidad de respuestas que desea ver (*limit*). El sistema devuelve una lista de autores personales o corporativos respectivamente con todos sus datos.

Eliminar autor personal del conjunto de datos local y Eliminar autor corporativo del conjunto de datos local.

Estos servicios permiten al cliente eliminar un autor en la fuente de datos que él seleccionó. También el usuario tiene que proporcionar la URI (identificador) (*uri*) del autor tanto para autores personales como para autores corporativos respectivamente y además un número que representa una posición de una fuente de datos (este número es equivalente a la posición de las fuentes de datos en la respuesta ofrecida por el servicio "listar fuentes de datos"). El sistema devuelve un mensaje de notificación en caso de que se haya modificado correctamente y en caso contrario devuelve un mensaje con el error ocurrido.

Modificar autor personal en el conjunto de datos local y Modificar autor corporativo en el conjunto de datos local.

Estos servicios permiten al cliente modificar un autor en la fuente de datos que él seleccionó. También el usuario debe proporcionar el nombre actualizado (*name*) del autor que va a modificar, apellido actualizado (*lastname*) en el caso de los autores personales, URI antigua (*uri*), los datos de autoridad actualizados (*authority*), la nueva URI (*newuri)*, solamente es requerido ingresar la URI que identifica el autor y además un número que representa una posición de una fuente de datos (este número es equivalente a la posición de las fuentes de datos en la respuesta ofrecida por el servicio "listar fuentes de datos"). El sistema devuelve un mensaje de notificación en caso de que se haya modificado correctamente y en caso contrario devuelve un mensaje con el error ocurrido.

Mostrar las Fuentes de Datos (Datasources) de Autores Personales y Mostrar las Fuentes de Datos (Datasources) de Autores Corporativos.

Estos requisitos funcionales son desarrollados en la aplicación web utilizando el marco de trabajo AngularJS. Los mismos devuelven una lista con las diferentes fuentes de datos sobre las cuales pueden realizar operaciones como eliminar, buscar, modificar o insertar un autor que puede ser personal o corporativo respectivamente.

Mostrar Insertar Autor Personal en el conjunto de datos local y Mostrar Insertar Autor Corporativo en el conjunto de datos local

Estos requisitos funcionales son desarrollados en la aplicación web utilizando el marco de trabajo AngularJS. Según la información proporcionada por el usuario que debe ser nombre, apellidos en el caso de los autores personales, autoridad y URI el sistema inserta un autor en la fuente de datos local y muestra un mensaje que ha sido insertado con éxito, en caso contrario se muestra un mensaje de error.

Mostrar Buscar Autor Personal en el conjunto de datos local y Mostrar Buscar Autor Corporativo en el conjunto de datos local.

Estos requisitos funcionales son desarrollados en la aplicación web utilizando el marco de trabajo AngularJS. Según la información proporcionada por el usuario que debe ser nombre, apellidos en el caso

de los autores personales, autoridad y URI el sistema busca autores en la fuente de datos local, devolviendo una lista con los autores. Ninguno de los parámetros solicitados es obligatorio, pero tiene que especificarse al menos uno de ellos.

Mostrar Modificar Autor Personal en el conjunto de datos local y Mostrar Modificar Autor Corporativo en el conjunto de datos local.

Estos requisitos funcionales son desarrollados en la aplicación web utilizando el marco de trabajo AngularJS. Según la información actualizada proporcionada por el usuario de los campos nombre, apellidos en el caso de los autores personales, autoridad y además se debe proporcionar la URI que contenía el autor anteriormente. El sistema muestra un mensaje que ha sido modificado con éxito, en caso contrario se muestra un mensaje de error.

Mostrar Eliminar Autor Personal dado un identificador en el conjunto de datos local y Mostrar Eliminar Autor Corporativo dado un identificador en el conjunto de datos local

Estos requisitos funcionales son desarrollados en la aplicación web utilizando el marco de trabajo AngularJS. El usuario proporciona una URI (identificador) y el sistema elimina el autor de la fuente de datos. El sistema muestra un mensaje que ha sido eliminado con éxito, en caso contrario se muestra un mensaje de error.

Mostrar Insertar Usuario

Este requisito funcional es desarrollado en la aplicación web utilizando el marco de trabajo AngularJS. El usuario proporciona un nombre, apellidos y contraseña. El sistema muestra un mensaje que ha sido insertado con éxito, en caso contrario se muestra un mensaje de error.

Mostrar Modificar usuario dado un identificador.

Este requisito funcional es desarrollado en la aplicación web utilizando el marco de trabajo AngularJS. Donde el usuario selecciona de una lista de usuarios el que va a modificar. En el caso de haberse modificado el sistema muestra un mensaje de éxito.

Mostrar Buscar usuario.

Este requisito funcional es desarrollado en la aplicación web utilizando el marco de trabajo AngularJS. El usuario introduce un nombre de usuario y el sistema le devuelve una lista con una serie de opciones.

Mostrar Eliminar usuario dado un identificador.

Este requisito funcional es desarrollado en la aplicación web utilizando el marco de trabajo AngularJS. El usuario selecciona de la lista de usuarios la opción de “eliminar” y se muestra un mensaje satisfactorio cuando se halla eliminado el mismo.

Insertar usuario.

Este es un servicio que permite a la aplicación que lo consume insertar un usuario en la base de datos donde se encuentra almacenada la información de los usuarios. En los parámetros se especifican nombre (*name*), apellidos (*lastname*) y contraseña (*password*). El sistema devuelve un mensaje de satisfacción en caso de que se haya modificado correctamente y en caso contrario devuelve un mensaje con el error ocurrido.

Buscar usuario.

Este es un servicio que permite a la aplicación que lo consume buscar un usuario en la base de datos donde se encuentra almacenada la información de los usuarios. Se le debe especificar el nombre (*name*) del usuario a buscar. El sistema debe devolver una lista con el nombre, apellidos, usuario y el id del usuario.

Eliminar usuario.

Este servicio permite a la aplicación o usuario que lo consume eliminar un elemento de la base de datos que contiene la información acerca de los usuarios del sistema. Se le debe especificar el id (*user*) del usuario. El sistema devuelve un mensaje de satisfacción en caso de que se haya modificado correctamente y en caso contrario devuelve un mensaje con el error ocurrido.

Autenticar usuario.

Esta página permite por medio del uso del marco de trabajo Spring Security proteger aquellos servicios que brinda la herramienta. Debido a que se utiliza un marco de trabajo especializado en seguridad reduce la posibilidad de ataque a la herramienta. Se le debe especificar un usuario y contraseña existentes en la base de datos que contiene los usuarios, si es satisfactoria la autenticación entonces la aplicación redireccionará a la página principal de la misma, en caso que la información no sea válida la página requerirá que revise los datos.

2.4 Patrón arquitectónico MVC

La idea de los patrones como una manera de presentar, compartir y reutilizar el conocimiento sobre los sistemas de software es ahora ampliamente utilizado. Se propusieron los patrones arquitectónicos en la década de 1990 bajo el nombre de "estilos arquitectónicos", con una serie de cinco volúmenes de manuales sobre la arquitectura de software orientada patrón-publicados entre 1996 y 2007 (Sommerville 2011).

El patrón Modelo-Vista-Controlador (MVC) surge con el objetivo de reducir el esfuerzo de programación, necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos, a partir de

estandarizar el diseño de las aplicaciones. El patrón MVC es un paradigma que divide las partes que conforman una aplicación en el Modelo, las Vistas y los Controladores, permitiendo la implementación por separado de cada elemento, garantizando así la actualización y mantenimiento del software de forma sencilla y en un reducido espacio de tiempo. A partir del uso de marcos de trabajo (del inglés *"frameworks")* basados en el patrón MVC se puede lograr una mejor organización del trabajo y mayor especialización de los desarrolladores y diseñadores (Romero Fernández y González Díaz 2012).

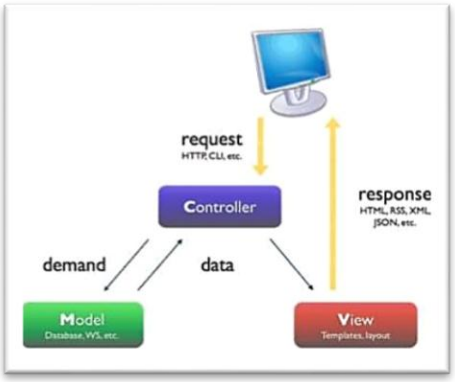

Fig. 4 Patrón MVC

El Patrón MVC en Spring Framework funciona de la siguiente forma(Pivotal Software 2016d)**:**

- ✓ Todas las peticiones HTTP se canalizan a través del *front controller*. En casi todos los *frameworks* MVC que siguen este patrón, el *front controller* no es más que un *servlet* cuya implementación es propia del *framework*.
- ✓ El *front controller* averigua, normalmente a partir de la URL, a qué controlador hay que llamar para servir la petición. Para esto se usa un HandlerMapping.
- ✓ Se llama al controlador, que ejecuta la lógica de negocio, obtiene los resultados y los devuelve al *servlet*, encapsulados en un objeto del tipo Model. Además, se devolverá el nombre lógico de la vista a mostrar.
- ✓ Un ViewResolver se encarga de averiguar el nombre físico de la vista que se corresponde con el nombre lógico del paso anterior.
- ✓ Finalmente, el *front controller* (el DispatcherServlet) redirige la petición hacia la vista, que muestra los resultados de la operación realizada.

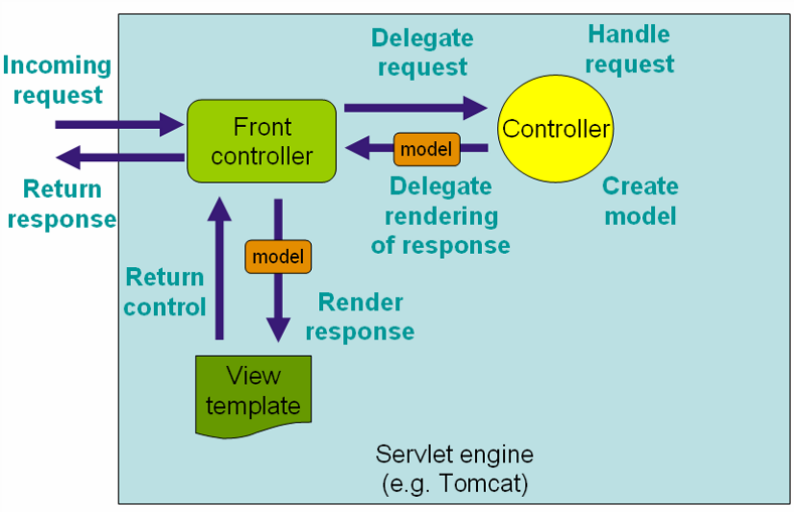

Fig. 5 Patrón MVC en Spring Framework

2.5 Fases del proceso de desarrollo

2.5.1 Fase de exploración

En esta fase los clientes describen las Historias de Usuario definiendo las características que van a tener cada una de ellas.

