UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Título: MetaLib Service (Herramienta de Software para la Metabúsqueda)
Autores: Forte Belén Magalí, Giraldo Pirrotta Jose María
Director: De Giusti, Marisa
Codirector: ---
Asesor profesional: ---
Carrera: Licenciatura en Sistemas

Resumen

Es notable el crecimiento de recursos bibliográficos producido en Internet durante la última década. La gran dispersión generada entre éstos, y la poca relevancia al momento de organizar y estructurar los mismos, dificultó de manera significante la labor de aquellas personas interesadas en localizar dichos recursos.
Si bien son evidentes los aportes brindados por las múltiples bibliotecas digitales y los repositorios institucionales disponibles en la web, se creyó conveniente generar una abstracción del distanciamiento entre cada instancia, permitiendo englobar múltiples catálogos a partir de una única búsqueda. Por este motivo, resultó necesario crear la herramienta MetaLib Service con el objetivo de agilizar los servicios de búsqueda bibliográfica y facilitar la labor de referencistas y bibliotecarios encargados de atender la creciente demanda de investigadores, docentes y alumnos de su institución.

Palabras Claves

Metabúsqueda. Bibliotecas digitales. Repositorios institucionales. Catálogos. Recursos bibliográficos. Metadatos. Referencistas. Bibliotecarios. Acervos bibliográficos. Recolección de recursos.

Trabajos Realizados

Se diseñó e implementó una herramienta de software denominada MetaLib Service, con el objetivo de poder brindar un servicio de metabúsqueda sobre un conjunto de catálogos en línea. Dicha herramienta podrá ser utilizada por cualquier persona, institución o entidad que encuentre con la necesidad de localizar documentos específicos de manera rápida y eficaz.
MetaLib Service cuenta con una API para aquellas aplicaciones de terceros que estén interesadas en consumir el servicio brindado.

Conclusiones

Los metabuscadores hacen un gran aporte a la hora de localizar recursos bibliográficos; sin embargo, resulta evidente la necesidad de contar con herramientas que permitan agilizar los servicios de búsqueda bibliográfica y que faciliten la labor de referencistas y bibliotecarios encargados de satisfacer los pedidos.
Por este motivo se desarrolló MetaLib Service, diseñado para asistir a alumnos, docentes y referencistas, en el proceso de recolección de recursos bibliográficos altamente específicos.

Trabajos Futuros

En una primera etapa, se buscará optimizar la interfaz web de la herramienta para brindar una mejor visualización de los listados de pedidos obtenidos a través de la misma.
Se ideara un mecanismo a partir del cual será posible procesar y analizar cada una de las operaciones persistidas en el sistema, con el fin de lograr la generación de múltiples estadísticas.
Se prevé agregar funciones que permitan la paginación de los resultados recolectados por la metabúsqueda.

Fecha de la presentación: Diciembre de 2012

Agradecimientos

Belén Forte

A mis padres Silvia y Raúl, por darme la vida y ayudarme desde un principio para poder llegar a esta instancia de mi vida.

A mis hermanos Daniela y Lisandro, por el apoyo incondicional en todas las decisiones camino a mis objetivos.

A mi novio Santiago, quién con su amor me dió las fuerzas necesarias para que pueda llegar a esta instancia.

A mi amigo y compañero de Tesina José, quién le puso empeño, paciencia y garra para poder lograr este trabajo.

Gracias.

José Giraldo

A mis padres Blanca y Arturo, por el incansable esfuerzo y dedicación para que pueda llegar a este momento en mi vida.

A mis hermanas Valeria y Cocó, por el apoyo y el cariño que me dieron durante todo este tiempo para que pueda lograr mis objetivos.

A mi amiga y compañera de Tesina Belén, quién confió en mí para emprender juntos este camino.

Gracias.

Belén Forte y José Giraldo

A nuestra directora de Tesina Marisa, por estar siempre presente en los momentos difíciles y por confiar en nosotros.

A Gonzalo Villarreal, por guiarnos durante todo el camino y brindarnos todo el apoyo necesario para lograr este trabajo.

Gracias.

Índice

Capítulo 1

Introducción

1.1. Motivación

El incremento exponencial de los recursos bibliográficos disponibles en Internet en el transcurso de los últimos años, potenciado por los cambios sociales y culturales que las diversas sociedades experimentaron durante la segunda mitad del siglo XX y especialmente en la última década, provocó el nacimiento de la *biblioteca digital* como una entidad virtual. Con la biblioteca digital se han generado cambios en la forma de administrar los recursos bibliográficos, cambiando de este modo las características básicas de la biblioteca tradicional. Uno de los cambios más importantes fue el reemplazo de los objetos físicos por su contraparte lógica, lo cual ha implicado la generación de índices lógicos de los recursos y permitió nuevos mecanismos de organización de tales recursos dentro de los catálogos de las distintas bibliotecas.

La estructuración y codificación de los objetos digitales se realiza básicamente por medio de estructuras de metadatos, que sirven para describir y organizar cada objeto digital y permitir su identificación unívoca y localización dentro del acervo de la biblioteca digital. Sin embargo, el crecimiento de los recursos digitales disponibles sumado a la necesidad de localización de documentos específicos por parte de personas con intereses particulares o puntuales en determinadas áreas, hizo necesaria la incorporación de herramientas complementarias para agilizar y automatizar las tareas relacionadas con la búsquedas y localización de estos recursos. Es así como surgen conceptos como el de metabúsqueda, mediante la cual un usuario puede consultar múltiples catálogos desde un único punto central y de manera casi transparente. Los sistemas de metabúsqueda permiten unificar acervos bibliográficos de múltiples bibliotecas digitales, y ofrecer a los usuarios un espectro de documentos más amplio. Este tipo de herramientas ayuda también a ampliar la base de usuarios de la biblioteca digital, construir puentes para aumentar la cooperación entre bibliotecas físicamente distantes y ofrecer servicios de provisión bibliográfica mucho más eficientes y eficaces.

Los metabuscadores hacen un gran aporte a la hora de localizar recursos bibliográficos; sin embargo, resulta evidente la necesidad de contar con herramientas que permitan agilizar los servicios de búsqueda bibliográfica y que faciliten la labor de referencistas y bibliotecarios encargados de atender la creciente demanda de investigadores, docentes y alumnos de su institución. Dicha labor consiste en localizar un recurso solicitado a través de un pedido, mediante una búsqueda en los diferentes repositorios y bibliotecas digitales de acceso abierto.

Una alternativa a explorar es la integración de motores de metabúsqueda con los sistemas de gestión de los servicios de referencia bibliográfica que ofrecen las bibliotecas. Dicha integración puede realizarse de manera transparente y automática, sin la necesidad de la intervención manual de los referencistas. Así por ejemplo, cada vez que se registra un nuevo pedido de bibliografía en el sistema de gestión de la biblioteca, el bibliotecario o referencista dispondrá "automáticamente" de una lista de resultados posibles basados en una recolección de recursos realizada por el herramienta de metabúsqueda, con el fin de optimizar el trabajo del referencista. Esta integración transparente entre el metabuscador y el software de gestión debe estar coordinada por un usuario administrador, cuyo trabajo será el de especificar ciertos parámetros de búsqueda, para refinar y efectivizar la misma. La actividad de recolección de recursos puede realizarse de forma simultánea sobre múltiples catálogos y repositorios institucionales, lo que ampliará aún más la oferta de recursos bibliográficos.

Docentes, estudiantes e investigadores son los principales consumidores de recursos bibliográficos altamente específicos, y por lo tanto de servicios de búsqueda de dichos recursos. Debido al amplio abanico de fuentes de recursos disponibles, se produce una dispersión de solicitudes realizadas por parte de dichos usuarios a sus bibliotecas, con lo que se pierde la posibilidad de caracterizar las búsquedas realizadas de acuerdo a perfiles pre-establecidos, lo que se traduce en una pérdida de tiempo. Adicionalmente, todas estos actores podrían beneficiarse considerablemente con servicios que les *propongan* recursos bibliográficos actualizados de manera automática y transparente a partir de la disponibilidad en la red de catálogos o en los repositorios de sus instituciones. Estas propuestas pueden ser realizadas directamente desde los sitios web de las instituciones, blogs de las cátedras o sistemas de e-learning. Lamentablemente, estas herramientas no están diseñadas para ofrecer con servicios bibliográficos, y mucho menos con herramienta de metabúsqueda, y resulta necesario realizarles modificaciones o extensiones para aprovechar las capacidades de los nuevos sistemas de las bibliotecas digitales.

1.2. Objetivos

En este trabajo se presenta un producto de software denominado MetaLib Service (Herramienta de Software para Metabúsqueda), el cual estará diseñado para asistir a alumnos, docentes y referencistas, en el proceso de recolección de recursos bibliográficos altamente específicos.

El objetivo será poder brindar, a partir de MetaLib Service, un servicio de metabúsqueda sobre un conjunto de catálogos en línea. Dicha herramienta podrá ser utilizada por cualquier persona, institución o entidad que necesite realizar búsquedas particulares o puntuales en determinadas áreas y se encuentre con la necesidad de localizar documentos específicos de manera rápida y eficaz. La herramienta contará con una API para aquellas aplicaciones de terceros que estén interesadas en consumir el servicio brindado, encargada de abstraer el funcionamiento principal, permitiendo así una integración clara y transparente.

1.3. Organización de la Tesina

Este informe se encuentra dividido en diversos capítulos que abarcan temas relacionados entre sí.

Resumen del Capítulo 2

En este capítulo se plantea un marco teórico de los distintos conceptos en los que se hará mayor hincapié a lo largo de los diferentes capítulos, con el objetivo de facilitar la comprensión de la posterior lectura.

Resumen del Capítulo 3

En este capítulo se describe MetaLib Service, una herramienta desarrollada para la realización de metabúsquedas. La misma puede ser utilizada por cualquier persona, institución o entidad que que necesite realizar búsquedas particulares o puntuales en determinadas áreas y se encuentre con la necesidad de localizar documentos específicos de manera rápida y eficaz.

Resumen del Capítulo 4

En este capítulo se presentan en detalle los entornos de aplicación y casos de prueba que fueron considerados, junto con los resultados obtenidos.

Resumen del Capítulo 5

Se presentan las conclusiones finales obtenidas y los trabajos e investigaciones futuras que se esperan realizar con el fin de optimizar la herramienta.

Anexo 1 - Integración y uso de la API

En este apartado, se busca brindar soporte a aquellos desarrolladores de aplicaciones de terceros que estén interesados en integrar la API, con el fin de interactuar con el servicio brindado por MetaLib Service.

Capítulo 2

Conceptos generales relacionados a repositorios digitales

2.1. Introducción

Hace unos años, la información ha sufrido la denominada “explosión digital” originada por la rápida evolución de la tecnología y el crecimiento de Internet con sus múltiples posibilidades de acceso y socialización de la información. Este nuevo hito informacional ha generado algunos problemas y nuevas oportunidades a los protagonistas de la organización documental, quienes deben abocarse a crear y adaptar herramientas para un tratamiento más eficiente de la información en el contexto digital.

Las nuevas representaciones de conocimiento, como son las publicaciones electrónicas tanto los textos, imágenes, entre otros, han disminuido las expectativas de relevancia en la recuperación de la información. Teniendo en cuenta que una búsqueda en Internet por medio de un buscador común arroja una cantidad excesiva de resultados, de dudosa calidad y, muchas veces, fuera de contexto con respecto a los parametros de busqueda, es necesario buscar soluciones que sean aplicables, reusables y, que además, permitan administrar y recuperar eficientemente la información.

En la actualidad existen múltiples bibliotecas digitales y repositorios institucionales, en los cuales la visibilidad de sus recursos preservados se incrementa gradualmente porque, a su vez, crece su necesidad de interoperar con otros sistemas de información digital. El término *preservación digital* se basa en la búsqueda de soluciones para conservar aquellos documentos digitales almacenados, sea cual sea su formato, el software, hardware o sistema que se utilizó para su creación, manteniendo así la información pese a los rápidos cambios tecnológicos.

Según la Digital Library Project, la biblioteca digital se puede definir del siguiente modo:

"El concepto de biblioteca digital no es únicamente el equivalente de repertorios digitalizados con métodos de gestión de la información. Es más bien, un entorno donde se reúnen colecciones, servicios, y personal que favorece el ciclo completo de la creación, difusión, uso y preservación de los datos, para la información y el conocimiento" (Santa Fe Workshop on Distributed Knowledge Work Environments).

Por otro lado Clifford Lynch, dice que un repositorio institucional universitario es:
"Un conjunto de servicios que ofrece la universidad a los miembros de su comunidad para la dirección y distribución de materiales digitales creados por la institución y los miembros de esa comunidad. Es esencial un compromiso organizativo para la administración de estos materiales digitales., incluyendo la preservación a largo plazo, cuando sea necesario, así como la organización y acceso o su distribución".
Las características más importantes de un repositorio institucional son:

- Pertenecen a una institución académica.
- Son acumulativos y perpetuos.
- Son abiertos e interactivos.

A partir de éstas dos definiciones, podemos concluir que las diferencias más notables entre estos dos conceptos se basan en que los repositorios son un lugar donde se almacena material generado por los propios académicos de la universidad, es decir , es un sistema de información que reúne, preserva, divulga y da acceso a la producción intelectual y académica de las comunidades universitarias, a diferencia de la biblioteca donde en general se coloca material que puede ser de utilidad para los usuarios y que provienen de diversas fuentes.

Tanto las bibliotecas digitales como los repositorios están compuestos por "objetos digitales", también llamados recursos. Dichos recursos se encuentran organizados a través de un proceso de catalogación. La catalogación es un subconjunto de un campo mucho mayor que, en ocasiones, es llamado control bibliográfico. El control bibliográfico ha sido definido como *"... estrechamente ligada, en un ciclo, a la creación, almacenamiento, manipulación y recuperación de datos bibliográficos"* (R. Smiraglia).
La catalogación puede ser definida como el medio a través del cual los catálogos son preparados. El catálogo es un conjunto organizado de registros que representan las obras que forman parte de una colección en particular. Los catálogos cumplen varias funciones, las cuales

fueron establecidas en 1904 por Charles A. Cutter en *Rules for a dictionary catalog*. Según Cutter los objetivos del catálogo son:

- Posibilitar que una persona encuentre un libro cuando algunos de los siguientes elementos es conocido: el autor, el título o el tema.
- Mostrar lo que la biblioteca tiene escrito sobre un autor dado, sobre un tema dado o en tipo específico de literatura.
- Asistir en la elección de un libro bibliográficamente (como lo relativo a su edición) o literaria/temáticamente (como lo relativo a su carácter).

En el año de 1961 la International Federation of Library Associations (IFLA) en su conferencia de París estableció los propósitos de los catálogos de autor/título los cuales establecen que el catálogo debe ser un instrumento eficiente para poder determinar si la biblioteca tiene un libro en particular determinado por su autor o por su título; si el autor no es nombrado en el libro, sólo por su título; si el autor y el título son insuficientes para identificar el libro, por un subtítulo apropiado para el título. Y, cuáles libros existen escritos por un autor determinado y cuáles ediciones de un libro en particular tiene la biblioteca.

Existen varios tipos de catálogo, los cuales son:

- Clasificados. Ordenados de acuerdo al número de clasificación de las obras listadas.
- Alfabéticos. Ordenados alfabéticamente de acuerdo al elemento de entrada de las obras listadas. A su vez, los catálogos alfabéticos pueden ser de tipo diccionario o divididos.
- En línea. La información se despliega de acuerdo al diseño que se tenga de los formatos de salida, la cual puede variar enormemente de uno a otro.