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas (Letelier y Penadés 2006).

A continuación, se explica cada uno de los datos que deben ser llenados en la HU:

- ✓ **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.
- ✓ **Observaciones:** información de interés.
- ✓ **Prototipo de Interfaz:** imagen de la funcionalidad desarrollada.

A continuación, se describe la estructura de algunas Historias de Usuario con prioridad alta para el negocio, las restantes se muestran en los Anexos específicamente en la sección Historias de Usuario.

Tabla 3 Historia de Usuario Listar autores corporativos

Historia de usuario	
Número: 1	**Nombre**: Listar autores corporativos.
Cantidad de modificaciones a la Historia de Usuario: 1	
Usuario: Gestor de autoridades.	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 3
Riesgo en desarrollo: Alto	**Puntos reales:** 3
Descripción: permite exportar un archivo en formato XML con el resultado de una búsqueda con nombre, autoridad o URI; dicho archivo contiene una lista de autores corporativos con su nombre, URI y registros de autoridad.	
Observaciones: para crear esta lista la aplicación consulta cada una de las fuentes de datos	

Tabla 4 Historia de Usuario Insertar autor personal y corporativo al conjunto de datos local

Historia de usuario	
Número: 3	**Nombre**: Insertar autor personal al conjunto de datos local e Insertar autor corporativo al conjunto de datos local.
Cantidad de modificaciones a la Historia de Usuario: 0	
Usuario: Gestor de Autoridades	**Iteración asignada:** 1
Prioridad en negocio: Alto	**Puntos estimados:** 1
Riesgo en desarrollo: Medio	**Puntos reales:** 1
Descripción: Inserta autores personales o autores corporativos, a los autores personales se le inserta URI(identificador), nombre, apellidos y autoridad; en el caso de los autores corporativos se le inserta URI(identificador), nombre y autoridad.	
Observaciones: Todos los campos son obligatorios.	

2.5.2 Fase de planificación

En la fase de planificación se realiza la estimación del esfuerzo que causará la implementación de cada HU, como en la metodología ágil XP las métricas son libres, puede utilizarse cualquier criterio definido para medir el desempeño del proyecto en cuestión (Letelier y Penadés 2006)

Estimación de esfuerzo por Historias de Usuarios

Para la realización de la aplicación propuesta se efectuó una estimación de esfuerzo por cada una de las HU identificadas, donde a continuación se muestran los resultados. Los puntos de estimación están en un rango de 1 a 3.

Tabla 5 Estimación de esfuerzo por Historia de Usuario

Historia de usuario	Puntos de estimación (semanas)
Listar autores personales.	3
Listar autores corporativos	2
Listar vocabularios controlados y Listar término autorizado.	1
Listar Fuentes de Datos (*Datasources*) locales por Autores Personales. y Listar Fuentes de Datos (*Datasources*) locales por Autores Corporativos.	2
Insertar autor personal al conjunto de datos local e Insertar autor corporativo al conjunto de datos local.	2
Buscar autor personal en el conjunto de datos local y Buscar autor corporativo en el conjunto de datos local.	1
Eliminar autor personal del conjunto de datos local y Eliminar autor corporativo del conjunto de datos local.	1
Modificar autor personal del conjunto de datos local y Modificar autor corporativo del conjunto de datos local.	1
Insertar y modificar usuario	2
Buscar y eliminar usuario	1
Autenticar usuario	1
Mostrar las Fuentes de Datos (*Datasources*) de Autores Personales y Mostrar las Fuentes de Datos (*Datasources*) de Autores Corporativos.	2
Mostrar Insertar Autor Personal en el conjunto de datos local, Mostrar Buscar Autor Personal en el conjunto de datos local, Mostrar Modificar Autor Personal en el conjunto de datos local,	1
Mostrar Eliminar Autor Personal dado un identificador en el conjunto de datos local, Mostrar Eliminar Autor Corporativo dado un identificador en el conjunto de datos local.	1

Mostrar Insertar usuario y Mostrar Buscar usuario.	1
Mostrar Eliminar usuario dado un identificador.	1
Mostrar Insertar Autor Corporativo en el conjunto de datos local, Mostrar Buscar Autor Corporativo en el conjunto de datos local y Mostrar Modificar Autor Corporativo en el conjunto de datos local.	1

Plan de iteraciones

Luego de haber identificado las HU y de realizar una estimación del tiempo requerido para la realización de cada una, se realizó la planificación estableciendo las iteraciones que necesarias antes de entregar el sistema.

- Iteración 1: tiene como objetivo la implementación de las HU: 1, 2, 3, 4, 5, 6, 7, 8.
- Iteración 2: tiene como objetivo la implementación de las HU: 9, 10, 11.
- Iteración 3: tiene como objetivo la implementación de las HU: 12, 13, 14, 15, 16, 17.

Plan de duración de las iteraciones

En el plan de duración de las iteraciones se muestran las historias de usuario que son implementadas en cada una de las iteraciones, así como la duración estimada y el orden de implementación de cada una de ellas (Fernández Escribano 2002).

Tabla 6 Plan de duración de las iteraciones

Iteración	Orden de las Historias de Usuario a implementar	Estimación (semana)
1	✓ Listar autores personales. ✓ Listar autores corporativos. ✓ Listar vocabularios controlados y Listar término autorizado. ✓ Listar Fuentes de Datos (*Datasources*) locales por Autores Personales. y Listar Fuentes de Datos (*Datasources*) locales por Autores Corporativos. ✓ Insertar autor personal al conjunto de datos local e Insertar autor corporativo al conjunto de datos local. ✓ Buscar autor personal en el conjunto de datos local y Buscar autor corporativo en el conjunto de datos local. ✓ Eliminar autor personal del conjunto de datos local y Eliminar autor corporativo del conjunto de datos local. ✓ Modificar autor personal del conjunto de datos local y Modificar autor corporativo del conjunto de datos local.	13

2	✓ Insertar y modificar usuario ✓ Buscar y eliminar usuario ✓ Autenticar Usuario	4
3	✓ Mostrar las Fuentes de Datos (*Datasources*) de Autores Personales y Mostrar las Fuentes de Datos (*Datasources*) de Autores Corporativos. ✓ Mostrar Insertar Autor Personal en el conjunto de datos local, Mostrar Buscar Autor Personal en el conjunto de datos local, Mostrar Modificar Autor Personal en el conjunto de datos local. ✓ Mostrar Eliminar Autor Personal dado un identificador en el conjunto de datos local, Mostrar Eliminar Autor Corporativo dado un identificador en el conjunto de datos local. ✓ Mostrar Insertar usuario y Mostrar Buscar usuario. ✓ Mostrar Eliminar usuario dado un identificador. ✓ Mostrar Insertar Autor Corporativo en el conjunto de datos local, Mostrar Buscar Autor Corporativo en el conjunto de datos local y Mostrar Modificar Autor Corporativo en el conjunto de datos local.	7

Plan de entrega

El plan de entrega es el compromiso final del equipo de desarrollo con los clientes. Es de vital importancia ya que la entrega tardía de la solución repercute negativamente en el desarrollo del producto, creando insatisfacción en el cliente(Fernández Escribano 2002). En esta etapa se definió para cada iteración una fecha de inicio y fin.

Tabla 7 Plan de Entregas

Iteración	Historia de usuario	Estimación (semana)	Fecha
1	✓ Listar autores personales. ✓ Listar autores corporativos ✓ Listar vocabularios controlados y Listar término autorizado.	13	4/1/2016- 4/4/2016

	✓ Listar Fuentes de Datos (*Datasources*) locales por Autores Personales y Listar Fuentes de Datos (*Datasources*) locales por Autores Corporativos. ✓ Insertar autor personal al conjunto de datos local e Insertar autor corporativo al conjunto de datos local. ✓ Buscar autor personal en el conjunto de datos local y Buscar autor corporativo en el conjunto de datos local. ✓ Eliminar autor personal del conjunto de datos local y Eliminar autor corporativo del conjunto de datos local. ✓ Modificar autor personal del conjunto de datos local y Modificar autor corporativo del conjunto de datos local.		
2	✓ Insertar y modificar usuario ✓ Buscar y eliminar usuario ✓ Autenticar Usuario	4	4/4/2016- 2/5/2016
3	✓ Mostrar las Fuentes de Datos (*Datasources*) de Autores Personales y Mostrar las Fuentes de Datos (*Datasources*) de Autores Corporativos. ✓ Mostrar Insertar Autor Personal en el conjunto de datos local, Mostrar Buscar Autor Personal en el conjunto de datos local, Mostrar Modificar Autor Personal en el conjunto de datos local. ✓ Mostrar Eliminar Autor Personal dado un identificador en el conjunto de datos local, Mostrar Eliminar Autor Corporativo dado un identificador en el conjunto de datos local. ✓ Mostrar Insertar usuario y Mostrar Buscar usuario. ✓ Mostrar Eliminar usuario dado un identificador. ✓ Mostrar Insertar Autor Corporativo en el conjunto de datos local, Mostrar Buscar Autor Corporativo en el conjunto de datos local y Mostrar Modificar Autor Corporativo en el conjunto de datos local.	7	2/5/2016- 20/6/2016

2.5.3 Fase de diseño

2.5.3.1 Patrones de diseño

Los patrones de diseño se derivan de las ideas expuestas por Christopher Alexander, quien sugirió que había ciertos patrones comunes en la construcción del diseño que eran inherentemente efectivos. El patrón es una descripción del problema y la esencia de su solución, de modo que la solución puede ser reutilizada en diferentes entornos. El patrón no es una especificación detallada. Más bien, se puede pensar en ella como una descripción del conocimiento y la experiencia acumulada, una solución de eficacia probada a un problema común (Sommerville 2011).

- ✓ Patrones *General Responsibility Assignment Software Patterns* (GRASP, por sus siglas en inglés): describen un conjunto de principios básicos de la asignación de responsabilidades a objetos. Algunos de los patrones que conforman este grupo son: experto, creador, bajo acoplamiento, alta cohesión y controlador.

- ✓ Patrones Banda de los Cuatro en español, formada por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides (GOF, por sus siglas en inglés): según su propósito, se clasifican en tres categorías: creacionales, estructurales y de comportamiento. Los patrones creacionales se encargan de la creación de instancias de los objetos. Los estructurales separan la interfaz de la implementación, además, plantean las relaciones entre clases, las combinan y forman estructuras mayores. Los patrones de comportamiento describen la comunicación entre objetos y clases.

Patrones para Asignar Responsabilidades (GRASP)

Durante el diseño de la herramienta se emplearon patrones GRASP, específicamente:

Experto: El Experto en Información se utiliza con frecuencia en la asignación de responsabilidades; es un principio de guía básico que se utiliza continuamente en el diseño de objetos. Normalmente, se les otorgan responsabilidades a las clases con la información necesaria para llevar a cabo una tarea. Indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo (Larman 2002). Se evidencia en la clase “DataSourceService” pues cuenta con la información necesaria para cumplir responsabilidades específicas.

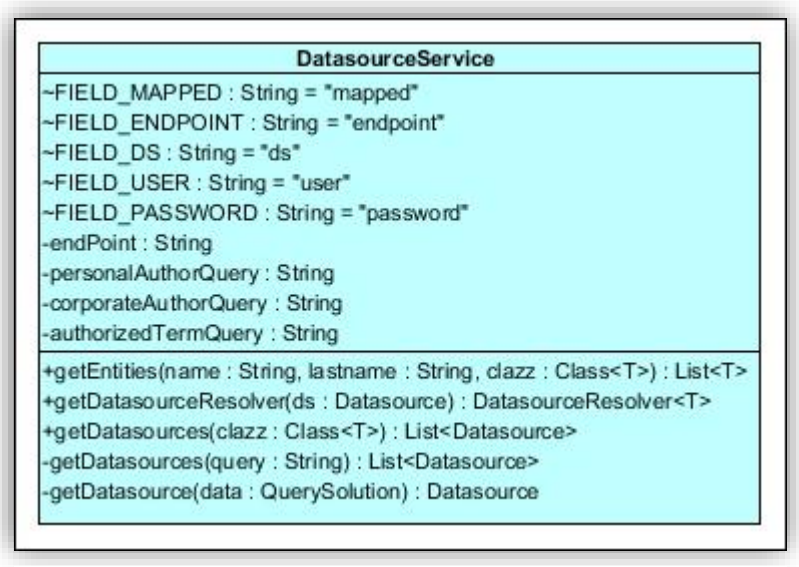

Fig. 6 Patrón Experto en Información

Creador: El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, una tarea muy común. La intención básica del patrón Creador es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Eligiéndolo como el creador se favorece el bajo acoplamiento (Larman 2002). La imagen muestra un fragmento del código que evidencia este patrón creando los objetos en cada uno de los "@Autowired"

```
@RestController
@RequestMapping("/api/author")
public class AuthorController {

    @Autowired
    private PersonalAuthorService personalAuthorService;
    @Autowired
    private CorporateAuthorService corporateAuthorService;
```

Fig. 7 Patrón Creador

Controlador: El patrón Controlador proporciona guías acerca de las opciones generalmente aceptadas y adecuadas. El controlador es una especie de fachada en la capa del dominio para la capa de la interfaz. Un error típico del diseño de los controladores es otorgarles demasiada responsabilidad. Normalmente, un controlador debería delegar en otros objetos el trabajo necesario; coordina o controla la actividad. No realiza mucho trabajo por sí mismo (Larman 2002).

```java
@RestController
@RequestMapping("/api/term")
public class ControledTermController {
    @Autowired
    private ControledTermsService controledTermsService;

    @RequestMapping(method = RequestMethod.GET)
    public List<AuthorizedTerm> getAuthorizedTerm(@RequestParam String term, @RequestParam String language, @RequestParam(required = false) String vocabulary){
        term = "\""+term+"\"";
        return controledTermsService.getAuthorizedTerms(term, language, vocabulary);
    }

}
```

Fig. 8 Patrón Controlador en la clase "controlledTermsService"

Bajo Acoplamiento: se pone en práctica, puesto que las clases poseen pocas relaciones entre sí, para que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión en las demás. Esta característica permite potenciar la reutilización y disminuye la dependencia entre las clases (Larman 2002).

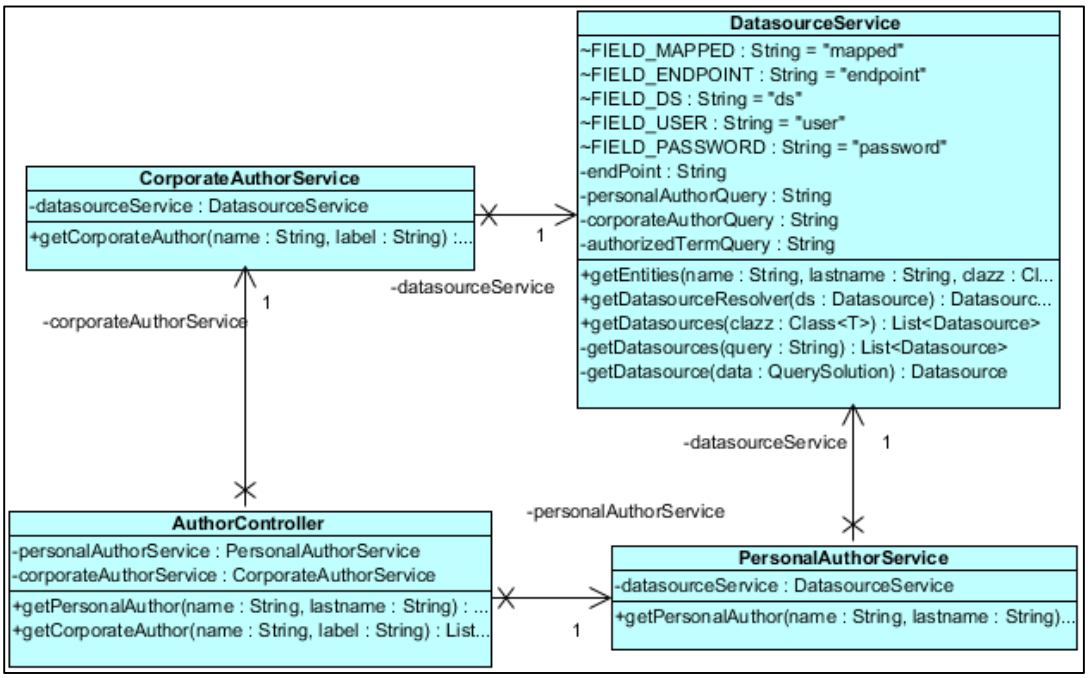

Fig. 9 Patrón Bajo Acoplamiento

Alta cohesión: La cohesión tiene que ver con la forma en la que se agrupan unidades de software en una unidad mayor. Por ejemplo, la forma en la que se agrupan funciones en una biblioteca, o la forma en la que se agrupan métodos en una clase, o la forma en la que se agrupan clases en una biblioteca, etc. Con una buena cohesión se incrementa la claridad y facilita la comprensión del diseño. Además, se simplifican el mantenimiento y las mejoras (Larman 2002).

En la clase "*DatasourceService*" se evidencia la alta cohesión entre los métodos porque el método *"getEntities"* solicita información al método *"getDatasources(Class clazz)",* este necesita información del método *"getDatasources(String query)"* y luego el método *"getEntities"* llama al método *"getDasourceResolver".*

```
public <T> List<T> getEntities(String name, String lastname, Class<T> clazz) {
    name = TextUtils.isEmpty(name) ? "" : name;
    lastname = TextUtils.isEmpty(lastname) ? "" : lastname;
    if ((name.isEmpty())&&(lastname.isEmpty()))
    return null;
    List<Datasource> datasources = getDatasources(clazz);
    List<T> entities = new ArrayList<T>();
    for(Datasource ds: datasources) {

        DatasourceResolver<T> resolver = getDatasourceResolver(ds);
        if(resolver == null) continue;

        try{
            entities.addAll(resolver
                .getElementByDynamicQuery(query, ds.getEndpoint(), ds.getUsername(),
        }catch (Exception e){
            e.printStackTrace();
        }
    }
    return entities;
}

public  <T> DatasourceResolver<T> getDatasourceResolver(Datasource ds) {


public <T> List<Datasource> getDatasources(Class<T> clazz){

private <T> List<Datasource> getDatasources(String query) {
```

Fig. 10 Patrón Alta cohesión

Patrones Gof

Singleton: Este patrón tiene como objetivo asegurar que una clase solo posee una instancia y proporcionar un método de clase único que devuelva esta instancia. En ciertos casos es útil gestionar clases que posean una única instancia (Debrauwer 2013). En la imagen se muestra un ejemplo donde la anotación "@Autowired" crea una inyección de la clase DataSourceController y creando una sola instancia de la misma, este patrón es implementado por el *framework*.

```
public class DataSourceController {
    @Autowired
    private DatasourceService datasourceService;

    @RequestMapping(value = "/listPersonalAuthor", method = RequestMethod.GET)
    public List<String> listbyPersonalAuthor(@RequestParam String type){

        List<Datasource> datasources= datasourceService.getDatasources(PersonalAuthor.class);
        List<String> finaldatasources = new ArrayList<>();
        for (Datasource ds: datasources){
            if (ds.getEndpoint().substring(0,4).contains(type)){
                finaldatasources.add(ds.getDatasource());
            }
        }
        return finaldatasources;
    }
```

Fig. 11 Patrón Singleton

Strategy: Tiene como objetivo adaptar el comportamiento y los algoritmos de un objeto en función de una necesidad sin cambiar las interacciones de este objeto con los clientes. Esta necesidad puede

ponerse de relieve en base a aspectos tales como la presentación, la eficacia en tiempo de ejecución o en memoria, la elección de algoritmos, la representación interna, etc. Aunque evidentemente no se trata de una necesidad funcional de cara a los clientes del objeto, pues las interacciones entre el objeto y sus clientes deben permanecer inmutables (Debrauwer 2013).