El proceso de catalogación se divide en 4 grandes ramas:

- Catalogación descriptiva.
- Análisis temático.
- Control de autoridad.
- Catalogación cooperativa.

Catalogación descriptiva

La catalogación descriptiva es esa fase del proceso de catalogación que tiene que ver con la identificación y descripción de una obra, el registrar la información en la forma de un registro

catalográfico, y la selección y formación de los puntos de acceso, con excepción de los puntos de acceso temático. La catalogación descriptiva describe los aspectos físicos de la obra e identifica la responsabilidad para el contenido intelectual, sin hacer referencia a su clasificación temática o a la asignación de los encabezamientos temáticos, ambos elementos propios de la catalogación temática.

Análisis temático

El análisis temático consiste en la determinación de los temas que aborda una obra o en palabras de un lenguaje controlado con que se define el contenido intelectual de la obra. Incluye tópicos, personas, organizaciones, estos se conocen como encabezamientos de materia.

Control de autoridad

El control de autoridad es el proceso de mantener consistencia en la forma usada para representar un punto de acceso y el proceso de mostrar las relaciones entre nombres, obras y temas. Esto se logra a través del uso de reglas.

Catalogación cooperativa

Todos los procesos anteriormente descritos tienen la característica de ser originales; este proceso, en cambio, posee la característica principal de requerir la participación de un grupo determinado de instituciones. Consiste en que una de las instituciones realiza la catalogación original de una obra, poniéndola a la disposición del grupo. Generalmente suelen utilizarse formatos internacionales y estandarizados para el intercambio de información, como podría ser MARC, Dublin Core, entre otros.

Actualmente existen varios estándares creados y en desarrollo para dar solución a los problemas relacionados con la interoperabilidad entre los distintos repositorios institucionales y las bibliotecas digitales con otros sistemas de información digital. A continuación se brinda una definición concreta sobre éste término:

“La interoperabilidad es la capacidad de un sistema o de un producto de trabajar con otros sistemas o productos sin un esfuerzo especial por parte del cliente. Desde este punto de vista computacional, la interoperabilidad permite generar un enlace entre sistemas de trabajo para

las diferentes tecnologías de información promoviendo una sana convivencia y operatividad... " (Martínez Equihua, Saúl, 2007).

Los estándares arriba mencionados deben ser aplicables y reusables en forma amplia en espacios que albergan información digital, siendo alguno de ellos:

- Transporte y Comunicación: Z39.50, OpenSearch, SRU/SRW, HTTP, OAI-PMH, cURL.
- Metadatos: XML, JSON, Dublin Core, Marc, entre otros.

A lo largo de éste capítulo se detallan, entre otras cosas, cada uno de los estándares mencionados anteriormente.

2.2. Transporte y comunicación

HTTP

El Protocolo de Transferencia de HiperTexto (Hypertext Transfer Protocol) es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP. La especificación completa del protocolo HTTP/1.0 está recogida en el RFC 1945[1]. Fue propuesto por Tim Berners-Lee, atendiendo a las necesidades de un sistema global de distribución de información como el World Wide Web.

Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión TCP/IP (*Transmission Control Protocol/Internet Protocol*), y el funcionamiento se basa en: un proceso servidor escucha en un puerto de comunicaciones TCP (por defecto, el 80), y espera las solicitudes de conexión de los clientes Web. Una vez que se establece la conexión, el protocolo TCP se encarga de mantener la comunicación y garantizar un intercambio de datos libre de errores.

HTTP se basa en sencillas operaciones de solicitud/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web

[1] http://www.w3.org/Protocols/rfc1945/rfc1945

(documento HTML, fichero multimedia o aplicación CGI) es conocido por su URL (*Uniform Resource Location*).

El estándar HTTP/1.0 recoge únicamente tres comandos, que representan las operaciones de recepción y envío de información y chequeo de estado:

- *GET* : se utiliza para recoger cualquier tipo de información del servidor.
- *HEAD*: solicita información sobre un objeto (fichero), por ejemplo tamaño, tipo, fecha de modificación.
- *POST*: sirve para enviar información al servidor, por ejemplo los datos contenidos en un formulario.

cURL

cURL es una librería de funciones escrita e ideada por Daniel Stenberg, la cual permite realizar una conexión y acceder a una multitud de servidores distintos como telnet, http, ftp o ldap. cURL es open source/software libre distribuido bajo la Licencia MIT.
El principal propósito y uso para cURL es automatizar transferencias de archivos o secuencias de operaciones no supervisadas. Es por ejemplo una buena herramienta para simular las acciones de un usuario en un navegador web. Libcurl es la biblioteca/API correspondiente que los usuarios pueden incorporar en sus programas; cURL actúa como un envoltorio (wrapper) aislado para la biblioteca libcurl.

En el desarrollo de la herramienta, particularmente para el desarrollo de la librería[2] para integrarla con aplicaciones de terceros, se ha utilizado la librería Libcurl. Las funciones más importantes de Libcurl son:

- *curl_init()*: recibe por parámetro la url a la que se desea conectarse, por ejemplo:

  $conn = curl_init('url');

  Inicia una nueva sesión y devuelve el manipulador curl para el uso de las funciones. En caso de producirse algún tipo de error, retorna *false*.

  Inicia una nueva sesión y devuelve el manipulador curl para el uso de las funciones. En caso de producirse algún tipo de error, retorna *false*.

[2] Ver más en *Anexo 1: Introducción y uso de la API*

- *curl_copy_handle()*: Copia el recurso cURL junto con todas sus preferencias. Recibe por parámetro el recurso cURL devuelto por *curl_init()*.
- *curl_error()*: Devuelve un mensaje claro del error en formato de texto de la última operación cURL o una cadena vacía si no ocurrió ningún error.
- *curl_close()*: Esta función cierra una sesión CURL y libera todos sus recursos.

OpenSearch

OpenSearch es un conjunto de tecnologías que tienen como objetivo integrar de forma más sencilla las consultas a buscadores por parte de cualquier aplicación. Los promotores de este estándar fueron A9.com y Amazon.com, y hoy en día se está convirtiendo en un estándar de facto en principales sistemas de búsqeda. OpenSearch fue desarrollado sobre el estándar XML y está disponible de acuerdo con la los términos de las licencias Creative Commons.

OpenSearch está formado por:

- Archivos descriptivos OpenSearch / OpenSearch Description Documents: archivos en formato XML que identifican y describen un motor de búsqueda. Explican cómo obtener los resultados de un servidor.
- OpenSearch RSS (en la versión 1.0) o OpenSearch Response (en la 1.1): formato para proporcionar los resultados de búsquedas.
- Agregadores: sitios que pueden mostrar resultados realizados mediante OpenSearch.

A continuación se detallan las distintas etiquetas por las cuales está compuesto el descriptor. Se tomará como ejemplo el descriptor del repositorio de la Universidad Nacional de La Plata (SeDiCi) a fin de explicar detalladamente qué representa cada etiqueta del XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<OpenSearchDescription xmlns="http://a9.com/-/spec/opensearch/1.1/">
<ShortName>SeDiCI</ShortName>
<LongName>Servicio de Difusión de la Creación Intelectual</LongName>
<Description>Repositorio institucional de la Universidad Nacional de La Plata</Description>
<InputEncoding>UTF-8</InputEncoding>
<OutputEncoding>UTF-8</OutputEncoding>
<Query role="example" searchTerms="software"></Query>
<Tags>SeDiCI, Open Access, Digital Library, UNLP, Academic resources</Tags>
<Contact>administradores@sedici.unlp.edu.ar</Contact>
```