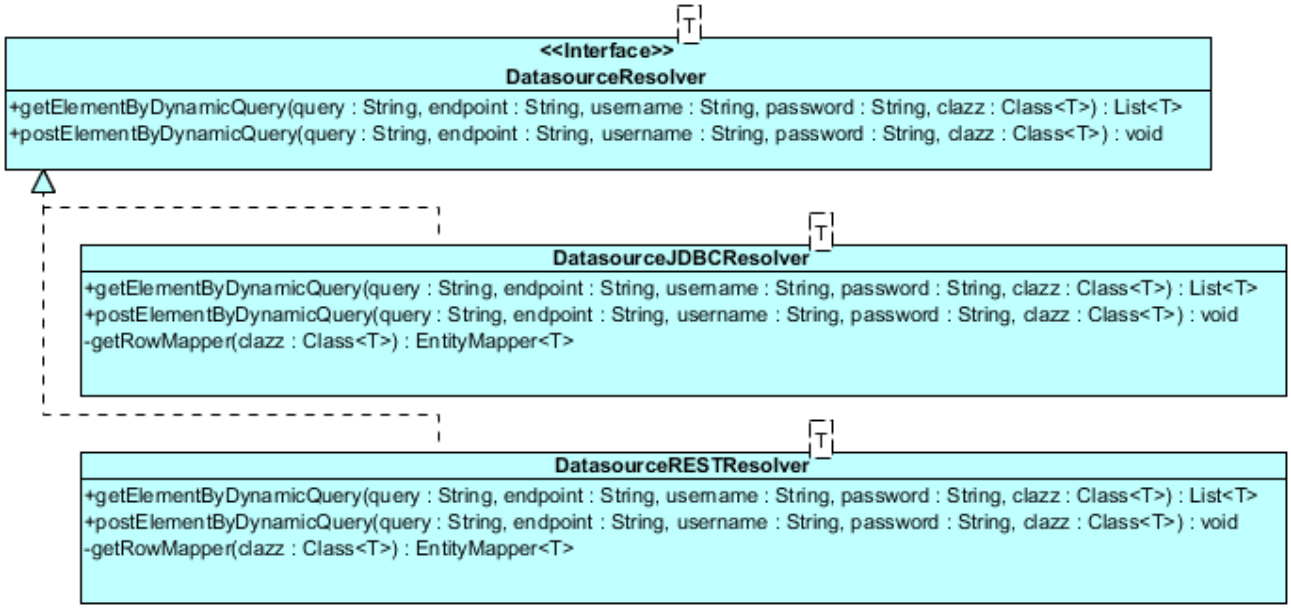

Fig. 12 Patrón Strategy

State: Permite a un objeto modificar su comportamiento a medida que su estado interno va cambiando, dando así la impresión de que el objeto "cambia de clase". Se utiliza cuando el comportamiento de un objeto depende de su estado y debe poder cambiar dinámicamente (en tiempo de ejecución) dicho comportamiento, a medida que cambie dicho estado (Debrauwer 2013).

```
public <T> DatasourceResolver<T> getDatasourceResolver(Datasource ds) {
    DatasourceResolver<T> resolver = null;
    switch (ds.getEndpoint().substring(0, 4)) {
        case "jdbc":
            resolver = new DatasourceJDBCResolver<T>();
            break;
        case "http":
            resolver = new DatasourceRESTResolver<T>();
            break;
    }
    return resolver;
}
```

Fig. 13 Patrón *State* en la clase "DataSourceService"

Facade: El objetivo del patrón es agrupar las interfaces de un conjunto de objetos en una interfaz unificada volviendo a este conjunto más fácil de usar por parte de un cliente. El patrón "Facade" encapsula la interfaz de cada objeto considerada como interfaz de bajo nivel en una interfaz única de nivel más elevado (Debrauwer 2013).

```
public interface DatasourceResolver<T> {
    List<T> getElementByDynamicQuery(String query, String endpoint, String username, String password, Class<T> clazz);
    void postElementByDynamicQuery(String query, String endpoint, String username, String password, Class<T> clazz);
}
```

Fig. 14 Patrón Facade

2.5.3.2 Tarjetas de Clase, Responsabilidad y Colaboración (CRC)

Para el diseño de las aplicaciones, la metodología XP no requiere la representación del sistema mediante diagramas de clases utilizando notación UML, en su lugar se usan otras técnicas como las tarjetas CRC. La técnica CRC propone una forma de trabajo, preferentemente grupal, para encontrar los objetos del dominio de la aplicación, sus responsabilidades y cómo colaboran con otros para realizar tareas (Vallespir 2002).

A continuación, se muestra la tarjeta CRC de una de las clases, las restantes se encuentran disponibles en los Anexos, específicamente en Tarjetas CRC.

Tabla 8 Tarjeta CRC de la clase AuthorController

Clase: AuthorController	
Descripción: Obtiene los datos necesarios para generar los servicios web: 1. Datos de autor corporativo. 2. Datos de autor personal.	
Responsabilidad	**Colaboraciones**
- Listar Autores Personales. - Listar Autores Corporativos.	- CorporateAuthorService. - PersonalAuthorService.

2.5.4 Fase de implementación

La implementación en XP tiene varias características particulares de la propia metodología. Las tareas de ingeniería generadas por las Historias de Usuario se desarrollaron siguiendo las siguientes prácticas (Fernández Escribano 2002):

- ✓ Adoptar un método de desarrollo basado en pruebas para asegurar que el código se comporta según lo esperado.
- ✓ Programación por parejas, para incrementar el conocimiento, la experiencia y las ideas.
- ✓ Asumir la propiedad colectiva del código, para que todo el equipo sea responsable de él.

2.5.4.1 Tareas de Ingeniería

Las tareas de ingeniería se usan para desglosar en actividades las HU y asignadas a los programadores para ser implementadas durante una iteración (Letelier y Penadés 2006). En ellas se especifica la fecha de inicio y fin de cada una, se nombra al programador responsable de cumplirla y se describe qué se deberá hacer en la misma. A continuación, se detallan las tareas de ingeniería que se deben ejecutar

para la Historia de Usuario 1, el resto se pueden encontrar en los Anexos en la sección Tareas de Ingeniería.

Tabla 9 Tarea de Ingeniería

Tarea	
Número de tarea: 1	**Nombre Historia de usuario:** Listar autores corporativos
Nombre de la tarea: Realizar el estudio de autor corporativo.	
Tipo de tarea: Estudio	**Puntos Estimados(días):** 1
Fecha inicio: 4/1/2016	**Fecha fin:** 5/1/2016
Programador responsable: Flavia González Barroso	
Descripción: Estudiar cada uno de los aspectos que conforman un autor corporativo.	

2.5.4.2 Estándares de codificación

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física del código.

Las realizaciones de revisiones frecuentes al código, la aplicación de forma continua de técnicas y estándares de codificación definidos, junto al empleo de buenas prácticas de programación, garantizan una alta calidad en el desarrollo de la aplicación, además de proporcionar un código legible y reutilizable.

Convenciones de nombres

En la implementación de la herramienta los nombres aparecen en idioma inglés.

Clases

Los nombres de las clases deberán ser sustantivos, en el caso de ser compuestos tendrán la primera letra de cada palabra que lo forme en mayúscula. Estos deben ser simples y descriptivos. Ejemplo: public class DatasourceService {}

Métodos

La denominación de los métodos debe ser un infinitivo demostrando así la acción que ejecutan, cuando sean compuestos tendrán la primera letra en minúscula, y la inicial de las siguientes palabras en mayúscula.

Ejemplo: insertar ()

Nomenclaturas utilizadas

CamelCase en la primera letra del identificador está en minúscula y la primera letra de las siguientes palabras concatenadas en mayúscula. Esta notación se utilizó para el nombre de los métodos.

2.6 Conclusiones del capítulo

En la realización de este capítulo se generaron los artefactos de la fase de exploración, planificación, diseño e implementación de la metodología XP. Se identificaron los requisitos funcionales, obteniéndose así las funcionalidades de AUCTORITAS 2.0 y se identificaron varios requisitos no funcionales a tener en cuenta para el desarrollo del sistema. También se realizó una descripción de cada una de las funcionalidades donde se aclaran los parámetros necesarios para su correcto funcionamiento y los resultados que devuelven cada una de ellas. Finalmente, se obtiene una descripción del patrón arquitectónico y los patrones de diseño utilizados.

Capítulo 3: Pruebas de AUCTORITAS 2.0

3.1 Introducción

El presente capítulo está orientado a las pruebas de la investigación que se llevan a cabo para dar total cumplimiento a los requisitos establecidos. Se evidencian las pruebas aplicadas a la solución con el objetivo de comprobar las funcionalidades en los diferentes escenarios y así verificar en todos los casos que los resultados sean los esperados. Para ello, se plasman las pruebas definidas por la metodología seleccionada, haciendo uso del *framework* JUnit y herramientas como LoadUI y SoapUI diseñadas para este fin.

3.2 Fase de pruebas

La fase de pruebas es la última de las fases propuestas por la Metodología XP. La verificación de los requisitos que debe cumplir la aplicación a través del análisis a las posibles combinaciones de entradas y de las salidas de datos, además de la comprobación de que el software trabaje como fue diseñado, son los objetivos que se persiguen con la realización de las pruebas al sistema. Como resultado permitirá un mayor control e identificación temprana de los defectos y fallos, de esta manera se garantiza la calidad del desarrollo con una reducción notable de los costos necesarios para corregir los errores. La calidad del sistema será medida en correspondencia con el número de no conformidades que sean detectadas (Valdez Huaraca et al. 2013).

3.3 Herramientas para la realización de pruebas unitarias

Las pruebas de unidad se enfocan en la verificación del desempeño de pequeñas unidades del diseño de software. Usando la descripción de diseño de nivel de componente como guía, importantes vías de control son probadas para cubrir los errores en el límite del módulo. La relativa complejidad de las pruebas y los errores que esas pruebas cubren están limitados por las restricciones establecidas para pruebas unitarias. La prueba de unidad se enfoca en la lógica del procedimiento interno y las estructuras de datos dentro de los límites de un componente. Este tipo de prueba puede ser llevada a cabo en paralelo para múltiples componentes (Pressman y Maxim 2015).

3.3.1 JUnit

JUnit es una herramienta para Java, desarrollada por Erich Gamma and Kent Beck, 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 (Ávila et al. 2013).

3.3.2 TestNG

TestNG es un *framework* de pruebas inspirado en JUnit y NUnit, pero introduce algunas nuevas funcionalidades que lo hacen más potente y fácil de usar, algunas como:

- ✓ Anotaciones.
- ✓ Configuración flexible de los test.
- ✓ Soporte a parámetros.
- ✓ Soporta una variedad de plugins y herramientas.