```
<Image height="16" type="image/vnd.microsoft.icon"
width="16">http://sedici.unlp.edu.ar/themes/Sedici/images/favicon.ico</Image>
<Url template="http://sedici.unlp.edu.ar/open-
search/discover?query={searchTerms}&start={startIndex?}&rpp={count?}&format=atom"
type="application/atom+xml; charset=UTF-8"></Url>
<Url template="http://sedici.unlp.edu.ar/open-
search/discover?query={searchTerms}&start={startIndex?}&rpp={count?}&format=rss"
type="application/rss+xml; charset=UTF-8"></Url>
</OpenSearchDescription>
```

Figura 2.1 *Ejemplo descriptor Open Search de SeDiCi*

- ShortName: indica el nombre corto que identifica al repositorio. La cantidad máxima de caracteres puede ser 16. El texto no debe contener texto HTML o de otro tipo, debe ser texto sin formato.
- Description: representa una pequeña descripción del servicio. Esta permitido hasta 1024 caracteres. El texto no debe contener texto HTML o de otro tipo, debe ser texto sin formato.
- InputEncoding: indica la codificación de los caracteres de entrada. Por defecto es UTF-8. Este elemento puede aparecer, ninguna, una o más veces.
- OutputEncoding: idem anterior pero hace alusión a los caracteres de salida. Por defecto es UTF-8. Este elemento puede aparecer, ninguna, una o más veces.
- Query: identifica consultas predefinidas. El estándar recomienda definir una consulta de ejemplo (utilizando el atributo role="example"), que en este caso es *software*. Los diferentes atributos que puede tener el elemento son:
    - role: contiene una cadena que identifica cómo el cliente de búsqueda debe interpretar la petición de búsqueda definida por este elemento de consulta. Este atributo es obligatorio y debe ser un valor definido en la especificación de roles (role values specification[3])
    - title: contiene una cadena que describe la petición de búsqueda. La cantidad máxima de caracteres puede ser 256. El texto no debe contener texto HTML o de otro tipo, debe ser texto sin formato.
    - totalResults:
    - searchTerms:
    - count:

[3] http://www.opensearch.org/Specifications/OpenSearch/1.1#Role_values

    - startIndex
    - startPage
    - language
    - inputEncoding
    - outputEncoding
- Tags: contienen las diferentes etiquetas que podrán definir qué es lo que se puede encontrar en el repositorio o para qué sirve. La cantidad máxima de caracteres puede ser 256. El texto no debe contener texto HTML o de otro tipo, debe ser texto sin formato. El elemento puede aparecer más de una vez.
- Contact: contiene una dirección de contacto del repositorio de búsqueda. El contenido se debe ajustar a los requisitos de la sección 3.4.1 de RFC 2822[4] "Addr-spec specification".
- URL: este atributo es el más importante, ya que con esta etiqueta se define cómo tiene que hacer la petición el navegador para obtener los resultados de búsqueda que desea el usuario. Los atributos de dicho campo pueden ser:
    - template: indica la forma en que se construirá la URL para la consulta. Dentro de este atributo se pueden introducir plantillas que se expanden de forma dinámica; la más habitual es `{searchTerms}`, la cual se expande a los términos de búsqueda introducidos por el usuario en la búsqueda.
    - type: indica el tipo de recurso que se describe. Debe ser un tipo MIME válido, ejemplo “text/html”, “application/rss+xml”, entre otros. Este campo es obligatorio.
    - rel: Ejemplo: “results”, “suggestions”, “self”, “collections”.[5]
    - indexOffset: indica el número de índice del primer resultado de la búsqueda. Por defecto es 1.
    - pageOffset: indica el número de página del primer conjunto de resultado de búsqueda. Por defecto es 1.

Es importante señalar que no es necesario definir todos estos atributos, de hecho únicamente son obligatorios Shortname, Description y URL.

Dspace

[4] http://tools.ietf.org/html/rfc2822

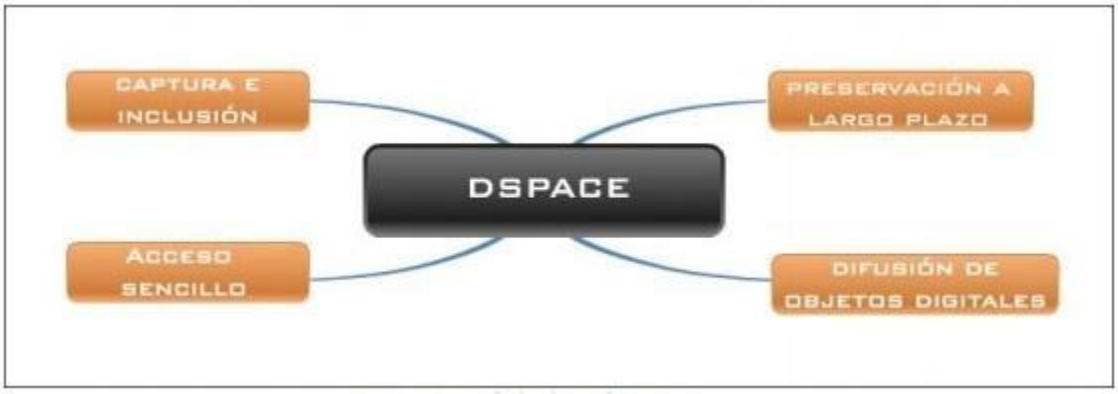
[5] http://www.opensearch.org/Specifications/OpenSearch/1.1#Url_rel_values

Un ejemplo de software compatible con OpenSearch es DSpace, utilizado por el repositorio de la UNLP (SeDiCi) y cuya funcionalidad ha sido integrada en este trabajo (en el siguiente capítulo se trata en detalle esta integración). DSpace es una plataforma de software de código abierto desarrollado por las bibliotecas del MIT (Massachusetts Institute of Technology) y Hewlett Packard Labs. Se encuentra desarrollado en Lenguaje Java bajo la licencia BSD de código abierto. Esto significa que cualquier organización puede utilizar, modificar, e incluso integrar el código en su aplicación comercial sin pagar derechos de licencia.

DSpace se trata de un sistema de repositorio digital que preserva y permite acceder fácilmente a todo tipo de contenido digital, incluyendo textos, imágenes, vídeos, entre otros. Tiene por objetivo organizar la producción intelectual de una institución, describirla, difundir y preservar sus objetos digitales en el tiempo. Además DSpace es personalizable para adaptarse a las necesidades de cualquier organización.

Las cuatro funcionalidades principales de DSpace son:

- Facilitar la captura y la inclusión de materiales, incluyendo sus metadatos.
- Facilitar el acceso a los mismos de una manera sencilla.
- Facilitar la preservación a largo plazo de dichos materiales.
- Facilitar la difusión de los objetos digitales.



[6]

Figura 2.2 *Funcionalidades de DSpace*

Estructura DSpace

La información que almacena DSpace está estructurada en 5 componentes básicos:

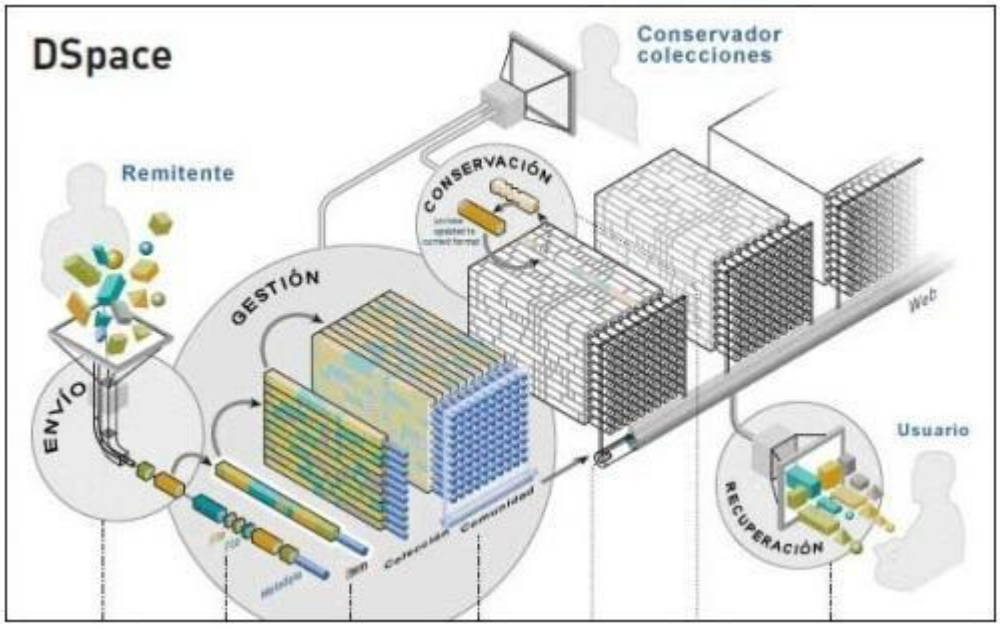
- Comunidades/Subcomunidades: son conjuntos de colecciones y subcomunidades.

[6] Imagen tomada de: Ibai Sistemas.(marzo, 2012) Informe de adaptación de DSpace a Europeana. Fase Danubio: Europeana Data Model (EDM)

- Colecciones: agrupaciones de ítems.
- Ítems: metadatos + archivos.
- Bundles: paquetes de archivos.
- Bitstreams: archivos.

Los ítems se subdividen en paquetes (Bundles) de Bitstreams. Un Bitstream puede ser un archivo html, una imagen, un documento word, etc. Un ejemplo de Bundle puede ser un grupo de archivos con código html y sus imágenes relacionadas que se unen para formar un documento html. Las comunidades y colecciones en DSpace se utilizan para hacer que el repositorio sea fácilmente navegable, con una estructura que, a menudo, representa la propia organización. Cada comunidad contiene metadatos descriptivos acerca de sí misma y de las colecciones que figuran en ella. Las comunidades pueden ser departamentos, centros de investigación, o alguna otra unidad administrativa dentro de la institución.

Generalmente el administrador de DSpace, trabaja en una comunidad para crear flujos de trabajo, aprobar contenidos, etiquetar metadatos, etc. Las colecciones pueden pertenecer a una comunidad o a varias comunidades (la colaboración entre las comunidades puede dar lugar a una colección compartida). Como ocurre con las comunidades, cada colección contiene metadatos descriptivos sobre sí misma y los temas contenidos en ella.



[7]

[7] Imagen tomada de: Ibai Sistemas.(marzo, 2012) Informe de adaptación de DSpace a Europeana. Fase Danubio: Europeana Data Model (EDM)

Figura 2.3 *Funcionamiento de DSpace*

Z39.50

El protocolo Z39.50 es un estándar internacional para la comunicación entre sistemas informáticos, el mismo posee ya muchos años de uso (comenzó a utilizarse en los 70), lo cual lo hace robusto y estable, y sigue siendo un eslabón clave en el desarrollo de sistemas en el ámbito de Bibliotecas Digitales. Z39.50 provee un mecanismo eficiente para recuperar información basado en la filosofía Cliente-Servidor, y es totalmente independiente de la plataforma (hardware y software) en la cual se ejecutan clientes y servidores

El estándar Z39.50 especifica formatos y procedimientos para gestionar el intercambio de mensajes entre un cliente y un servidor, permitiendo así que el usuario busque en bases de datos remotas, identifique los registros que se ajusten a determinados criterios, y recupere algunos o todos los registros identificados, así como otras informaciones asociadas en función de la base de datos.

Las aplicaciones que recolectan los recursos vía Z, tienen la ventaja de conseguir un acceso uniforme a un gran número de fuentes de información diversas y heterogéneas, incluso de manera simultánea, superando las diferencias entre los sistemas informáticos, los motores de búsqueda y las distintas bases de datos. Para alcanzar esta interoperabilidad entre distintos sistemas, Z39.50 facilita un lenguaje común para realizar las dos operaciones básicas que garantizan la recuperación de información: selección de información y obtención de la misma. Por ello, Z39.50 contempla la estandarización tanto de los mecanismos de codificación (cómo deben codificarse los datos para ser transferidos) como de la semántica del contenido (modelización de los datos con una semántica común para cada comunidad específica).

Desde el punto de vista de los mecanismos de codificación, es importante recordar que Z39.50 simplemente es un estándar de comunicación, es decir, sólo especifica qué debe transferirse, pero adicionalmente necesita un servicio de transporte fiable. Z39.50 especifica la estructura (Abstract Syntax Notation One, ASN.1) de los mensajes (Protocol Data Units, PDUs) que se intercambian entre cliente y servidor.

Cabe destacar que Z39.50 no sólo provee un método eficiente para recuperar información, sino que también facilita la interconexión de sistemas informáticos, sin importar el software y hardware que corran los sistemas involucrados.

El protocolo Z provee los siguientes servicios:

- 11 servicios nativos:

- Initialization: permite al cliente negociar una Z-asociación (funcionalidad soportada, juego de caracteres, idioma, etc.).
- Search: permite al cliente consultar las bases de datos de un servidor, crear el conjunto de resultados en el servidor y recibir información sobre dicho conjunto.
- Retrieval: incluye dos servicios distintos, Present, que permite al cliente solicitar uno o más registros de un conjunto de resultados, y Segmentation, que permite al servidor descomponer en varios segmentos la información solicitada en los casos necesarios.
- Result-set-delete: permite a un cliente solicitar que se borre un conjunto de resultados determinado, o todos ellos.
- Access Control: permite al servidor que evalúe al cliente, mediante palabras depaso, etc.
- Accounting/resource Control: incluye tres servicios distintos, Resource-control, que permite el servidor controlar e informar al cliente de los recursos consumidos o estimados, Trigger-resource-control, que permite al cliente solicitar que se inicie el control de recursos o cancelar la operación, y Resourcereport, que permite al cliente solicitar un informe de recursos tanto de una operación, como de una sesión completa.
- Sort: permite que el cliente solicite al servidor una ordenación del conjunto de resultados, o unir varios conjuntos y luego ordenar el resultado.
- Browse: permite recorrer una lista ordenada de términos (materias, títulos, etc.).
- Explain: ofrece detalles del servidor como bases de datos disponibles, índices, servicios disponibles, etc. con idea de que se puedan desarrollar clientes que se auto configuren en función de los servidores que encuentren.
- Extended Services: permite el acceso a servicios ajenos al protocolo que pueden perdurar una vez termine la Z-asociación como búsquedas periódicas, conservar conjuntos de resultados, etc.
- Termination: permite al cliente o al servidor interrumpir las operaciones activas e iniciar el cierre de la Z-asociación.

- 7 servicios extendidos:

  Permiten definir tareas que se desarrollan por el cliente Z fuera de la sesión Z39.50. El servidor mantiene una base de datos especial con estas tareas definidas por el cliente.

Dichas tareas suelen ser más complejas que las operaciones de búsqueda (Search) y recuperación (Retrieval).

- PersistentResultSet: permite guardar un conjunto de resultados para su uso posterior.
- PersistentQuery: permite guardar estrategias de búsquedas.
- PeriodicQuerySchedule: permite definir calendarios periódicos de búsqueda.
- ItemOrder: permite solicitar ejemplares.
- DatabaseUpdate: permite actualizar la base de datos.
- ExportSpecification: permite crear especificaciones de exportación.
- ExportInvocation: permite invocar una especificación de exportación.

Para garantizar el otro aspecto de la interoperabilidad, conjuntamente a esta codificación estándar de mensajes, Z39.50 se apoya en el concepto de conocimiento de una semántica compartida. Los distintos ámbitos proveedores o consumidores de contenidos de información han ido acordando estructuras y atributos comunes en cada uno de ellos, lo que permite un acceso uniforme a información heterogénea dentro de cada ámbito o comunidad.
El modelo de arquitectura básico del estándar Z39.50 se apoya en este concepto de semántica en función del contenido. Es decir, cada servidor ofrece una visión de sus bases de datos en función del dominio o el ámbito en que nos encontremos, una representación virtual de los registros que contiene, donde la estructura lógica real de la base de datos permanece oculta, y sólo se refleja la que corresponde a la semántica de ese dominio.

SRU/SRW

Los protocolos SRU (Search/Retrieve URL) y SRW (Search/Retrieve Web Service) son la evolución del protocolo Z39.50, los tres estándares diseñados y mantenidos por la Library of Congress (LOC), como aporte para la estandarización la búsqueda y la recuperación de la información en ambientes Web.
Después de 20 años de trabajo, la International Maintenance Agency, de la LOC, empezó a desarrollar a partir de Z39.50 un protocolo para el intercambio, exportación de registros bibliográficos, búsquedas y recuperación de la información existente en Internet, es así como nace SRU/SRW, acrónimo que especifica, las búsquedas y la recuperación de información en ambientes de URL y de servicios Web.

Funcionalidad de SRU/SRW

SRU (Search / Retrieve Via URL) es un protocolo estándar de búsqueda para realizar consultas a través de Internet. Este protocolo utiliza CQL como lenguaje para realizar dichas consultas. Una de las principales ventajas de SRU radica en su sencillez. Un requerimiento SRU desde un cliente a un servidor es simplemente una URL HTTP, la cual consta de dos partes: la url base, y la parte de búsqueda. Los requerimientos SRU pueden incluir caracteres Unicode.
La url base es simplemente la dirección (y puerto) del servidor contra el que se hará la consulta. La parte de búsqueda consiste en una serie de parámetros, separados por el carácter"&", con la forma "clave=valor".

El servidor recibe el requerimiento y parsea la url, obteniendo los nombres de los parámetros y los valores de los mismos. Luego de verificar los nombres de cada parámetro, el servidor ejecuta la consulta especificada y genera una respuesta con formato XML, la cual es enviada al cliente para que la intérprete.

La idea de SRW es muy similar a la de SRU. La diferencia radica en que los mensajes entre el cliente y el servidor no son enviados a través de una URL, sino que se utiliza el formato XML sobre el protocolo HTTP, por medio de SOAP. Si bien a primera vista la diferencia parece ser mínima, este cambio acarrea un conjunto de ventajas muy significativas. Los beneficios que se desprenden directamente son: mejor soporte para extensión, autenticación, y características de WebServices.

CQL

CQL es un lenguaje formal para representar consultas para sistemas de recuperación de información como ser índices web, catálogos bibliográficos e información sobre colecciones de museos. CQL se diseñó con el objetivo de que las consultas puedan realizarse en un lenguaje que sea fácil de escribir y de leer, que sea lo más intuitivo posible, y sobre todo esto, que mantenga la expresividad de lenguajes de consulta más complejos. CQL trata de combinar en las expresiones la simplicidad e intuitividad de los lenguaje más simples, como es CCL, para realizar las consultas de todos los días, con la riqueza de los lenguajes de consulta más expresivos, como lo son SQL, XQuery, entre otros, para permitir la aplicación de conceptos más complejos cuando es necesario.

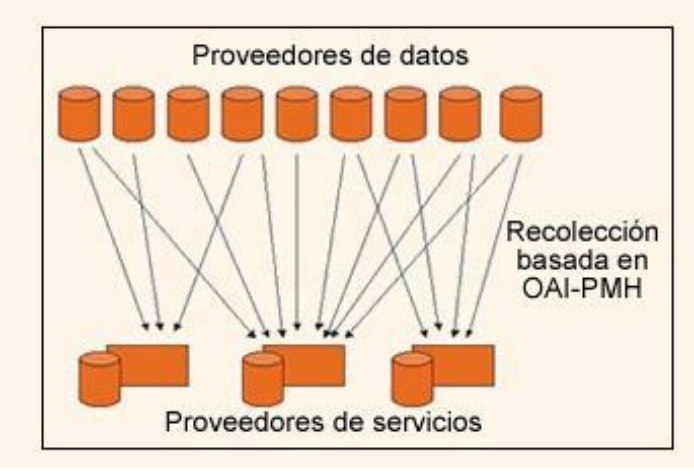
OAI-PMH

El protocolo OAI-PMH, en inglés Open Archives Initiative – Protocol for Metadata Harvesting, fue diseñado específicamente para transmitir datos de un sistema de información hacia otro a través de internet, tratando de ser sencillos en su implementación pero siendo muy claros en cuanto a las reglas de uso para promover el estándar y ser aplicado eficientemente al facilitar la difusión de contenidos sobre la Web, con el fin de convertirse en un mecanismo altamente usado para compartir recursos documentales a través de internet.

Este protocolo orientado a la recolección de metadatos, se trata de una simple interfaz que permite el acceso a los metadatos de contenidos o recursos en formato XML provenientes de distintas fuentes, plataformas y repositorios; se encarga exclusivamente de la comunicación de metadatos, no de los textos completos de los documentos que se referencian; requieren adhesión a los metadatos Dublin Core, al cual se adapta porque es muy sencillo, flexible y puede ser soportado por la mayoría de los proveedores de datos y servicios; pero permite y apoya otros formatos de metadatos.

Existen dos clases de participantes en el marco de OAI-PMH:

- Proveedores de datos: que son los propios archivos de depósitos de documentos que proporcionan los metadatos relacionados con los documentos que almacenan.
- Proveedores de servicios: recopilan los metadatos de diferentes archivos para proporcionar servicios de valor añadido a los usuario finales: sistemas de búsquedas e identificación, filtrado, alertas temáticas, medición del uso e impacto de documentos, etc.



[8]

[8] Imagen tomada de: http://travesia.mcu.es/portalnb/jspui/html/10421/1823/page2.htm

Figura 2.4 *Recolección de recursos basada en OAI-PMH*

Cabe destacar que el protocolo OAI-PMH:

- Genera y promueve estándares de interoperabilidad que facilitan la difusión, intercambio y accesibilidad a documentos de diferentes naturaleza.
- Permite almacenar los metadatos en un solo punto de la red donde ser realizan las diferentes consultas.
- Se ocupa de la gestión de la información, por lo que no define la creación de los metadatos, ni da los parámetros para realizar un consulta.
- Utiliza transacciones HTTP utilizado en la transferencia de contenidos web, donde está definida la sintaxis y la semántica que utilizan los clientes y servidores para comunicarse entre sí.
- Está basado en un modelo cliente/servidor que transmite preguntas y respuestas entre un proveedor de datos y un proveedor de servicios.

Las peticiones se emiten utilizando los métodos *GET* o *POST* del protocolo HTTP y constan de una lista de opciones con la forma de pares del tipo: clave=valor. Existen seis peticiones que un cliente puede realizar a un servidor:

- GetRecord: Utilizado para recuperar un registro concreto. Necesita dos argumentos: identificador del registro pedido y especificación del formato bibliográfico en que se debe devolver.
- Identify. Utilizado para recuperar información sobre el servidor: nombre, versión del protocolo que utiliza, dirección del administrador, etc.
- ListIdentifiers. Recupera los encabezamientos de los registros, en lugar de los registros completos. Permite argumentos como el rango de fechas entre los que queremos recuperar los datos.
- ListRecords. Igual que el anterior pero recupera los registros completos.
- ListSets. Recupera un conjunto de registros. Estos conjuntos son creados opcionalmente por el servidor para facilitar una recuperación selectiva de los registros. Sería una clasificación de los contenidos según diferentes entradas. Un cliente puede pedir que se recuperen solo los registros pertenecientes a una determinada clase. Los conjuntos pueden ser simples listas o estructuras jerárquicas.
- ListMetadataFormats. Devuelve la lista de formatos bibliográficos que utiliza el servidor.

El protocolo soporta múltiples formatos para expresar los metadatos, no obstante requiere que todos los servidores ofrezcan los registros utilizando Dublin Core no calificado, codificado en XML. Además de éste formato cada servidor es libre de ofrecer los registros en otro/s formatos adicionales (MARC por ejemplo). Un cliente puede pedir que los registros se le sirvan en cualquiera de los formatos soportados por el servidor.

Una vez cosechados los registros a través del protocolo OAI-PMH, éstos podrán ser indexados a través de una herramienta de indexación. Estas últimas son herramientas de software diseñadas especialmente para almacenar documentos textuales y proveer medios para su posterior recuperación a partir de palabras o frases de consulta. Están preparadas para lidiar con un volumen de documentos muy elevados, proporcionando además una serie de características adicionales que permiten en cierta medida, tratar con problemas de heterogeneidad, descartar palabras irrelevantes, dividir palabras, entre otras cosas.

Si bien en la actualidad existen múltiples herramientas clasificadas como *Indexadores de Texto*, sólo algunas han sido tan exitosas y marcado tendencia, como es el caso de Apache Solr. Esta herramienta es un motor de búsqueda el software de código abierto y está construido sobre el indexador de texto Apache Lucene. Este motor recibe documentos XML como entrada, los conduce a través de un conjunto de filtros y analizadores y genera un índice invertido sobre el cual se ejecutan los algoritmos de búsqueda. Esta forma de trabajo requiere en primera instancia la transformación de los documentos a indexar a un formato XML que respete el esquema (campos de datos contenidos en cada documento, como título, autor, fecha, etc) establecido en la configuración del motor, donde además se definen los filtros y analizadores por los cuales estos datos deben pasar antes de ser insertados en el índice.

Es importante mencionar que dicha herramienta se destaca por proveer tiempos de búsqueda del orden de los milisegundos sobre millones de registros, y aporta funciones que permiten optimizar los resultados obtenidos en cuanto a su relevancia.

2.3. Metadatos

El medio por el cual se realiza una catalogación de un documento es a través de los metadatos. El alcance más difundido que se observa del término lo presenta como “datos sobre datos” o “datos referentes a datos” para identificar archivos digitales de conjuntos de datos científicos,

sociales y geoespaciales. En el caso de Internet, representan la información electrónica dispersa y representan a la descripción bibliográfica de recursos electrónicos. Cuando se trata de bibliotecas digitales y de repositorios institucionales el uso de metadatos surge a partir de la aparición de algunos estándares de metadatos como es Dublin Core.

Los metadatos fueron creados para poder establecer una semántica capaz de operar y recuperar la información existente en la red. Son más ágiles que los estándares tradicionales y permiten que los objetos sean entendidos, compartidos y explotados de manera eficaz por todo tipo de usuarios a lo largo del tiempo y que además sean reutilizables.

Clasificación de los metadatos

Los metadatos pueden describir tanto colecciones de objetos como sus partes componentes. También incluyen datos sobre los procesos en los que están involucrados y definen relaciones entre los procesos. Las clases más generalizadas son: administrativos, estructurales y descriptivos.

- Metadatos administrativos: se refieren a las características y propiedades del recurso, facilitando la gestión y procesamiento tecnológico y físico de las colecciones digitales tanto a corto como a largo plazo. Incluyen información sobre la creación y el control de calidad, gestión de derechos, control de acceso y utilización y condiciones de preservación (ciclos de actualización, migración, entre otros). Incluyen datos técnicos como tipo y modelo de escáner, resolución, profundidad del bit, espacio de color, formato de archivo, compresión, fuente de luz, propietario, fecha de registro.

- Metadatos descriptivos: describen e identifican los recursos de información para su posterior búsqueda y recuperación, así como la localización cuando se trata de un entorno web. Contienen atributos físicos (medio, condición de las dimensiones) y atributos bibliográficos (título, autor/creador, idioma, palabras claves). En esta categoría se encuentran el formato MARC y Dublin Core (DC).

- Metadatos estructurales: facilitan la navegación y presentación de los recursos electrónicos, proporcionando información sobre la estructura interna de los mismos, así como la relación y unión entre los diferentes materiales que forman el objeto digital. Dicha información puede ser página, sección, capítulo, partes, índices, tabla de contenido, etc.

[9]

Figura 2.5 *Tipos de Metadatos*

Todos estos tipos de metadatos conviven con el objeto digital y existen diferentes modelos para cada uno. Los administrativos y estructurales en su mayoría son gestionados automáticamente por el software, cuando se crea el recurso. Los metadatos descriptivos son ingresados por el catalogador.

Dublin Core

Dublin Core es uno de los esquemas de metadatos más conocidos para catalogar documentos en Internet. Este estándar se creó a partir de un taller de metadatos realizado precisamente en la ciudad de Dublin, Ohio (Estados Unidos) en 1995 en el que participaron el NCSA (National Center for Supercomputing Applications) y OCLC (On Line Library Computer Center), junto con representantes de la IETF (Internet Engineering Task Force) y en el que bibliotecarios, proveedores de contenido y expertos en lenguaje de marcado pretendieron desarrollar estándares para describir los recursos de información y facilitar su recuperación. La primera versión generó grandes expectativas, pues se componía únicamente por un pequeño conjunto de descriptores con los cuales se podía describir en parte y de forma muy sencilla un recurso.

[9] Imagen tomada de: http://www.bn.gov.ar/descargas/catalogadores/ponencias/251109_09a.pdf
[9]

Ya en el año 2001 se aprobó como norma estatal el conjunto de elementos de Dublin Core según el organismo de normalización de los Estados Unidos, dando lugar a la norma Z39-85:2001 DUBLIN CORE METADATA SET ELEMENT.

Con el auge de internet y principalmente por su nivel de penetración en el mercado y en el ambiente académico, se comenzó a hacer un uso cada vez más extenso del estándar, razón por la cual en el 2003 se aprobó como norma ISO 15836, basándose ciertamente en la norma Z39-85.

Desde la creación de este estándar surgió el DCMI (Dublin Core Metadata Initiative), organización cuya misión es proveer estándares para buscar, compartir y administrar información, basándose en los principios como la construcción bajo consensos abiertos, neutralidad tecnológica y modelos de negocios.

Estructura Dublin Core

El conjunto de elementos de metadatos Dublin Core es un conjunto de metadatos previsto para describir documentos. Los elementos poseen etiquetas descriptivas que pretenden transmitir un significado semántico a los mismos.
Cada elemento es opcional y puede repetirse. Además, los elementos pueden aparecer en cualquier orden. Aunque algunos entornos, como HTML, no diferencian entre mayúsculas y minúsculas, es recomendable escribir correctamente cada metadato, según su definición, para evitar conflictos con otros entornos, como por ejemplo XML.

El estándar Dublin Core se compone de 15 descriptores básicos en los cuales se extraen las características más importantes del recurso a catalogar. Estos se agrupan en 3 grandes conjuntos.

- Elementos relacionados principalmente con el *contenido* del recurso:
    - Title (titulo)
    - Subject (tema)
    - Description (descripción)
    - Source (fuente)
    - Language (lenguaje)
    - Relation (relación)
    - Coverage (cobertura).

- Elementos relacionados principalmente con el recurso cuando es visto como una *propiedad intelectual:*
    - Creator (autor)
    - Publisher (editor) y, otras colaboraciones
    - Contributor (otros autores/colaboradores)
    - Rights (derechos).
- Elementos relacionados principalmente con la *instanciación* del recurso:
    - Date (fecha)
    - Type (tipo de recurso)
    - Format (formato)
    - Identifier (identificador)

Etiqueta del elemento Dublin Core	Definición
dc.title	Nombre por el que se conoce el recurso a catalogar.
dc.subject	Tópico o clasificación del recurso, también puede llevar información sobre las palabras claves asociadas.
dc.description	Explicación del recurso, esta descripción puede incluir un resumen, tabla de contenidos o una representación gráfica del recurso.
dc.source	Cadena de texto que describe de forma única al recurso o el sitio donde se encuentra el recurso.
dc.language	Idioma en el que se encuentra el recurso.
dc.relation	Es el enlace a un segundo o tercer

	recurso que se relaciona con el recurso original. Al igual que el descriptor source es un enlace único e inequívoco del recurso en el sistema.
dc.coverage	Describe la ubicación geoespacial del recurso.
dc.creator	Entidad o persona que fue autor o creador del recurso catalogado.
dc.publisher	Entidad o persona responsable de colocar disponible o accesible el recurso.
dc.contributor	Entidad responsable de hacer colaboraciones al contenido del recurso.
dc.rights	Información acerca de los derechos de autor asociados al recurso, para que el usuario final conozca sus condiciones de uso y acceso.
dc.date	Fecha en la cual se creó o se colocó disponible el recurso.
dc.format	Describe el tipo de formato de datos en el cual se encuentra el recurso.
dc.type	Es el tipo de información o el tipo de recurso como tal, por ejemplo, tesis,

	libros, artículos, etc.
dc.identifier	Texto que identifica al recurso, por ejemplo, ISSN, ISBN, etc.

Figura 2.6 *Etiquetas descriptivas de Dublin Core*

Este estándar de metadatos es usado en el mundo en diferentes disciplinas de estudio, ya que al no ser restringido a un perfil de aplicación específico, puede ser implementado sobre cualquier sistema de información y ser interoperable con otros sistemas de información que ofrezcan sus contenidos según las etiquetas definidas por Dublin Core.

MARC

MARC es un estándar para la catalogación de documentos; fue desarrollado por la Biblioteca del Congreso de Estados Unidos en los años 70, cuando comenzaron a utilizar computadoras en dicha biblioteca. Este estándar especifica las normas para la presentación de los atributos de un documento con el fin de ser legible por máquina, por ello su nombre MARC (Machine Readable Cataloging). Desde su creación, ha sido el más utilizado por las bibliotecas para sus catálogos bibliográficos, ya que facilitó en gran parte la organización de todos los documentos existentes con el objetivo de mejorar su capacidad de recuperación.

MARC se basa en la norma ISO 2709, una de las concernientes a las descripciones bibliográficas y el intercambio de la información por medios magnéticos, en el cual se especifican los requerimientos para la generación de un formato de intercambio adecuado para descripciones bibliográficas[10].

Actualmente, la mayoría de catálogos bibliográficos que tienen las bibliotecas de todo el mundo utilizan este estándar para realizar la catalogación de su acervo bibliográfico. Un registro MARC evidencia la compresión de los datos ya que no almacena los nombres de los campos como tal, sino que éstos se encuentran codificados en las etiquetas y subcampos para permitir de cierta forma un ahorro en el almacenamiento de la información.

[10] International Organization for Standardization Format for Bibliographic Information Interchange on Magnetic Tape, ISO 2709

Componentes

Un registro MARC contiene la información de un ejemplar bibliográfico de manera codificada, y está conformado principalmente por tres componentes:

- Cabecera: elementos de información que esencialmente proveen datos para el procesamiento del registro. Los elementos de información contienen números o valores codificados y se identifican por la posición relativa del carácter. La Cabecera posee una longitud fija de 24 caracteres y constituye el primer campo de un registro MARC.
- Directorio: conjunto de entradas que corresponden a la etiqueta, la longitud y el punto de inicio de cada campo variable dentro de un registro. Cada entrada posee una longitud de 12 caracteres de posición. Las entradas del Directorio correspondientes a los campos variables de control aparecen, primero, en una secuencia que sigue el orden numérico de las etiquetas de campo; le siguen las entradas de los campos de datos variables en orden ascendente de acuerdo al primer carácter de la etiqueta. La secuencia almacenada de los campos de datos variables del registro no corresponde necesariamente al orden de las entradas correspondientes del Directorio. Las etiquetas duplicadas se distinguen únicamente por la localización de los campos respectivos dentro del registro. El Directorio finaliza con un carácter terminador de campo (ASCII 1E hex).
- Dentro de un registro bibliográfico MARC, los datos se organizan como campos variables, y cada uno de estos se identifica mediante una etiqueta numérica de tres caracteres, que se almacena en la entrada correspondiente en el Directorio. Cada campo termina con un carácter terminador de campo. El último campo de un registro finaliza tanto con un terminador de campo como con un terminador de registro (ASCII 1D hex).

Estructura de los campos variables de datos

En la siguiente tabla se puede observar la estructura de los campos variables de datos. Cada una de las X representa un número y cada combinación de los mismos representa una característica diferente del ítem a catalogar.

Etiqueta	Descripción
0XX	Información de control, números de identificación y clasificación, etc.

1XX	Asientos principales.
2XX	Títulos y párrafo del título (título, edición, pie de imprenta).
3XX	Descripción física, etc.
4XX	Menciones de serie.
5XX	Notas
6XX	Campos de acceso temático.
7XX	Asientos secundarios diferentes a los de materias y series; campos ligados.
8XX	Asientos secundarios de series, existencias, etc.
9XX .	Reservados para implementación local.

Figura 2.7 *Estructura de los campos variables de datos MARC*

A continuación se detallarán algunos de los campos más importantes, los cuales fueron utilizados en el desarrollo de la herramienta:

Etiqueta	Descripción
020	Número internacional normalizado para libros (ISBN).
022	Número internacional normalizado para publicaciones seriadas (ISSN).
100	Asiento principal - Nombre personal. Nombre de persona utilizado como asiento principal en un registro bibliográfico.
245	Área del título y mención de responsabilidad de la descripción bibliográfica de una

	obra.
260	Publicación, distribución, etc. (pie de imprenta). Información relacionada con la publicación, impresión, distribución, emisión, puesta en circulación o producción de una obra.
300	Descripción física. Incluye extensión, dimensiones y otros detalles físicos tales como la descripción del material acompañante y el tipo y tamaño de la unidad.
362	Fechas de publicación y/o designación secuencial.

Figura 2.8 *Campos MARC mas relevantes*

Capítulo 3

MetaLib Service - Herramienta de Software para la Metabúsqueda

3.1. Introducción

En este capítulo se describe MetaLib Service, una herramienta pensada para brindar un servicio de suscripción y realización de metabúsquedas. La misma puede ser utilizada por cualquier persona, institución o entidad que necesite realizar búsquedas particulares o puntuales en determinadas áreas y se encuentre con la necesidad de localizar documentos específicos de manera rápida y eficaz.

La herramienta se compone de un módulo principal, que tiene como objetivo fundamental realizar una recolección de recursos a partir de una petición de búsqueda realizada por un usuario. Cuenta con una interfaz web, a través de la cual un administrador puede configurar ciertas características particulares al funcionamiento de la herramienta. Esta interfaz, permite también la realización de consultas en línea sobre un conjunto específico de catálogos disponibles en internet.

Se diseñó además una API para aplicaciones de terceros que estén interesadas en consumir el servicio de metabúsqueda y ofrecerlo a sus usuarios. Esta API se encarga de abstraer el funcionamiento principal de la herramienta, y permite así una integración clara y transparente para los desarrolladores de otros sistemas. Dicha integración requiere de una configuración básica que permitirá autenticar a los usuarios/aplicaciones e interactuar de forma remota con la herramienta.

MetaLib Service está desarrollada con software de uso libre y es totalmente portable, brindando la posibilidad de ser utilizada bajo cualquier sistema operativo. A continuación se especifica la funcionalidad de la herramienta.

3.2. Herramienta de Software para la metabúsqueda

Perfiles de Usuarios

La herramienta cuenta con cuatro perfiles de usuarios diferentes:

- Usuario Registrado
- Usuario de API
- Usuario No Registrado
- Usuario Administrador

Independientemente del tipo de usuario, se definirá a partir de valores preestablecidos y de forma automática para cada uno de estos, un conjunto de catálogos que se utilizarán en cada transacción.

El primer tipo de usuarios, son aquellos que se registraron en la herramienta vía web. Al realizar ésta operación deberán indicar un nombre de usuario que deberá ser único en el sistema, mediante el cual se lo logrará identificar. A partir de ésta inscripción, los nuevos usuarios quedarán habilitados para realizar consultas a los catálogos que tengan asociados. Este tipo de usuarios, sólo podrá realizar búsquedas de tipo síncronas a través de la interfaz web, obteniendo los resultados de forma inmediata (más adelante se describen los tipos de búsquedas síncronas y asíncronas).
En la siguiente imagen se puede ver el formulario de búsqueda web para este tipo de usuario.

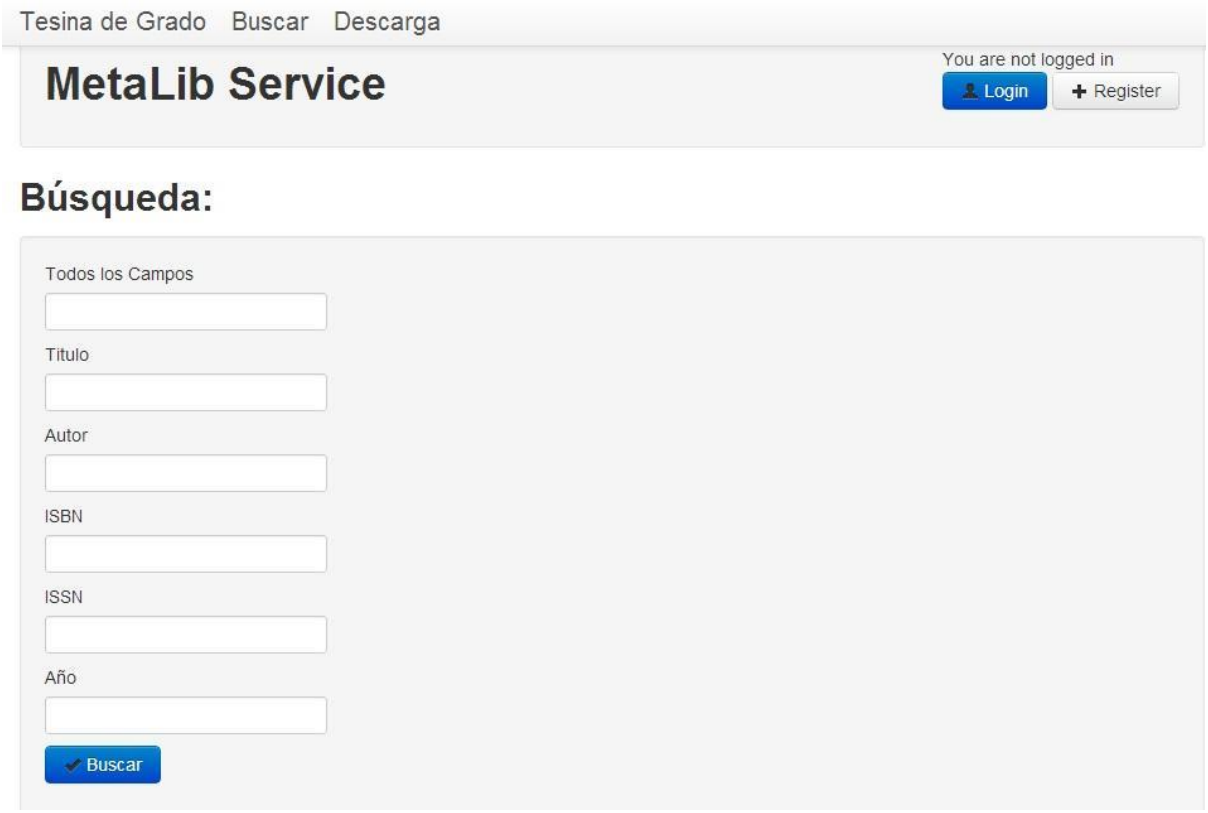


Figura 3.1 *Formulario web para solicitud de búsqueda síncrona*

El siguiente tipo de usuario corresponde a aquellos que deseen utilizar este servicio a través de una aplicación de terceros. Para esto, luego de descargar la librería[11] desde el sitio oficial, deberán solicitar a través del formulario de adhesión a la herramienta, la clave de identificación (identificationKey, también conocida como API KEY) mediante la cual podrán ser identificados en cada una de las peticiones que se hagan al servidor desde su aplicación a través de la librería. En dicho formulario, el interesado, deberá ingresar una descripción detallada de la aplicación que utilizará el servicio. Esta información será evaluada por un administrador, y quedará a su criterio la aceptación/rechazo de dicha solicitud. El usuario será notificado vía correo electrónico de la decisión tomada y de ser aceptado, se le hará conocer su API KEY. Es importante mencionar que para este tipo de usuarios, quedarán habilitadas tanto el tipo de búsqueda síncrona como asíncrona. Si la solicitud fue rechazada, el usuario no podrá hacer uso de la API debido a que nunca dispondrá del API KEY y sólo podrá realizar búsquedas síncronas via web. Si la API KEY ingresada por el usuario es inválida, no se le permitirá realizar la metabúsqueda informando la situación al mismo.

[11] Para mayor detalle, dirigirse al apartado Librería.

El tercer tipo de usuario son aquellos que desean realizar algún tipo de pedido vía web sin estar registrados en la herramienta. Las consultas se harán sobre una lista acotada de servidores de consulta, provocando de este modo una recolección menos abarcativa que en el resto de los casos. Éste tipo de usuario está habilitado para realizar búsquedas síncronas.

Por último, el usuario administrador es el encargado de:

- Aceptar/rechazar todos las solicitudes de alta de usuario para el uso de la librería. En ambos casos, se le enviará un correo electrónico al usuario indicando la decisión tomada.
- Agregar, modificar o eliminar servidores. Estos servidores son los que están disponibles al momento de realizar la búsqueda. El administrador será capaz de agregar nuevos servidores; para ello se tendrá que indicar mínimamente la url del catálogo y a que protocolo se encuentra asociado dicho servidor. Al eliminar un catálogo, se inhabilitan automáticamente todas las búsquedas pendientes a dicho servidor.
- Editar o eliminar los usuarios de la herramienta. La edición comprende la reasignación de repositorios y catálogos en línea, es decir, el administrador tendrá el poder de modificar los servidores de consulta que tenga asociado cada uno de los usuarios. A su vez podrá eliminar a un usuario de la herramienta, provocándose de esta forma que dicho usuario quede inhabilitado para ingresar a través de su identificación personal.

Figura 3.2 *Listado de usuarios registrados en el sitio.*

Búsquedas

Tipos de búsquedas

Existen dos maneras de realizar las búsquedas, síncronas o asíncronas. Estas se diferencian por el momento en que se realiza el tratamiento de la solicitud. Para las síncronas se brinda una atención inmediata, en cambio para el caso asíncrono, el momento de atención .de la petición estará condicionado por un evento externo.

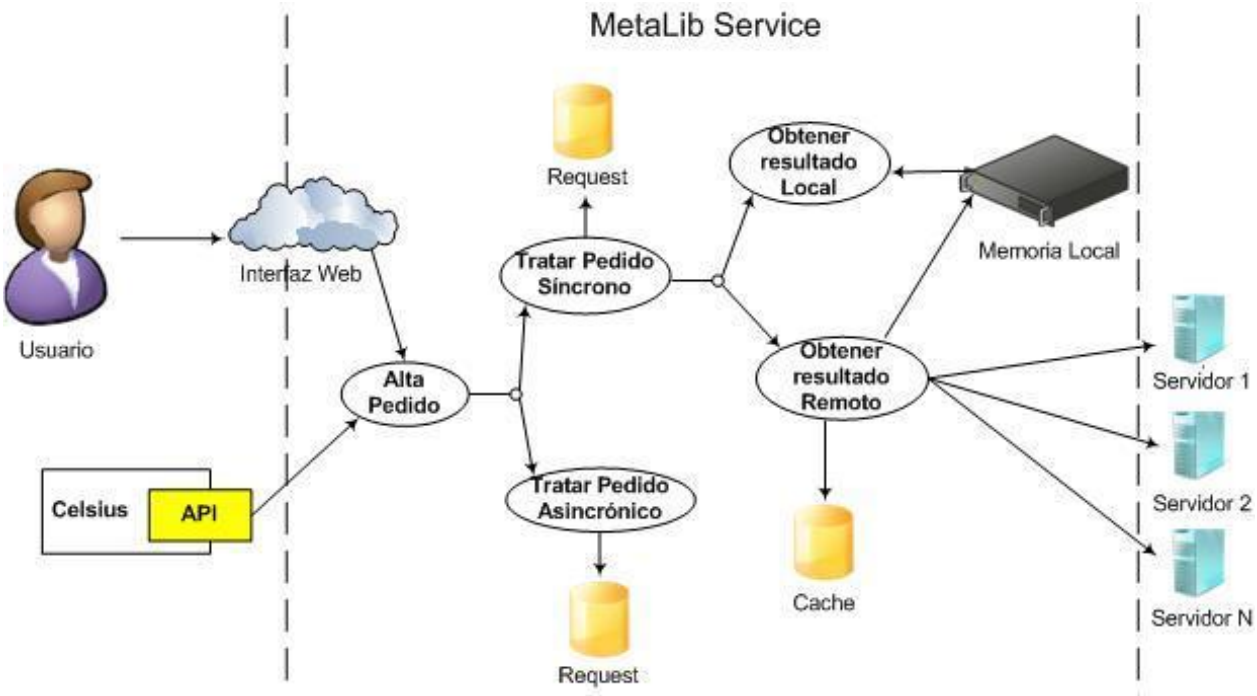


Figura 3.3 *Tipos de Búsquedas*

Asícronas

Para las búsquedas asíncronas, existe un proceso que se encargará de seleccionar una solicitud entre un conjunto de peticiones pendientes de tratar, con el fin de realizar la recolección de recursos que satisfagan dicha petición.

Este proceso será ejecutado a través de una tarea programada (cron) que permite automatizar la ejecución periódica de un script o tareas cada ciertas unidades de tiempo. Esta operación deberá ser detallada en el fichero de configuración crontab[12], en el cual será posible definir un conjunto indeterminado de acciones que se ejecutarán automáticamente. Esta proceso de automatización de tareas se suele llamar *cron job.*

[12] El comando crontab -l permite visualizar el crontab del usuario; el comando crontab -e permite la edición.

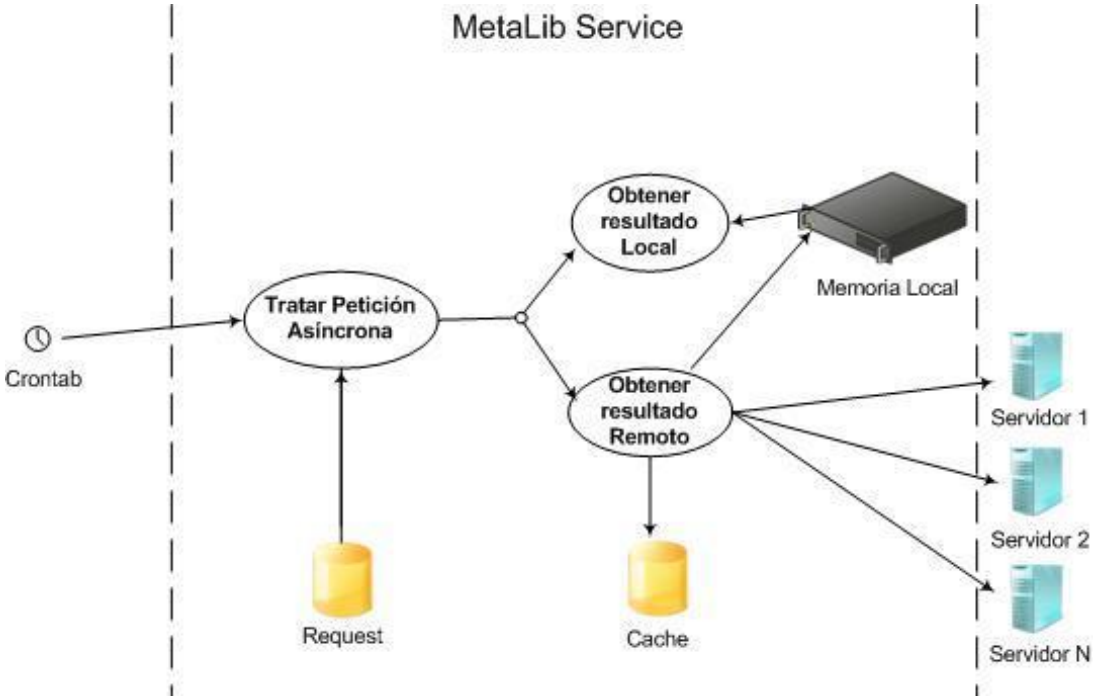


Figura 3.4 *Atención de Pedido de Búsqueda Asíncrono.*

Síncronas

Las peticiones síncronas son atendidas de forma inmediata. Es decir, una vez ingresada y registrada la petición en el sistema, se procederá automáticamente a la recolección distribuida y posterior retorno de recursos al usuario que disparó la petición. Cabe destacar que el proceso de búsqueda puede ser particularmente lento (del orden de varios minutos), pues depende de la cantidad de datos que son recolectados, de los tiempos de respuestas de cada uno de los catálogos y repositorios consultados, y de la cantidad de los catálogos/repositorios sobre los que se realizan las búsquedas. Por este motivo, para las búsquedas realizadas a través de la interfaz web se ha acotado el número de repositorios y catálogos en línea.

Mecanismo de búsqueda

El mecanismo de búsqueda tiene como objetivo realizar la recolección y unificación de recursos bibliográficos a partir de la información introducida por el usuario. Dicha tarea se realiza a través de un conjunto determinado de catálogos que se encuentran asociados a la herramienta. El administrador de la aplicación puede determinar para cada cliente en particular, el conjunto de catálogos relacionados a cada una de sus búsquedas.

El procedimiento de recolección de recursos se puede dividir en varias etapas:

- completar campos de búsqueda
- registro de la petición en el sistema
- búsqueda en los servidores asociados al usuario que realizó la petición
- generación de documento único de respuesta para el usuario

Completar campos de búsqueda

Esta etapa es el punto de partida de la operación y está relacionada plenamente con el usuario que desea realizar el pedido, ya que él será el encargado de completar los filtros que más se adecuen al pedido que desea buscar, para luego ejecutar la búsqueda lo más exacta y restringida posible.

Para ello, la herramienta cuenta con varios parámetros para realizar la búsqueda, los cuales son:

- Título: está relacionado al título de la revista, artículo, libro, tesis, entre otros, que se desea buscar. Cuanto más exacto es el nombre introducido, más exacta serán los resultados recuperados de la consulta.
- Autor: representa a uno o varios autores del libro, revista, artículo, entre otros.
- ISSN: en caso de contar con este dato, al ser el identificador único de la revista, se restringe al máximo la búsqueda.
- ISBN: sucede lo mismo que con el ISSN, con la diferencia que éste dato es el identificador único de un libro.
- Año: éste campo representa la fecha de publicación del recurso que se desea localizar..
- Comodín: es importante entender el objetivo que persigue éste parámetro para no distorsionar la consulta que se realizará luego. Si bien la creación de la consulta utilizada para la recolección de datos quedará a criterio del servidor, al incorporarse valor a este filtro básicamente se estára indicando que se localice dicha cadena de caracteres en cualquiera de los parámetros antes mencionados.

Ingreso de la petición

En primer lugar se evalúa a través de un mecanismo de detección de búsquedas repetidas[13] sí la petición ingresada ya se había realizado previamente. De ser así se satisfará la búsqueda desde la memoria local de la herramienta, caso contrario, la recolección de recursos será de forma remota, a través de cada uno de los servidores relacionados al usuario.
En ambos casos, se almacena localmente la petición a fin de poder llevar un control histórico sobre los pedidos que se realizan, y con el principal objetivo de poder reutilizar el resultado de las búsquedas anteriores[14] y, adicionalmente, poder realizar mediciones y estadísticas en un futuro.

Búsqueda en los servidores asociados

Una vez que la petición de búsqueda ha sido registrada, la aplicación procede a realizar la recolección en cada uno de los servidores relacionados al usuario que está solicitando el pedido. Como cada uno de los servidores trabaja con un protocolo particular, la consulta se creará en base al protocolo que tenga asociado dicho servidor.

Actualmente el servicio permite integrar cualquier servidor que trabaje mediante el protocolo Z39.50 o mediante el protocolo OpenSearch. Sin embargo, la herramienta se ha diseñado de forma tal que resulte simple la incorporación de nuevos protocolos, asegurando que esta incorporación no interferirá con los protocolos que ya están en funcionamiento. Sólo será necesario encapsular el comportamiento clave para que los servidores que utilicen dicho protocolo puedan ser consultados de forma sencilla y transparente. Luego de implementar tal comportamiento, con una simple llamada se podrá realizar la búsqueda a dicho servidor.

[13] Se especifica en el apartado Determinación y/o descubrimiento de búsquedas repetidas.
[14] Se especifica en el apartado Memoria Caché

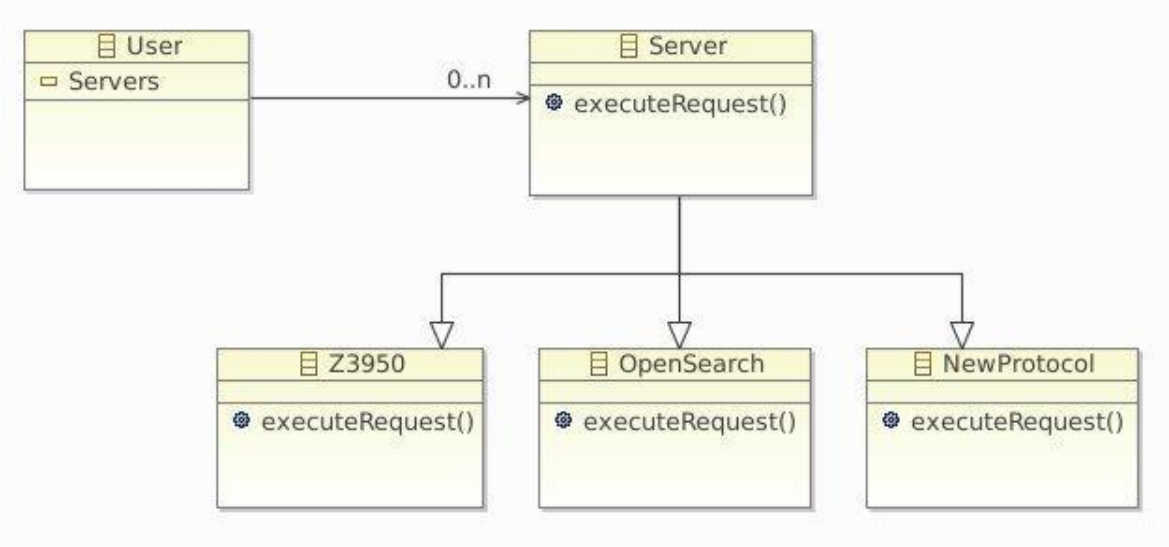


Figura 3.5 *Diagrama de Clases*

La elección de los protocolos no fue un tema menor, ya que en la actualidad existe gran variedad de bibliotecas digitales que trabajan bajo múltiples y diferentes protocolos. Z39.50 fue seleccionado debido a que es un protocolo estándar internacional para la comunicación entre sistemas informáticos. El otro protocolo que ha sido tenido en cuenta es OpenSearch, debido a que es un estándar que tiene como objetivo integrar de forma más sencilla las consultas a buscadores por parte de cualquier aplicación. OpenSearch es relativamente nuevo y, hoy en día, se está convirtiendo en un estándar de facto en los principales sistemas de búsquedas. Además, a diferencia de lo que ocurre con los otros estándares, OpenSearch es extensible, y dentro de las propias especificaciones se indica cómo realizar dichas extensiones. La mayor parte de los buscadores, navegadores y sitios web soportan OpenSearch y muchos fabricantes de software están comenzando a utilizar este formato para proporcionar acceso a los contenidos de sus aplicaciones.

Otro aspecto determinante al momento de decidir el conjunto de protocolos a tener en cuenta, fue el tipo de información con el que trabaja cada uno. Es decir, mediante Z39.50 es posible acceder principalmente a recursos como libros, revistas, capítulos o artículos de las mismas, y está orientado principalmente a catálogos abiertos en línea de diferentes bibliotecas. En cambio, el protocolo OpenSearch recolecta información sobre la producción académica de distintas instituciones, orientado principalmente a Repositorios Institucionales como el caso del Servicio de Difusión de la Creación Intelectual de la Universidad Nacional de La Plata. Con este enfoque, las búsquedas realizadas tendrán un panorama más abarcativos de datos, ya que al

retornar diferente tipos de recursos, los protocolos se complementan entre sí y el campo de búsqueda se extiende.

Como se ha mencionado anteriormente, y como es de esperarse, al trabajar con distintos protocolos, las consultas a los repositorios se realizan de manera particular dependiendo del protocolo asociado al mismo.

Consultas

En el caso de tratarse de un servidor que trabaja con Z39.50, las consultas se realizan a través de la librería Yaz; es un conjunto de herramientas que proporciona fácil acceso a los protocolos Z39.50 y SRW/SRU, y además posee un conjunto de herramientas de alto nivel para la aplicación de las funciones de servidor y cliente. Una de las principales ventajas que posee esta librería es que tiene soporte de la notación polaca inversa (RPN).

Dentro de la librería Yaz, se destacan las siguientes funciones:

- yaz_connect: realiza la conexión al servidor Z39.50.
- yaz_ccl_conf: ésta función es en la que más se hará hincapié ya que es la encargada de configurar el analizador CLL(Lenguaje de Consulta de Comandos), es decir, configura la consulta analizadora CCL para un servidor con definiciones de puntos de acceso (Calificadores CCL) y su asignación al RPN.

  Los parámetros que recibe son:

  - id: representa al recurso de conexión devuelto de la función yaz_connect.
  - config: es un arreglo de configuración. Cada clave del arreglo es el nombre de un campo CLL y el correspondiente valor que mantiene es una cadena que especifica una asignación RPN. Dicha asignación es una secuencia de el tipo de atributo y de los valores de los atributos pares. Ambos valores son separados por un signo (=).

    El analizador CCL de la herramienta está configurado para soportar el árbol de los campos CCL: *título*, *ISSN, ISBN, año, autor y comodín*.

Campo	CCL
Título	1=4
Autor	1=1
Isbn	1=7
Year	1=31
Issn	1=8
Comodín	1=1016

Figura 3.6 *Campos CCL (Lenguaje de Consulta de Comandos)*

Como se puede observar en la tabla, cada campo es asignado a su equivalente en el conjunto de atributos definidos en BIB-1.

La elección de estos campos se basó en la importancia que tienen dichos valores en los recursos bibliográficos, además de que, a partir del ISSN o ISBN, la búsqueda se acota mucho más a los resultados esperados.

- yaz_cll_parse: realiza la invocación a un analizador CLL, es decir, asigna una consulta específica CLL al RPN. En la herramienta, la consulta CLL es la que se mencionó en la función previa.
- yaz_syntax: especifica la sintaxis de registro preferida para la recuperación. En el caso particular de la herramienta, se ha tomado la decisión de trabajar con registros Marc (usmarc) ya que hoy en día es el estándar de recuperación y comunicación de información bibliográfica y relacionada más conocido y usado en las bibliotecas. En caso que el catálogo retorne los datos, por ejemplo en formato Dublin Core, automáticamente se machean los campos DC con los campos Marc y de ésta manera se unifica la estructura de datos retornada, para luego poder hacer un tratamiento unívoco de los registros.
- yaz_range: especifica un rango de registros a recuperar. En la herramienta éste valor se encuentra configurable.
- yaz_search: se prepara para una futura búsqueda (de la consulta RPN) en la conexión dada.
- yaz_wait: al llamarse esta función, todas las peticiones que están pendientes de ejecutarse, se van a completar.
- yaz_hits: retorna la cantidad de registros que devolvió la consulta realizada.
- yaz_record: recupera un registro del resultado de la consulta en una posición dada. Esta función recibe en qué tipo será devuelto el registro recuperado, en el caso de la herramienta se ha optado por el tipo XML. Cabe destacar que al tratarse de la recuperación de registros Marc, ésta función automáticamente retornará un MARCXML.

Esta decisión estuvo basada en que, una vez recuperados los registros, los datos serán procesados por un analizador XML.

En caso de tratarse de un servidor que trabaja bajo el protocolo OpenSearch, la creación de las consultas son más sencillas y directas debido a que la lógica de creación de las mismas queda a criterio de cada uno de los servidores. De esta manera queda limitada la actividad de la herramienta, ya que sólo será posible enviar un parámetro que se considere clave para la identificación de los recursos.

Ejemplo resultado en formato XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
   <search_date>Sun Nov 18, 2012 20:19</search_date>
   <servers>
      <entry>
         <name>Center for Research Lib</name>
         <url>catalog.crl.edu:210/INNOPAC</url>
         <records>
            <entry>
               <title>Nature.</title>
               <issn>0028-0836</issn>
               <publication_site>Washington></publication_site>
               <editorial>Macmillan Journals Ltd.</editorial>
               <number_page>--</number_page>
               <chronological_data>Began in Nov 1869</chronological_data>
            </entry>
            <entry>
            . . .
            </entry>
            <entry>
            . . .
            </entry>
         </records>
      </entry>
      <entry>
      . . .
      </entry>
      <entry>
      . . .
```

```
        </entry>
    </servers>
</result>
```