Está diseñado para cubrir todas las categorías de pruebas: unidad, funcional, fin a fin, integración, etc. Fue creada por *Cédric Beust* (Beust 2015).

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.

3.2.1 Pruebas Unitarias

Las pruebas unitarias son establecidas antes de escribir el código y son ejecutadas constantemente ante cada modificación del sistema. Los clientes escriben las pruebas funcionales para cada historia de usuario que deba validarse. En este contexto de desarrollo evolutivo y de énfasis en pruebas constantes, la automatización para apoyar esta actividad es crucial (Letelier y Penadés 2006). 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 clases de pruebas que invocan a las clases y métodos a probar, permitiendo comparar sus resultados con los esperados. A continuación, se muestran dos clases para la realización de pruebas, una de ellas con los servicios que va a ofrecer Auctoritas a los clientes y la otra se le realizan las pruebas a los servicios que son consumidos a través de la aplicación basada en AngularJS para la gestión de autores locales (ver Fig. 15 y Fig. 16).

```
    @Test
    public void corporateAuthorTest() {
        System.out.println("authors");
        String expResult = "Universidad de Ciencias Informáticas (UCI), Cuba@es-ES";
        List<CorporateAuthor> result = authorController.getCorporateAuthor("universidad","infor");
        String res=result.get(0).getName();
        assertEquals(expResult, res);
    }

    @Test
    public void vocabularyTest() {
        System.out.println("vocabulary");
        String expResult = "http://controledterms.cu/agrovoc";
        List<Vocabulary> result = vocabularyControler.getVocabulary();
        String res = result.get(0).getUri();
        assertEquals(expResult, res);
    }
    @Test
    public void controlledtermTest() {
        System.out.println("terms");
        String expResult = "Data management systems@en";
        List<AuthorizedTerm> result = controlledTermController.getAuthorizedTerm("databases","en","http://controledterms.cu/acm");
        String res = result.get(0).getTerm();
        assertEquals(expResult, res);
```

Fig. 15 Prueba de Servicios ofrecidos por Auctoritas 2.0

```
@RunWith(SpringJUnit4ClassRunner.class)
@SpringApplicationConfiguration(classes = AuctoritasApplication.class)
public class angularServicesTests {
    @Resource
    private DataSourceController dataSourceController;
    @Resource
    private LocalPersonalAuthorsController localPersonalAuthorsController;
    @Resource
    private LocalCorporateAuthorsController localCorporateAuthorsController;

    @Test
    public void dsByPersonalAuthorTest() {
        System.out.println("dataSources");
        String expResult = "http://auctoritasdescription.cu/#Postgre";
        List<String> result = dataSourceController.listbyPersonalAuthor("jdbc");
        String res = result.get(0);
        assertEquals(expResult, res);
    }
    @Test
    public void dsByCorporateAuthorTest() {
        System.out.println("dataSources");
        String expResult = "";
        List<String> result = dataSourceController.listbyCorporateAuthor("jdbc");
      //  String res = result.get(0);
        assertEquals(expResult, expResult);
    }
    @Test
    public void addlocalPersonalAuthorTest() throws Exception {

        int expResult =200;
        int result = localPersonalAuthorsController.postpersonalAuthor("mordi","Mordilon Bicho","el arte de dormir","123456789",0);
        assertEquals(expResult, result);
}
    @Test
    public void modifylocalPersonalAuthorTest() throws Exception {

        int expResult =200;
        int result = localPersonalAuthorsController.update("Flavia","Gonzalez Bicho","el arte de dormir","http://facebook.com/flavia","ht
        assertEquals(expResult, result);
```

4: Run 0: Messages 6: TODO Spring Terminal

Fig. 16 Pruebas a los servicios que son consumidos a través de la aplicación basada en AngularJS

Los resultados de las pruebas son almacenados dentro de una lista. Luego, el marco de trabajo JUnit comprueba que cada uno de los resultados obtenidos coincide con los resultados esperados y muestra en una ventana los resultados obtenidos, cuando estos son satisfactorios para todas las pruebas realizadas, se observa una línea verde en la ventana, en caso contrario aparece una línea roja (ver Fig. 17 y Fig. 18).

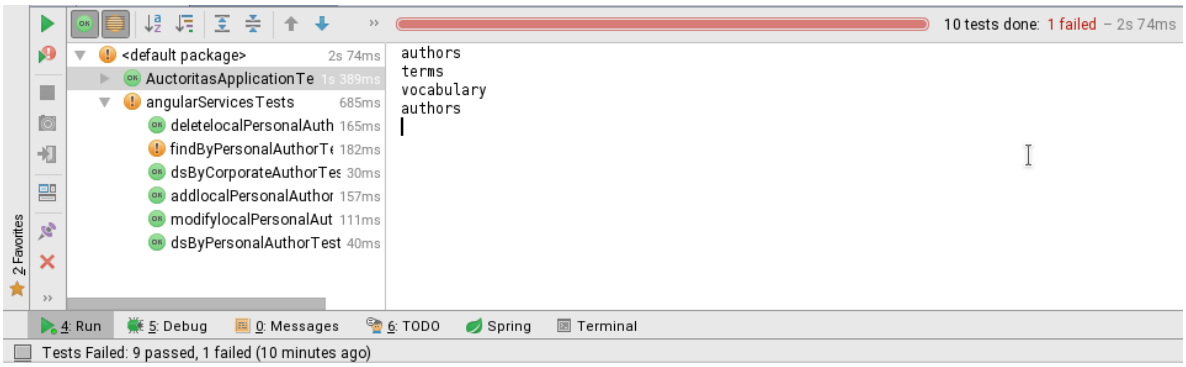

Fig. 17 Resultados no satisfactorios de pruebas unitarias

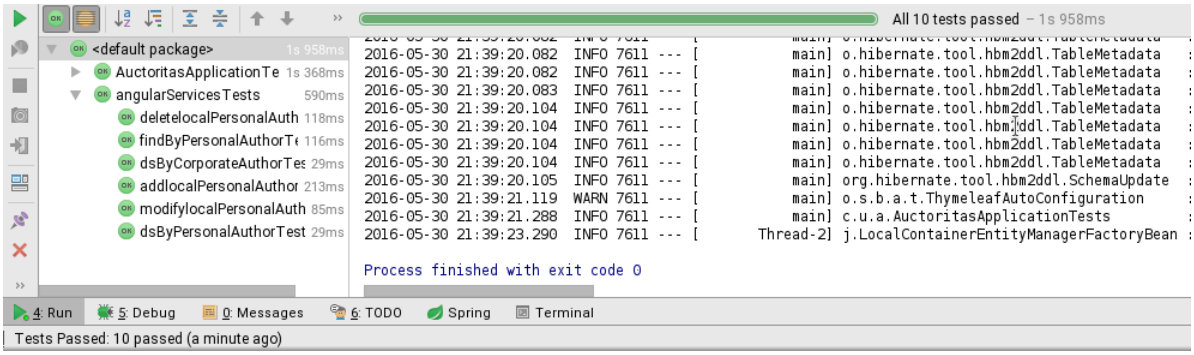

Fig. 18 Resultados satisfactorios de pruebas unitarias

Se desarrollaron un total de 15 pruebas con el JUnit. Se obtuvieron en la primera iteración un total de 3 errores que fueron subsanados. En la segunda iteración se obtuvieron resultados satisfactorios para cada prueba realizada.

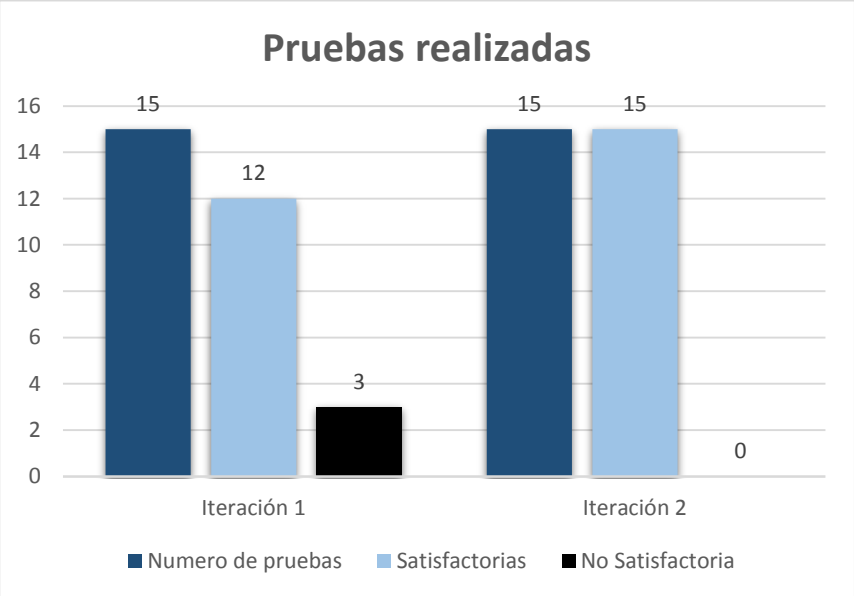

Fig. 19 Iteraciones en la fase de pruebas

3.2.2 Pruebas de caja negra.

La prueba de caja negra no es una alternativa de las técnicas de prueba de la caja blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la caja blanca. Los campos que son estudiados desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno. Donde examina qué es lo que hace, pero sin dar importancia a cómo lo hace. Define las entradas y salidas, es decir, su interfaz; en cambio, no se precisa definir ni conocer los detalles internos de su funcionamiento. Es el estudio de un módulo o elemento de un sistema, desde su parte externa (Cantone 2006).

Para la realizar estas pruebas se hizo a través de: partición equivalente que no es más que es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores, que de otro modo requerirían la ejecución de muchos casos antes de detectar el error genérico (Sommerville 2011).

Caso de Prueba

Un caso de prueba incluye un conjunto de entradas, condiciones de ejecución y resultados esperados para conseguir un objetivo particular o condición de prueba por ejemplo verificar el cumplimiento de un requisito específico (Aristegui O 2010). A continuación, se muestra un ejemplo de caso de prueba de uno de los requisitos, los restantes se encuentran en los Anexos, en la sección Casos de Prueba.