Si bien esto parece facilitar las cosas, resulta una tarea compleja decidir cuál de los campos será enviado para realizar la consulta. La principal desventaja que presenta la interacción con este tipo de catálogos, reside en la necesidad de tener un conocimiento mínimo sobre el funcionamiento del servidor. Esto resulta un impedimento o limitación al momento de cargar servidores de este tipo en la aplicación.

En éste tipo de servidores, el punto clave está en obtener el descriptor del servicio, para obtener a partir de este, aquellos campos claves que nos permitirán la conexión con el repositorio. En la herramienta que aquí se presenta, dicho valor es configurable al momento de ingresar un nuevo servidor en la base de datos.

Generación de documento único

Una vez finalizada la recolección de registros de todos los repositorios involucrados en la búsqueda, se genera un documento unívoco asociado a la consulta que contendrá el resultado final de la búsqueda. Para la creación de dicho documento, se estableció una estructura jerárquica con el fin de mantener normalizado los resultados y facilitar el posterior procesamiento de los datos.

La estructura de dicho archivo, posibilita una recuperación eficiente de los recursos, además de permitir realizar una discriminación de la cantidad y tipos de registros encontrados en cada uno de los repositorios consultados, así como la fecha y hora en que fue realizada la consulta. Este último dato es de gran importancia al momento de determinar la validez de la información.

La herramienta permite una libre elección en el formato en el cual se retorna el resultado, ya que tiene la capacidad de retornar resultados tanto en formato XML como en JSON.

En este caso, fueron seleccionados estos formatos ya que ambos lenguajes son los más utilizados para el intercambio de datos, siendo XML un lenguaje con historia, estándar, seguro, de fácil lectura para el usuario, con estructura jerárquica y además cuenta con funciones claras para el manejo de la estructura, en caso de los archivos JSON ocupan menor tamaño, son por lo general más pequeños y al momento de realizar el tratamiento de los datos, requieren el mínimo procesamiento para una correcta recuperación de la información.

Determinación y/o descubrimiento de búsquedas repetidas

Debido a que el proceso de recolección de datos a partir de cada uno de los catálogos resulta ser lento y costoso, se ideó un mecanismo mediante el cual se determina la existencia de búsquedas duplicadas con el fin de reducir el tiempo de acceso a la información por parte de los distintos clientes. De ésta manera se evitará en algunos casos, el costoso procedimiento de recolección de registros. Para aquellos casos en los que la herramienta determine una duplicidad en consultas realizadas, el usuario accede rápidamente a una respuesta válida previamente elaborada y almacena en memoria por la aplicación.

Para cada una de las solicitudes de búsqueda entrante, se comparan sus datos contra los datos de cada una de las solicitudes previamente realizadas por otros usuarios de la aplicación. Para lograr entender el proceso de detección de búsquedas duplicadas, comenzaremos analizando la estructura que representa dicha búsqueda.

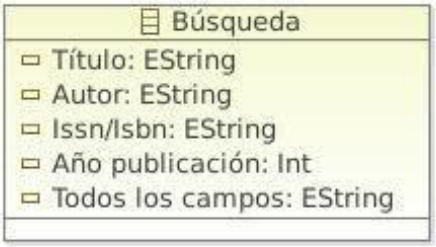


Figura 3.7 *Estructura utilizada para crear una búsqueda.*

Se consideró necesario reconocer y/o detectar campos duplicados entre las dos búsquedas involucradas en el proceso de análisis y detección de duplicidad. Para solucionar esto se utilizarán dos funciones nativas de php, por un lado la función *similar_text*, que calcula la similitud entre dos strings, contando la coincidencia de caracteres, y por el otro, la función *levenshtein*, que realiza el cálculo de la distancia Levenshtein entre dos cadenas de texto. Ambas, usadas en conjunto y con valores de operación apropiados a cada caso en particular, resultan ser muy efectivas.

La distancia Levenshtein se define como el número mínimo de caracteres que se tienen que sustituir, insertar o borrar para transformar una cadena(*str1*) en otra cadena(*str2*). La complejidad del algoritmo es *O(m*n)*, donde *n* y *m* son la longitud de *str1* y *str2*. En su forma más simple la función tomará sólo los dos strings como parámetros y calculará sólo el número de operaciones de insertar, reemplazar y eliminar, necesarias para transformar *str1* en *str2*. Esta función devuelve la distancia Levenshtein entre los dos strings argumentos ó -1, si uno de los strings argumentos es mayor que el límite de 255 caracteres.