Tabla 10 Caso de Prueba Insertar Autor Personal en el conjunto de datos local

Escenario	Descripción	URI	Name	Last Name	Authority	Respuesta del sistema	Flujo central
EC 1.1 Insertar Autor Personal en el conjunto de datos local con datos correctos	*Inserta un Autor Personal en el conjunto de datos local*	*V* *http//:id1.uci.cu*	*V* *Enma*	*V* *Wealler*	*V* *Baby Dady*	*mensaje: Author inserted*	*1- Selecciona el tipo de autor a insertar (Autor Personal). 2- Seleccione la funcionalidad List Datasources. 3- luego de escoger la fuente de datos seleccione la funcionalidad (Add Author). 4- Inserte los datos correspondientes. 5 Pulsar en Add*

EC 1.2 Insertar Autor Personal en el conjunto de datos local con datos incorrectos	*Inserta un Autor Personal en el conjunto de datos local*	*V* *http//:id1.uci.cu* *V* *http//:id1.uci.cu*	*I* *7978657* *V* Emma	*V* *Wealler* *I* 23r	*V* *Baby Dady* *V* Baby Dady	*Error Incorrect Data*	*1- Selecciona el tipo de autor a insertar (Autor Personal). 2- Seleccione la funcionalidad List Datasources. 3- luego de escoger la fuente de datos seleccione la funcionalidad (Add Author). 4- Inserte los datos correspondientes. 5 Pulsar en Add*
EC 1.2 Insertar Autor Personal en el conjunto de datos local sin datos	*Inserta un Autor Personal en el conjunto de datos local*	*I*	*I*	*I*	*I*	*Insert All Data*	*1- Selecciona el tipo de autor a insertar (Autor Personal). 2- Seleccione la funcionalidad List Datasources. 3- luego de escoger la fuente de datos seleccione la funcionalidad (Add Author). 4- Inserte los datos correspondientes. 5 Pulsar en Add*

A continuación, se muestra una descripción con las variables necesarias para llevar a cabo el caso de prueba.

Tabla 11 Descripción de las variables

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	*URI*	*String*	*No*	*cadenas de texto con números y símbolos*
2	*Name*	*String*	*No*	*cadenas de texto*
3	*Last Name*	*String*	*No*	*cadenas de texto*
4	*Authority*	*String*	*No*	*cadenas de texto con números y símbolos*

Resultados de las pruebas

- ✓ En la primera iteración se probaron 11 casos de pruebas y se detectaron 2 no conformidades.
- ✓ En la segunda iteración se probaron 11 casos de pruebas y no se detectaron no conformidades.
- ✓ Todas las no conformidades detectadas fueron resueltas satisfactoriamente, quedando el sistema libre de errores.

Los elementos que se tuvieron en cuenta para la realización de la aplicación fueron:

- ✓ Interfaz sencilla.
- ✓ Cumplimiento de las funcionalidades presentadas.
- ✓ Rapidez en la obtención de la información.

3.2.3 Pruebas de Carga

El objetivo de las pruebas de rendimiento es encontrar problemas de rendimiento, cuando una aplicación bajo prueba expone características que en una situación normal no haría para una carga de trabajo específica. Es difícil construir casos de prueba de rendimiento eficaces que pueden encontrar problemas de rendimiento en un corto período de tiempo, ya que requiere ingenieros de pruebas para probar muchas combinaciones de acciones y datos para aplicaciones no triviales (Grechanik 2012).

Las pruebas de carga se llevan a cabo para verificar que la aplicación pueda cumplir con sus objetivos de rendimiento deseados; estos objetivos de rendimiento a menudo se especifican en un acuerdo de nivel de servicio. Una prueba de carga le permite medir los tiempos de respuesta, tasas de rendimiento, y los niveles de utilización de recursos, y para identificar el punto de ruptura de la aplicación, en el supuesto de que el punto de ruptura se produce por debajo de la condición de carga máxima (Meier et al. 2007).

3.2.3.1 Herramientas para la realización de pruebas de carga

Una prueba de carga se realiza generalmente para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperadas. Esta carga puede ser el número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la carga.

LoadUI 2.6.5

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. Con esta herramienta se logra optimizar el tiempo dedicado a las pruebas y reduce el tiempo de implementación de servicios REST y SOAP. Esta

herramienta permite probar la velocidad y la escalabilidad de los nuevos cambios en la API[12] en cuestión de minutos, no días. Muestran una vista previa del comportamiento de la aplicación antes de ser desplegada. Permite que los desarrolladores creen un código más fiable en cuanto a rendimiento (SmartBear 2016a).

SoapUI 4.6.1

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 (SmartBear 2016b).

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.

Se seleccionaron los servicios principales para realizarles pruebas de carga. Teniendo en cuenta el requisito no funcional que el sistema debe soportar al menos 20 peticiones concurrentes por segundo en un lapso de tiempo de 10 segundos se obtuvo los resultados que se muestran en la Tabla 11. La tabla muestra en una de sus columnas el nombre de los servicios principales de la aplicación, en las demás muestra un promedio de la cantidad de peticiones realizadas en 10 iteraciones en el espacio de tiempo que se indica en la columna.

Tabla 12 Resultado de las pruebas de carga

Servicios	**5**	**10**	**15**	**20**
	Promedio peticiones/s	Promedio peticiones/s	Promedio peticiones/s	Promedio peticiones/s
Listar autores personales	49,1	99,1	149,1	199
Listar autores corporativos	49,2	99,3	149,1	199,4
Listar vocabularios controlados	49,5	99,4	149,1	199,5
Listar término autorizado	49,2	99,4	149,2	199,1
Listar Fuentes de Datos (*Datasources*) locales por Autores Personales.	49,6	99,4	149,2	199,3

[12] 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.

Buscar autor personal en el conjunto de datos local.	49,2	99,3	149,1	199,4
Eliminar autor personal del conjunto de datos local.	49,1	99,4	149,2	199,2

Por medio de estas pruebas se comprobó que la aplicación AUCTORITAS 2.0 es capaz de soportar las peticiones concurrentes establecidas por el cliente (ver Tabla 11). Quedó demostrado que los tiempos de respuesta de los servicios que realizan las operaciones más complejas están por debajo de 1 segundo, de esta manera se le da cumplimiento al requisito no funcional de eficiencia que plantea que los tiempos de respuesta deben ser menor a tres segundos (ver tabla 12).

Tabla 13 Tiempo de respuesta promedio

Servicios	**5**	**10**	**15**	**20**
	Tiempo de respuesta en milisegundos	Tiempo de respuesta en milisegundos	Tiempo de respuesta en milisegundos	Tiempo de respuesta en milisegundos
Listar autores personales	0,101	0,100	0,100	0,100
Listar autores corporativos	0.101	0,100	0,100	0,100
Listar vocabularios controlados	0,101	0,100	0,100	0,100
Listar término autorizado	0,101	0,100	0,100	0,100
Listar Fuentes de Datos (*Datasources*) locales por Autores Personales.	0,100	0,100	0,100	0,100
Buscar autor personal en el conjunto de datos local.	0,101	0,100	0,100	0,100
Eliminar autor personal del conjunto de datos local.	0,101	0,100	0,100	0,100

A continuación, se muestra una de las pruebas de carga realizada al servicio listar autor corporativo. Las restantes se encuentran en los Anexos en la sección Pruebas de carga.

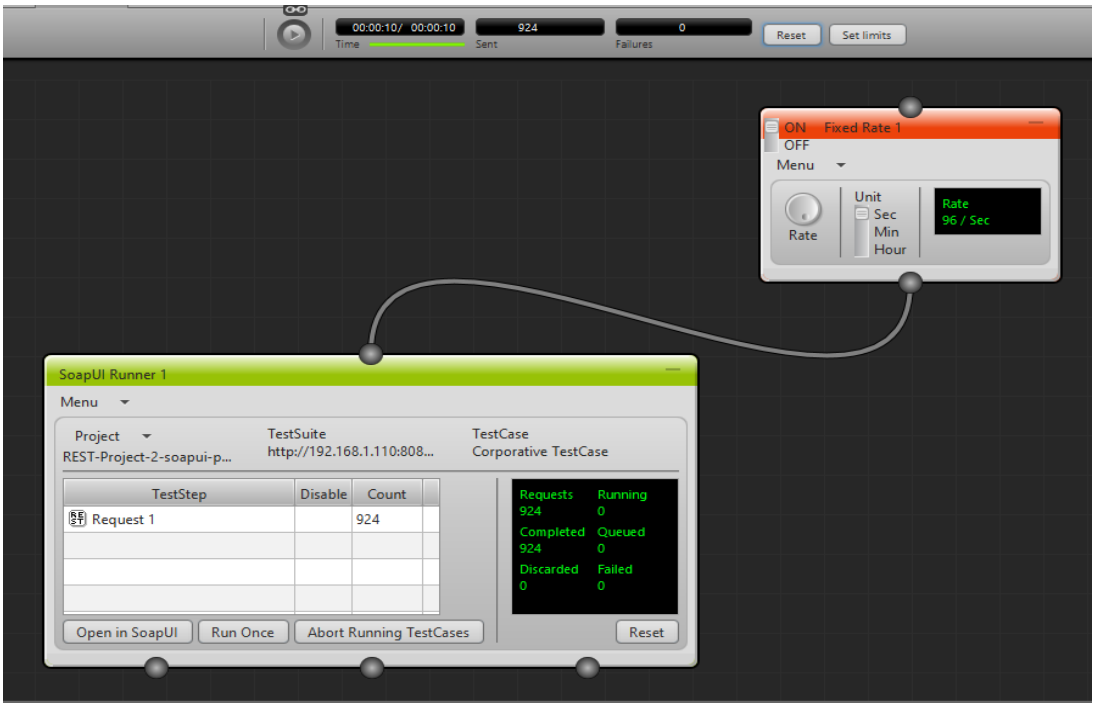

Fig. 20 Prueba de carga del servicio autor corporativo

3.3 Utilización de datos de prueba

3.3.1 Archivo de Nombres de Autoridad de la LOC

Para la validación de los servicios referentes a los Autores se utilizó el archivo de nombres de autoridad 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 (Library of Congress 2016a).

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 fue previamente descargado y procesado a través de la herramienta de Pre-procesamiento desarrollado en la versión 1.0 de Auctoritas, gracias a esto se logró reducir el tamaño del mismo y lograr almacenarlo en la Base de Datos relacional. Se realizaron pruebas sobre 10 registros de este fichero.

3.3.2 Vocabulario de la ACM para las Ciencias de la Computación

Este vocabulario contribuyó con la validación de los datos de los servicios referentes a los vocabularios controlados, 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 las pruebas), fue desarrollado como una ontología poli jerárquica 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, Word y HTML(Asociation of Computer Machinery 2015). Este archivo fue almacenado en un servidor Virtuoso, para la realización de consultas al mismo en lenguaje SPARQL desde la herramienta AUCTORITAS 2.0, lo cual permitió probar los servicios web “Listar vocabularios controlados” y “Término autorizado”.