La función *similar_text* calcula la similitud entre dos strings. Esta implementación utiliza llamadas recursivas que pueden o no pueden acelerar todo el proceso. La complejidad de este algoritmo es O(N**3) donde N es la longitud del string más largo. Esta función devuelve el número de caracteres coincidentes en ambos strings.

Como primer tarea se debe determinar una cota, o punto de inflexión para cada una de las funciones utilizadas en el proceso de comparación de las búsquedas. La comparación de dicho punto de inflexión, con el valor obtenido de la comparación entre dos campos, determinará el éxito o fracaso de dicha tarea. La elección de la cota se definió luego de realizar un análisis mediante el cual se pusieron a prueba un conjunto de valores previamente seleccionados, teniendo en cuenta el campo de aplicación que tendrá la herramienta.
Para todos los ejemplos mencionados a continuación, se utilizó la misma cota o punto de inflexión en ambas funciones:

SimilarText = 0
Levenshtein = 0

Comparación Isbn/Issn	SimilarText	SimilarText Ecuación	Levenshtein
0021-9607 = 0021-9606	8	1	1
0021-9606 = 0021-9606	9	0	0
1021-9606 = 0021-9606	8	1	1
0021-8606 = 0021-9606	8	1	1

Figura 3.8 *Comparación de los campos Isbn/Issn a través de las funciones* `SimiliarText` *y* `Levenshtein`

Comparación Años	SimilarText	SimilarText Ecuación	Levenshtein
2008 = 2009	3	1	1
209 = 2009	3	0,5	1
2007 = 2009	3	1	1
1009 = 2009	3	1	1
2009 = 2009	4	0	0

Figura 3.9 *Comparación de los campos Año a través de las funciones* `SimiliarText` *y* `Levenshtein`

Comparación Título	SimilarText	SimilarText Ecuacion	SimilarText Promedio	Levenshtein
Belen Magali Forte = Belen Magali Forte	18	0	50	0
B. Magali Forte = Belen Magali Forte	14	2	43.75	4
Belen M. Forte = Belen Magali Forte	13	2.5	41.93	5
Forte Belen Magali = Belen Magali Forte	12	6	33.3	12
Belen Forte = Belen Magali Forte	11	3.5	37.93	7

Figura 3.10 *Comparación de los campos título a través de las funciones* `SimiliarText` *y* `Levenshtein`

similarTextEcuación=[((size.campo1+size.campo2)/2)-similarText(campo1, campo2)]
*similarPromedio= [(similarText(campo1, campo2)*100) / (size.campo1+size.campo2)]*

Como se mencionó anteriormente, el nivel de exactitud buscado durante la comparación para los campos citados en las tablas, es relativamente elevado. Se realiza una comparación rigurosa, por lo que un par determinado de campos será considerado éxito en el proceso de comparación, si y sólo si ambos valores, Similar_text Ecuación y Levenshtein son iguales a 0. Como se puede observar en la tabla, a medida que se acentúan las diferencias entre las cadenas de texto comparadas, los valores de la ecuación SimilarText y Levenshtein se alejan de cero.

Todos los campos involucrados en un análisis de detección de duplicado, serán previamente normalizados a través de una secuencia de filtros, con el fin de dejarlos libres de un conjunto determinado de caracteres. Estos pueden parecer insignificantes, pero resultan determinantes para el proceso de comparación de cadenas de texto. Algunos de los filtros que se aplican son:

- Convertir mayúsculas a minúsculas..
- Eliminar puntos, comas, acentos, guiones, comillas, paréntesis, entre otros.

Memoria caché

La memoria caché está formada por un conjunto de datos para su rápido acceso. Estos datos son una copia de otros originales, los cuales tienen un elevado costo de recuperación.

Cuando se realiza una recolección de recursos, se hace una copia del resultado obtenido en la caché; en caso de que el mismo u otro usuario de la aplicación repita dicha búsqueda, los accesos siguientes se realizan a dicha copia, haciendo que el tiempo de acceso medio al resultado sea menor. De este modo, y con el fin de reducir el tiempo de acceso por parte de los usuarios a la información brindada por los diferentes catálogos, se diseñó un mecanismo de caché, cuya función es brindar soporte a la aplicación, permitiéndole a la misma recuperar resultados de búsquedas ya realizadas.

Cada uno de los resultados, producto de una consulta previa sobre numerosos catálogos, se relaciona directamente con una búsqueda en particular. Esta asociación N:1 permite una rápida localización y efectiva recuperación, lo cual brinda la posibilidad de generar una respuesta ágil sin la necesidad de consultar uno a uno los servidores definidos para dicha acción. Los resultados se mantienen vigentes durante una determinado tiempo, configurable desde la administración, con el fin de mantener una integridad de la información brindada sin que se produzca una inconsistencia en la información. Para esto, de forma automática y periódica, se ejecuta un proceso encargado de determinar la validez de las búsquedas que fueron previamente realizadas. Este proceso toma como referencia la fecha en que se realizó cada búsqueda para determinar un validez.

El hecho de que una búsqueda registrada en la aplicación deje de estar vigente, es decir, no se tenga en cuenta al momento de determinar existencias de búsquedas duplicadas, no quiere decir que su resultado deje de estar disponible para otras operaciones o tareas, como podría ser la generación de estadísticas, o análisis del volumen y tipo de información retornada por un catálogo en un transcurso de tiempo determinado[15].

Librería

Con el objetivo de poder integrar la herramienta con otras aplicaciones y sistemas de terceros interesados en consumir este servicio y brindarlo a sus usuarios, se diseñó una librería que puede ser incorporada a dichos sistemas de manera muy simple. Esta librería abstrae a los desarrolladores de cuestiones técnicas propias del servicio (conexión, protocolos, procesamiento de resultados).

[15] Se especifica en el *Capítulo 5 - Conclusiones y Trabajos Futuros*

La librería dialoga con la aplicación centralizada a través de una API diseñada *ad-hoc*. Dicha API funciona sobre el protocolo HTTP, mediante el cual accede a URLs que devuelven datos. Para facilitar la solicitud de URLs por parte de un cliente y procesar las respuestas del servidor a esas solicitudes, se utiliza una librería adicional llamada cURL, la cual tiene una serie de funciones y procedimientos para acceder al contenido de URLs.

Cada uno de los métodos de comunicación brindados por la librería para la interacción con el servidor, fue creado sobre las bases de REST (transferencia de estado representacional), un término acuñado por Roy Fielding, que básicamente supone que cada URL identifica de manera única a un recurso en Internet. Las diferentes acciones realizadas sobre ese recurso no hacen uso de diferentes URLs , sino de diferentes métodos de solicitud HTTP.

Otro de los conceptos que se han tenido en cuenta al momento de diseñar la API fue el de la exclusión de usuarios no registrados en el sistema. Independientemente de que la librería se pueda obtener de forma libre y gratuita, la misma exige el uso de credenciales para restringir de este modo el acceso a la aplicación, y obliga así a los interesados del servicio a ponerse en contacto con la administración para obtener dichas credenciales.

Actualmente hay un total de 8 funciones definidas para interactuar con la aplicación[16]. La funcionalidad que actualmente se le brinda a un usuario es la siguiente:

- Creación dinámica y flexible del objeto utilizado para generar la consulta a cada uno de los servidores:
  Esta función hace uso de 5 métodos creados específicamente para la definición de cada uno de los campos permitidos en la búsqueda. Todos los campos, Título, Autor, Issn, Isbn, Año de Publicación, son encapsulados de este modo, en un único objeto, a partir del cual se genera la consulta.

- Realizar una petición de búsqueda síncrona:
  Mediante esta función se indica la necesidad de obtener los resultados de forma inmediata. Utiliza como parámetro el objeto mencionado anteriormente.

- Realizar una petición de búsqueda asíncrona:

[16] Se especifica en el apartado *Anexo 1 - Integración y uso de la API.*

A diferencia del caso síncrono, esta función realizará una consulta diferida. El resultado será un identificador único de petición, mediante el cual se podrá consultar en un futuro.

- Exigir la atención de un pedido asíncrono:
  Esta función exige la atención de un pedido que se encuentra en estado pendiente de atención, requiere el identificador unico de peticion como parametro.

- Consultar el estado en que se encuentra una petición asíncrona previamente realizada:
  A partir del identificador único de determinada petición, se puede preguntar a la aplicación, en que estado se encuentra dicho pedido, es decir, si ya fue atendido o no.

- Obtener todas las peticiones asíncronas que aún no fueron atendidas:
  Retorna todas las peticiones asíncronas pendientes, es decir, que aún no fueron atendidas, que tiene el usuario en un momento determinado.

- Obtener una actualización de los resultados relacionados a una consulta previamente ejecutada, con el fin de constatar la integridad de los datos con los que se cuenta:
  Esta función requiere como parámetro el identificador único de la petición, mediante el cual determina la consistencia del resultado relacionado a la misma, y de ser necesario, lo actualiza, permitiendo una posterior recuperación.

- Al momento de definir el tipo de resultado que queremos obtener, podemos optar por hacerlo en formato json o xml:
  Existen dos métodos encargados de definir el tipo de resultado que se desea obtener. Esta tarea resultan claves para el posterior procesamiento del resultado.

Capítulo 4

Entornos de aplicación y casos de prueba

4.1.Introducción

En este capítulo se explican cada uno de los casos de prueba que fueron considerados. Si bien son múltiples las posibilidades de aplicación, sólo dos fueron tenidas en cuenta, para cada una de las cuales se realizaron numerosos estudios y pruebas de funcionamiento.

Al momento de seleccionar los entornos de aplicación se buscó cubrir dos aspectos que se consideraron fundamentales para la determinación del éxito de la herramienta:

- Por un lado, se buscó probar el poder identificar y recolectar recursos a partir de la creación y posterior ejecución de una consulta de búsqueda sobre un múltiple número de repositorios y catálogos en línea (Aplicación Web).
- Por otro lado, se buscó probar el poder de integración que tenía la herramienta MetaLib Service para con aplicaciones de terceros (Celsius).

4.2 Aplicación Web

Con el fin de lograr un rápido y libre acceso a la herramienta, se diseñó una interfaz web, por medio de la cual se presenta un formulario encargado de recoger los datos ingresados por un usuario con la necesidad de localización de un recurso en particular. A través de los campos ingresados, se genera una consulta y posterior recolección de recursos a partir de cada uno de los catálogos en línea asociados a la herramienta. Cada uno de los registros obtenidos, serán presentados a través de la misma interfaz.

La aplicación Web está desarrollada con software de uso libre y es totalmente portable, brindando la posibilidad de ser utilizado bajo cualquier navegador web de cualquier sistema

operativo. Como la mayoría de las aplicaciones web, no requiere la instalación de ningún software adicional ni requiere configuración alguna por parte de los posibles usuarios.

Si bien la acción de búsqueda está respaldada por un mecanismo de detección de búsquedas duplicadas, mediante el cual será posible valerse de un rápido acceso a resultados relacionados a búsquedas ya realizadas, todas las peticiones realizadas a través de dicha interfaz web, son atendidas de forma síncrona, esto justifica la demora en la visualización de los registros obtenidos. Recordemos que el proceso de ejecución de una petición síncrona es relativamente lento (en comparación a una búsqueda en un buscador comercial), y el tiempo final de la búsqueda depende principalmente de los tiempos de búsqueda de los servicios de terceros (repositorios, catálogos en línea).

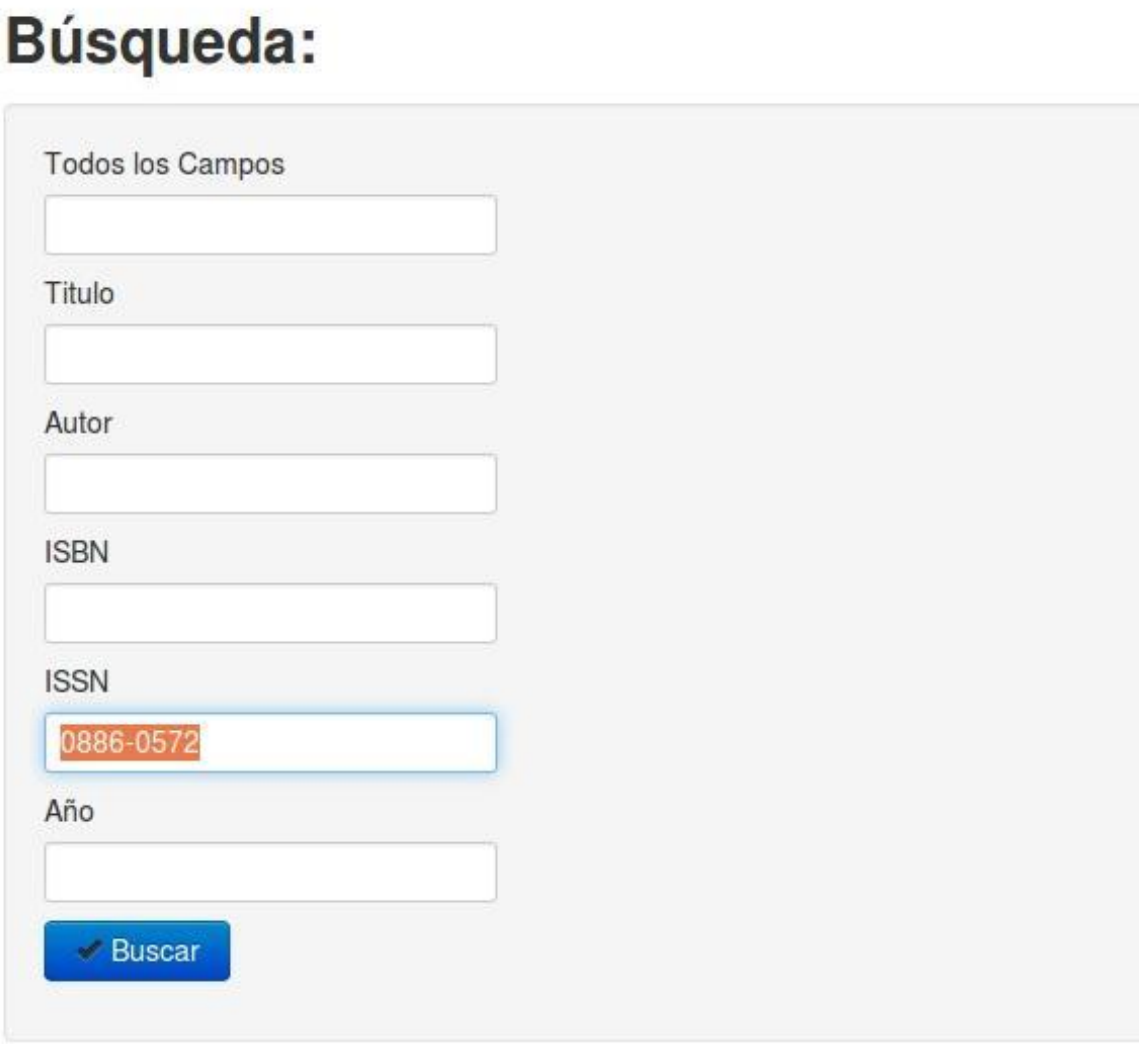


Figura 4.1 *Formulario de solicitud de pedido*

Performance de una búsqueda satisfecha de forma remota:

En el siguiente ejemplo podemos observar una recolección de recursos a partir de un ISSN ingresado por el usuario. La acción fue analizada por medio de una herramienta (*HTTPWatch*[17]) que permite obtener los detalles de cada una de las conexiones HTTP establecidas. De este modo se pudo determinar el tiempo que llevo el proceso de recolectar los recursos a partir de cada uno de los 27 catalogos involucrados en el proceso de búsqueda. La petición llevo un tiempo total de 2,2 minutos.