3.2.3 AGROVOC tesauro multilingüe de agricultura

AGROVOC es un vocabulario controlado que abarca todos los ámbitos de interés de la Organización de las Naciones Unidas para la Alimentación y la Agricultura (FAO), entre ellos la alimentación, la nutrición, la agricultura, la pesca, las ciencias forestales y el medio ambiente. Lo publica la FAO y una comunidad de expertos se encarga de su edición (W3C 2013).

AGROVOC consta de más de 32000 conceptos, disponibles en hasta 23 lenguas: árabe, chino, checo, inglés, francés, alemán, hindi, húngaro, italiano, japonés, coreano, lao, malayo, persa, polaco, portugués, ruso, eslovaco, español, telugu, tailandés, turco y ucraniano (AIMS 2016).

Este archivo fue almacenado en un servidor Virtuoso, para la realización de consultas al mismo en lenguaje SPARQL desde la herramienta AUCTORITAS 2.0, lo cual permitió probar los servicios web “Listar vocabularios controlados” y “Término autorizado”.

3.4 Conclusiones del capítulo

En el capítulo quedaron presentados los elementos principales de las etapas de pruebas. A partir de los resultados obtenidos, se demuestra que la solución propuesta cumple con los requisitos planteados. Los resultados de las pruebas que fueron aplicadas verificaron una correcta ejecución del código y probaron que el sistema cumple con las funcionalidades requeridas. Se verificó que los servicios que se ofrecen poseen un buen tiempo de respuesta, aunque la herramienta maneje varias fuentes de datos y la resolución de los mismos sea compleja. Resultando para las pruebas unitarias un total de 3 no conformidades que fueron corregidas en una segunda iteración satisfactoriamente, para las pruebas de Caja Negra un total de 2 no conformidades que fueron resueltas y las pruebas de carga mostraron un rendimiento eficiente para los servicios web.

Conclusiones generales

Con la realización y culminación de la presente investigación se cumplieron los objetivos propuestos, arribando a las siguientes conclusiones:

- ✓ La implementación de la solución garantizó una aplicación funcional, que brinda un conjunto de servicios que facilitan el control de autoridades, además responde a los requisitos planteados.
- ✓ La nueva herramienta soporta más de una fuente de datos ampliando el espectro de los resultados.
- ✓ La utilización de un mecanismo OBDA (Ontology Based Data Access) elimina las consultas atadas al código de la aplicación.
- ✓ Al utilizar Spring Framework la aplicación garantiza seguridad sobre la información que brindan sus servicios.
- ✓ Las pruebas realizadas a la solución obtenida, posibilitaron comprobar la calidad de la misma utilizando un conjunto de datos reales, garantizando la entrega de un producto de software libre de errores y listo para su uso.
- ✓ La implementación de AUCTORITAS 2.0 y las pruebas realizadas a la misma contribuyen a la solución de las limitantes de AUCTORITAS 1.0.

Recomendaciones

Se recomienda para futuras versiones de la herramienta propuesta:

- ✓ Desarrollar el sistema utilizando el framework de desarrollo AngularJS v2.0, ya que es un marco de trabajo más completo que el actual utilizado. En el desarrollo de la presente versión no se utiliza porque aún no se encuentra en una versión estable, además, no se integra con el marco de trabajo Spring.
- ✓ Internacionalizar la interfaz web de manera que posea más de un idioma para que las personas que no tengan dominio del idioma inglés puedan utilizarla.

Referencias Bibliográficas

AIMS, 2016. AGROVOC tesauro multilingüe de agricultura. [en línea]. [Consulta: 25 mayo 2016]. Disponible en: http://aims.fao.org/es/agrovoc.

ALEGSA, L., 2010. Diccionario de Informática y Tecnología. [en línea]. [Consulta: 23 enero 2016]. Disponible en: http://www.alegsa.com.ar/Dic/lenguaje de programacion.php.

APACHE MAVEN PROJECT, 2016a. Welcome to Apache Maven. [en línea]. [Consulta: 23 enero 2016]. Disponible en: https://maven.apache.org/.

APACHE MAVEN PROJECT, 2016b. What is maven. [en línea]. [Consulta: 20 junio 2016]. Disponible en: https://maven.apache.org/what-is-maven.html.

ARISTEGUI O, J.L., 2010. Los casos de prueba en la prueba del software. *Revista Digital Lámpsakos*, no. 3, pp. 27-34.

ASOCIATION OF COMPUTER MACHINERY, 2015. The 2012 ACM Computing Classification System toc. [en línea]. [Consulta: 3 febrero 2016]. Disponible en: https://www.acm.org/about/class/2012.

ATHERTON, P., NOVOA, B. y OTHERS, 1983. *Manual para sistemas y servicios de información*. 1983. S.l.: UNESCO, París (Francia).

ÁVILA, A., CAMILLONI, L., MAROTTA, F., VALLESPIR, D. y APA, C., 2013. Pruebas Unitarias en Java JUnit y TestNG. *Milveinticuatro*, pp. 46-48.

BERNERS-LEE, T., 2013. Linked Data, 2006. , pp. 1-7.

BEUST, C., 2015. TestNG Now available. [en línea]. [Consulta: 20 marzo 2016]. Disponible en: http://testng.org/doc/index.html.

BIBLIOTECAS UNIVERSIDAD DE SALAMANCA, 2016. *Los encabezamientos de materia*. 2016. S.l.: s.n.

BOERAS VELÁZQUEZ, M., CABRERA BARROSO, L., LLANO CASTRO, E. y GONZALEZ SÁNCHEZ, M.A., 2012. Aplicando el método de Boehm y Turner. , vol. 5, no. 6, pp. 1-12.

BOOTSTRAP TEAM, 2016. Bootstrap. [en línea]. [Consulta: 23 mayo 2016]. Disponible en: http://getbootstrap.com/.

BORST, W.N., 1997. *Construction of engineering ontologies for knowledge sharing and reuse*. The Netherlands: Universiteit Twente. ISBN 9036509882.

CALVANESE, D., GIACOMO, G. De, LEMBO, D., LENZERINI, M., POGGI, A., RODRIGUEZ-MURO, M., ROSATI, R., RUZZI, M. y SAVO, D.F., 2011. The MASTRO system for ontology-based data access. , vol. 2, pp. 43-53. DOI 10.3233/SW-2011-0029.

CALVANESE, D., LIUZZO, P., MOSCA, A., REMESAL, J., REZK, M. y RULL, G., 2016. Ontology-based data integration in EPNet : Production and distribution of food during the Roman Empire. *Engineering Applications of Artificial Intelligence* [en línea], pp. 1-18. ISSN 0952-1976. DOI 10.1016/j.engappai.2016.01.005. Disponible en: http://dx.doi.org/10.1016/j.engappai.2016.01.005.

CALVO POYATO, C., HIDALGO LÓPEZ, A. y BLANCO MARTÍNEZ, R., 2004. *Directrices para Registros de Autoridad y Referencias*. S.l.: s.n. ISBN 8436938380.

CAMPOS HERRERA, A., 2007. Las reglas angloamericanas de catalogación y la norma ISO. *Acimed* [en línea], vol. 16, no. 2. Disponible en: http://scielo.sld.cu/scielo.php?pid=S1024-94352007000800010&script=sci_arttext.

CANTONE, D., 2006. *Implementación y debugging*. S.l.: USERSHOP.

DEBRAUWER, L., 2013. *Patrones de diseño en Java Los 23 modelos de diseño*. S.l.: s.n. ISBN 4459816954.

DEPARTAMENTO DE PROCESO TÉCNICO DE LA BIBLIOTECA NACIONAL DE ESPAÑA, 2016. Manual de Autoridades. [en línea]. [Consulta: 28 enero 2016]. Disponible en: http://www.bne.es/es/Micrositios/Publicaciones/AUTORIDADES/005_Registros/006_Encabezamientos/.

DÍAZ RODRÍGUEZ, Y., 2012. Control de autoridades de nombres personales de autores cubanos en ciencias de la salud. [en línea]. [Consulta: 28 enero 2016]. Disponible en: http://www.acimed.sld.cu/index.php/acimed/article/view/221/213.

FERNÁNDEZ ESCRIBANO, G., 2002. Introducción a Extreme Programming. . S.l.:

FIGUEROA, R.G., SOLÍS, C.J. y CABRERA, A.A., 2011. METODOLOGÍAS TRADICIONALES VS . METODOLOGÍAS ÁGILES. , pp. 1-9.

FUENTES, L. y TROYA, M., 2009. Lección 1 Desarrollo de Software Basado en Componentes. , pp. 1-22.

GARCÍA DE JALÓN, J., IGNACIO RODRÍGUEZ, J., MINGO, I., IMAZ, A., BRAZÁLEZ, A., LARZABAL, A., CALLEJA, J. y GARCÍA, J., 2000. *Aprenda Java como si estuviea en primero*. San Sebastian: UNIVERSIDAD DE NAVARRA.

GARRIDO ARILLA, M.R., 1996. *Teoría e Historia de la catalogación de documentos*. S.l.: s.n.

GAUCHAT, J.D., 2012. *El gran libro de HTML5, CSS3 y Javascript*. 1. Barcelona, España: marcombo. ISBN 978-84-267-1782-5.

GODOY, D.A., BELLONI, E., KOTYNSKI, H., DOS SANTOS, H. y SOSA, E., 2014. Simulando Proyectos de Desarrollo de Software Administrados con Scrum. , pp. 485-489.

GRECHANIK, M., 2012. Automatically Finding Performance Problems with Feedback-Directed Learning Software Testing. , pp. 156-166.

GRUBER, T.R., 1993. A Translation Approach to Portable Ontology Specifications by A Translation Approach to Portable Ontology Specifications. , vol. 5, no. April, pp. 199-220.

GUERRERO, C., 2014. ESTUDIO COMPARATIVO DE MARCOS DE TRABAJO PARA EL DESARROLLO SOFTWARE ORIENTADO A ASPECTOS. [en línea]. Disponible en: http://www.scielo.cl/scielo.php?pid=S0718-07642014000200008&script=sci_arttext&tlng=pt.

JAIN, N., MANGAL, P. y MEHTA, D., 2014. AngularJS : A Modern MVC Framework in JavaScript. , vol. 5, no. 12.

JETBRAINS, 2016a. Enjoy productive java. [en línea]. [Consulta: 8 mayo 2016]. Disponible en: https://www.jetbrains.com/idea/.

JETBRAINS, 2016b. Making Development an Enjoyable Experience. [en línea]. [Consulta: 8 mayo 2016].

Disponible en: https://www.jetbrains.com/idea/features/.

JOINT UNIVERSITY LIBRARIANS ADVISORY COMMITTEE, 2016. Hong Kong Chinese Authority Name Project (HKCAN). [en línea]. [Consulta: 8 mayo 2016]. Disponible en: http://www.julac.org/?page_id=2137.

KASIAK, T. y GODOY, D.A., 2012. Simulación de Proyectos de Software desarrollados con XP : Subsistema de Desarrollo de Tareas. , pp. 572-576.

KHARLAMOV, E., JIMENEZ-RUIZ, E., PINKEL, C., REZK, M., SKJÆVELAND, M.G., SOYLU, A., XIAO, G., ZHELEZNYAKOV, D., GIESE, M., HORROCKS, I. y WAALER, A., 2015. OPTIQUE : Ontology-Based Data Access Platform. , pp. 12-15.

KUSHIYAMA, G., 1931. CHARLES AMMI CUTTER. *Historical American Biography*, pp. 1-9.

LARMAN, C., 2002. *UML y Patrones*. 2. S.l.: Prentice Hall.

LEIVA MEDEROS, A.A., SENSO, J.A., DOMÍNGUEZ-VELASCO, S. y HÍPOLA, P., 2013. Authoris : a tool for authority control in the Semantic Web. *Library Hi Tech*, vol. 31, pp. 536-553.

LETELIER, P. y PENADÉS, M.C., 2006. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). [en línea], Disponible en: http://www.cyta.com.ar/ta0502/v5n2a1.htm.

LIBRARY OF CONGRESS, 2016a. Library of Congress Names. [en línea]. [Consulta: 20 febrero 2016]. Disponible en: http://id.loc.gov/authorities/names.html.

LIBRARY OF CONGRESS, 2016b. NACO - Name Authority Cooperative Program. [en línea]. [Consulta: 8 enero 2016]. Disponible en: http://www.loc.gov/aba/pcc/naco/.

LOZANO, C., 2016. SGBD CARACTERISTICAS VENTAJAS DESVENTAJAS REQUERIMIENTOS. [en línea]. [Consulta: 15 mayo 2016]. Disponible en: https://www.academia.edu/8199329/SGBD_CARACTERISTICAS_VENTAJAS_DESVENTAJAS_REQUERIMIENTOS.

MEIER, J.D., FARRE, C., BANSODE, P., BARBER, S., REA, D. y MICROSOFT CORPORATION, 2007. Types of Performance Testing. [en línea], Disponible en: https://msdn.microsoft.com/en-us/library/bb924357.aspx.

NODE.JS FOUNDATION, 2016. Node Js. [en línea]. Disponible en: https://nodejs.org/en/.

ORACLE, 2016. What is Java and why do I need it? [en línea]. [Consulta: 13 enero 2016]. Disponible en: https://java.com/en/download/faq/whatis_java.xml.

PASCUAL, C.H., 1999. EL CONTROL DE AUTORIDADES. , pp. 121-136.

PESET, F., FERRER-SAPENA, A. y SUBIRATS-COLL, I., 2011. Open data y Linked open data : su impacto en el área de bibliotecas y documentación. , no. 1990. DOI 10.3145/epi.2011.mar.06.

PIVOTAL SOFTWARE, 2016a. Let's build a better Enterprise. [en línea]. [Consulta: 5 mayo 2016]. Disponible en: https://spring.io/.

PIVOTAL SOFTWARE, 2016b. Spring Boot. [en línea]. [Consulta: 5 mayo 2016]. Disponible en: http://projects.spring.io/spring-boot/.

PIVOTAL SOFTWARE, 2016c. Spring Data. [en línea]. [Consulta: 5 mayo 2016]. Disponible en:

http://projects.spring.io/spring-data/.

PIVOTAL SOFTWARE, 2016d. Web MVC framework. [en línea]. [Consulta: 1 mayo 2016]. Disponible en: http://docs.spring.io/autorepo/docs/spring/3.2.x/spring-framework-reference/html/mvc.html.

POSTGRESQL, 2013. Sobre PostgreSQL. [en línea]. [Consulta: 1 mayo 2016]. Disponible en: http://www.postgresql.org.es/sobre_postgresql.

PRESSMAN, R.S. y MAXIM, B.R., 2015. *Software Engineering A practitioner`s approach 8th edition*. Eighth edi. New York: McGraw-Hill Education. ISBN 978-0-07-802212-8.

QUINTANA, G., MARQUÉS, M., ALIAGA, J.. y ARAMBURU, M.., 2008. Aprende SQL. *Publicaciones de la Universidad Jaume*, vol. 8.

REAL ACADEMIA ESPAÑOLA, 2016a. Apoyo. *Diccionario de la lengua española Edición del Tricentenario* [en línea]. [Consulta: 28 junio 2016]. Disponible en: http://dle.rae.es/?id=3HwlXFW|3HxjGqk.

REAL ACADEMIA ESPAÑOLA, 2016b. Sistema. *Diccionario de la lengua española Edición del Tricentenario* [en línea]. [Consulta: 28 junio 2016]. Disponible en: http://dle.rae.es/?id=Y2AFX5s.

ROMERO FERNÁNDEZ, Y. y GONZÁLEZ DÍAZ, Y., 2012. Patrón Modelo-Vista-Controlador. , vol. 11, no. 1, pp. 47-57.

RUANO ALVAREZ, W.A. y CALZADILLA REYES, D., 2015. *AUCTORITAS Sistema de apoyo para el control de autoridades*. S.l.: Universidad de las Ciencias Informáticas.

SMARTBEAR, 2016a. API Load Testing for REST and SOAP LoadUI NG Pr. [en línea]. [Consulta: 2 mayo 2016]. Disponible en: https://smartbear.com/product/ready-api/loadui/overview.

SMARTBEAR, 2016b. SoapUI. The Swiss-Army Knife of Testing. [en línea]. [Consulta: 2 mayo 2016]. Disponible en: https://www.soapui.org/about-soapui/what-is-soapui.html.

SOMMERVILLE, I., 2011. *Software Engineering Ninth Edition*. S.l.: Addison-Wesley. ISBN 9780137035151.

STANFORD CENTER FOR BIOMEDICAL INFORMATICS RESEARCH, 2016. Protégé. [en línea]. [Consulta: 5 mayo 2016]. Disponible en: http://protege.stanford.edu/products.php.

TABARES MARTÍN, L., LEIVA MEDEROS, A.A. y FERNÁNDEZ PEÑA, F.O., 2014. Diseño de un sistema para la generación de entradas de autoridad desde la perspectiva de los datos enlazados. ,

THE APACHE SOFTWARE FOUNDATION, 2016. Apache Tomcat®. [en línea]. [Consulta: 5 febrero 2016]. Disponible en: http://tomcat.apache.org/.

TORRES TORRES, V.A., NUÑEZ TORRES, E., MOLINA HERNÁNDEZ, Y., CABALLERO FERIA, D. y PEÑA GONZÁLEZ, Y., 2015. Inteligencia de negocios para la empresa de servicios de la Unión del Níquel. , pp. 39-55.

VALDEZ HUARACA, A.G., MENDOZA VALDEZ, J.L., TORRES ALARCÓN, S.J., PACHAS LAURA, C.O., MATÍAS SEBASTIAN, J. y ALFARO MEDINA, J., 2013. *Pruebas de Sistemas y Pruebas de Aceptación*. 2013. S.l.: s.n.

VALLESPIR, D., 2002. CRC y un Taller. , pp. 1-10.

VIRTUOSO OPEN SOURCE, 2016. Virtuoso Open Source. [en línea]. [Consulta: 14 marzo 2016]. Disponible en: http://virtuoso.openlinksw.com/.

W3C, 2008. El W3C expone los datos en la Web con SPARQL Tecnología potente para consultas distribuidas y diversidad de datos. [en línea]. [Consulta: 25 marzo 2016]. Disponible en: http://www.w3c.es/Prensa/2008/nota080115_sparql.

W3C, 2012. OWL. [en línea]. [Consulta: 24 marzo 2016]. Disponible en: https://www.w3.org/2001/sw/wiki/OWL.

W3C, 2013. AGROVOC. [en línea]. [Consulta: 16 marzo 2016]. Disponible en: https://www.w3.org/community/ontolex/wiki/AGROVOC.

W3C, 2014. Skosmos. [en línea]. [Consulta: 14 marzo 2016]. Disponible en: https://www.w3.org/2001/sw/wiki/Skosmos.

W3C, 2016a. Apache_Jena @ www.w3.org. [en línea]. [Consulta: 24 marzo 2016]. Disponible en: https://www.w3.org/2001/sw/wiki/Apache_Jena.

W3C, 2016b. Guía Breve de CSS. [en línea]. [Consulta: 27 marzo 2016]. Disponible en: http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo.

W3C, 2016c. Guía Breve de Servicios Web. [en línea]. [Consulta: 26 marzo 2016]. Disponible en: http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb.

W3C, 2016d. Guía Breve de Web Semántica. [en línea]. [Consulta: 25 marzo 2016]. Disponible en: http://www.w3c.es/Divulgacion/GuiasBreves/WebSemantica.

W3C, 2016e. HTML. [en línea]. [Consulta: 20 enero 2016]. Disponible en: https://www.w3.org/html/.

WARNER, S., 2015. Linked Data for Libraries: Experiments between Cornell, Harvard and Stanford. *Day 2* [en línea]. [Consulta: 22 enero 2016]. Disponible en: http://swib.org/swib15/programme.html.

WIKI DURASPACE, 2016. VIVO. [en línea]. [Consulta: 28 marzo 2016]. Disponible en: https://wiki.duraspace.org/display/VIVO/VIVO.

WYNAR, B.S., TAYLOR, A.G. y OSBORN, J., 1985. *Introduction to cataloging and classification*. S.l.: Libraries Unlimited Englewood, CO.

YADAGIRI, N. y RAMESH, P., 2013. Semantic Web and the Libraries : An Overview. *International Journal of Library Science*, vol. 7, pp. 80-94.

YU, L., 2011. *A Developer's Guide to the Semantic Web*. S.l.: Springer Science & Business Media.

ZACCHIROLI, S., 2011. Génie Logiciel Avancé Cours 6 — Extreme Programming slides in English. , no. Paris 7.