Figura 4.2 *Performance de una búsqueda satisfecha de forma remota.*

Performance de una búsqueda satisfecha de forma local:

[17] A través de HttpWatch se puede obtener información básica sobre las conexiones HTTP, como por ejemplo las direcciones URL, los códigos de estado, tamaño de la respuesta, el tipo MIME y el tiempo.

Figura 4.3 *Performance de una búsqueda satisfecha de forma local.*

El mismo ejemplo mencionado anteriormente fue ejecutado nuevamente con el fin de analizar la respuesta de la aplicación ante una petición de búsqueda para la cual ya se cuenta con el resultado en memoria local. La acción fue perfilada con la misma herramienta (`HTTPWatch`) y se pudo determinar que los tiempos de respuesta se redujeron considerablemente. La petición llevo un tiempo de 0.387 segundos.

Si bien el tiempo de atención de un pedido de búsqueda puede variar dependiendo de la cantidad de catálogos involucrados en proceso de recolección, así como la cantidad de recursos seleccionados de cada uno de estos, es evidente lo eficaz que resulta ser el mecanismo de atención de pedidos respaldado por un mecanismo de detección de búsquedas duplicadas

4.3 Celsius Network

El Proyecto de Enlace de Bibliotecas (PrEBi) de la Universidad Nacional de La Plata, servicio sobre el cual se integró la herramienta MetaLib Service, brinda un servicio de búsqueda y

provisión bibliográfica a docentes, investigadores y alumnos de la UNLP. Dicho servicio cuenta con más de 3200 usuarios locales, y se enmarca dentro de la iniciativa Library Linkage (LibLink) del Consorcio Iberoamericano para la Educación en Ciencia y Tecnología (ISTEC), del cual participan universidades, centros de investigación e instituciones de América y España. La Iniciativa LibLink tiene como objetivo compartir los acervos bibliográficos de todas las bibliotecas de sus instituciones participantes. Para realizarlo, cada institución recibe pedidos de bibliografía de sus usuarios locales, y los solicita a alguna de las bibliotecas previa búsqueda en cada uno de los catálogos de las mismas. Este proceso secuencial demora mucho tiempo, ya que los administradores deben ingresar uno por uno a cada catálogo de cada institución, y realizar una búsqueda allí. Luego, una vez que la solicitud bibliográfica es detectada en un catálogo, se le envía una solicitud a la biblioteca de dicha institución o directamente a la instancia de Celsius de la institución en caso que la tuvieran. Toda la gestión de usuarios, pedidos, registro de búsquedas, catálogos y bibliotecas participantes se realiza por medio del software Celsius Network, desarrollado en PrEBi UNLP y distribuido a todos los miembros de LibLink. Este software nació en el año 2001 como una herramienta de generación estadística, y evolucionó constantemente para integrar nuevas funciones, actualizar sus herramientas e incorporar herramientas de trabajo distribuido entre los participantes. Actualmente, Celsius se ha convertido en el soporte informático por defecto de la iniciativa LibLink.

Usuarios de Celsius

El software de Celsius cuenta con un creciente número de usuarios, los cuales cargan pedidos de búsqueda en el sistema, a través de un formulario de carga de pedido presentado en el sitio web. Entre los campos ofrecidos para el alta de un pedido, haremos especial hincapié en aquellos que serán considerados al momento de realizar la metabúsqueda, el resto de los campos, sólo existen para el funcionamiento interno del sistema. Estos campos son: Título de Revista o Libro, Autor y isbn_issn.

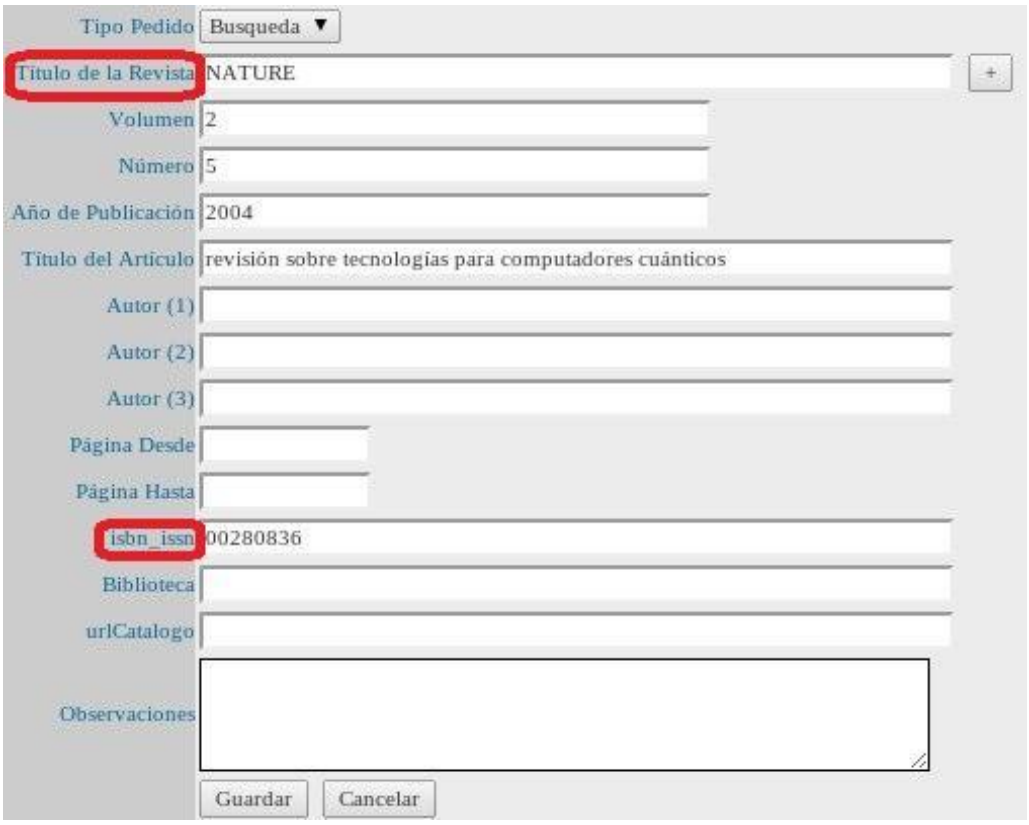


Figura 4.4 *Formulario de solicitud de pedido.*

Una vez completado y enviado el formulario con la información del pedido, en primer lugar, se cargará la solicitud en estado pendiente dentro de Celsius, y luego, de forma automática y sin la necesidad de intervención del administrador del sitio, se realizará la petición asíncrona de búsqueda de pedido a la aplicación MetaLib Service por medio de la API.

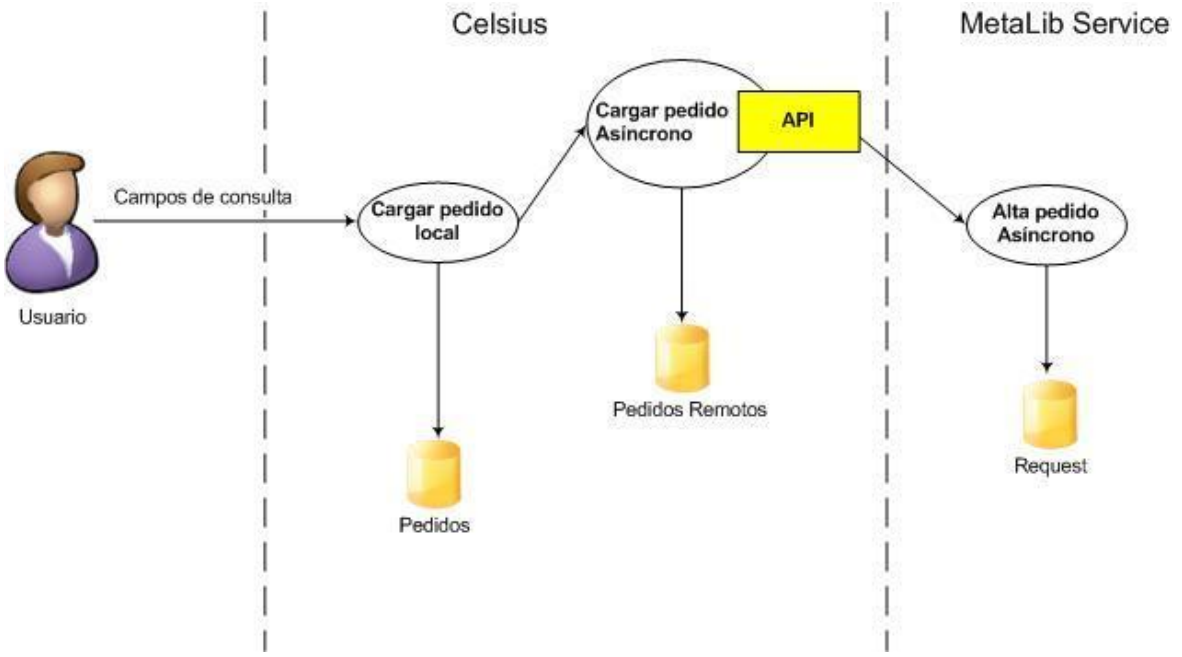


Figura 4.5 *Solicitud de búsqueda a partir de una aplicación de tercero (Celsius).*

Una vez que el usuario cargó el pedido en Celsius, el mismo queda a la espera de que algún administrador del sitio pueda satisfacer su petición de búsqueda.

Administradores de Celsius

El trabajo de los administradores, y particularmente la parte que a nosotros nos concierne, comienza a partir de la atención de un pedido en estado pendiente. En este punto, el pedido ya fue solicitado de forma automática y asíncrona a la aplicación MetaLib Service y se encuentra a la espera de ser atendido.

Si bien el lapso de tiempo que espera el cron (administrador de procesos ejecutados en segundo plano a intervalos regulares de Unix) para seleccionar y ejecutar una petición en estado pendiente es relativamente pequeño en comparación con el costoso procedimiento que implica la búsqueda de un pedido, puede darse el caso, que en el momento en el que el administrador de Celsius solicita los resultados de la metabúsqueda, los mismos aún no estén disponibles, debido a que dicha tarea aún no fue realizada. Por este motivo, existe la posibilidad de exigir la atención de un pedido en particular desde la administración de Celsius, provocando la recolección de recursos que satisfagan al pedido generado.

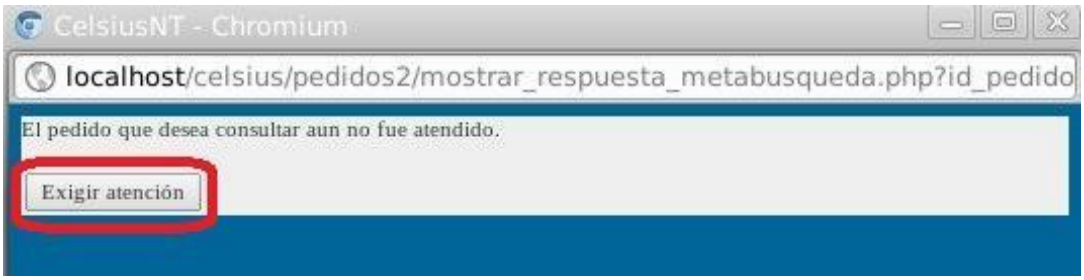


Figura 4.6 *Pedido de atención por parte de un administrador de Celsius.*

De este modo, un pedido que fue cargado desde Celsius de forma asíncrona en la aplicación MetaLib Service y se encuentra a la espera de ser atendido, puede comenzar el ciclo de ejecución por dos eventos distintos:

- La aplicación Celsius, a través de su administrador, exige la atención del pedido de forma explícita indicando el identificador del mismo.
- El proceso encargado de seleccionar y ejecutar pedidos pendientes de atención, lo tomó de entre la lista total de pedidos. Dicho proceso es ejecutado de forma automática y periódica por medio de cron.

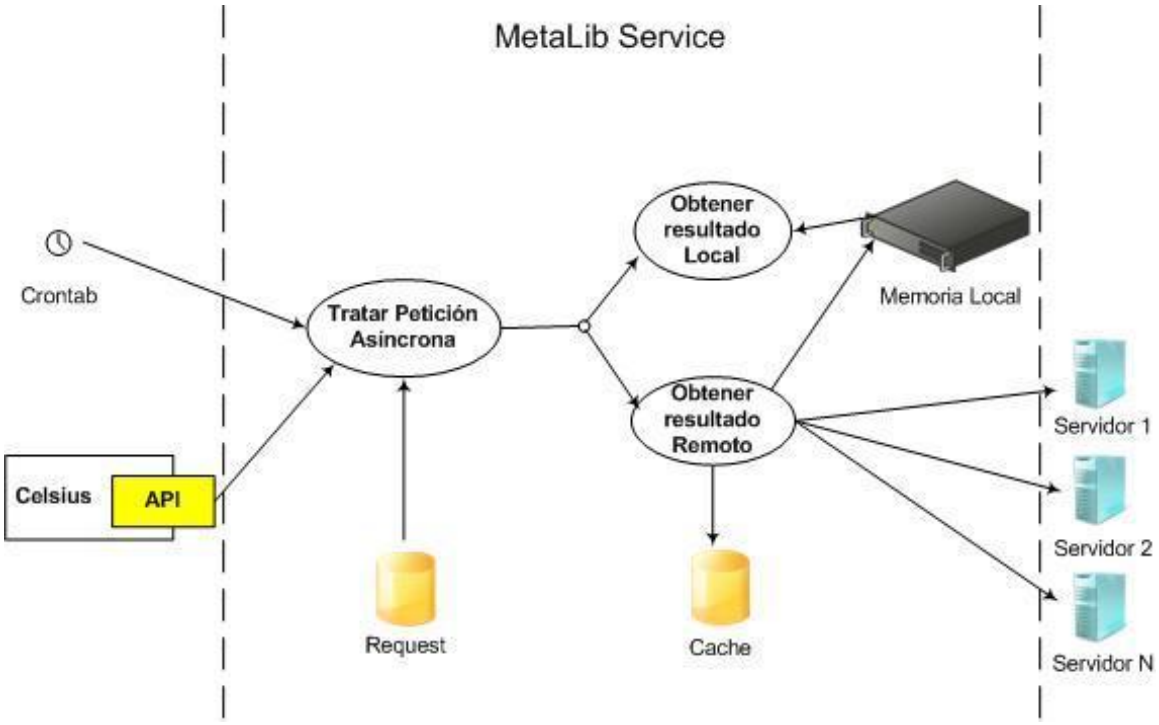


Figura 4.7 *Eventos causantes de la atención de un pedido en estado pendiente (asincro).*

Lo importante aquí es que, una vez ejecutada la metabúsqueda, el administrador de Celsius tendrá a su disposición y de forma automática una lista de resultados ofrecidos por la aplicación MetaLib Service.

La estructuración de la respuesta es tal, que le permitirá al administrador de Celsius determinar la existencia del recurso buscado en los catálogos involucrados en la búsqueda, o en el peor de los casos, descartar los mismos, evitando de este modo, la costosa tarea de consultar cada uno de forma individual.

Figura 4.8 *Metabúsqueda ejecutada por un administrador de Celsius.*

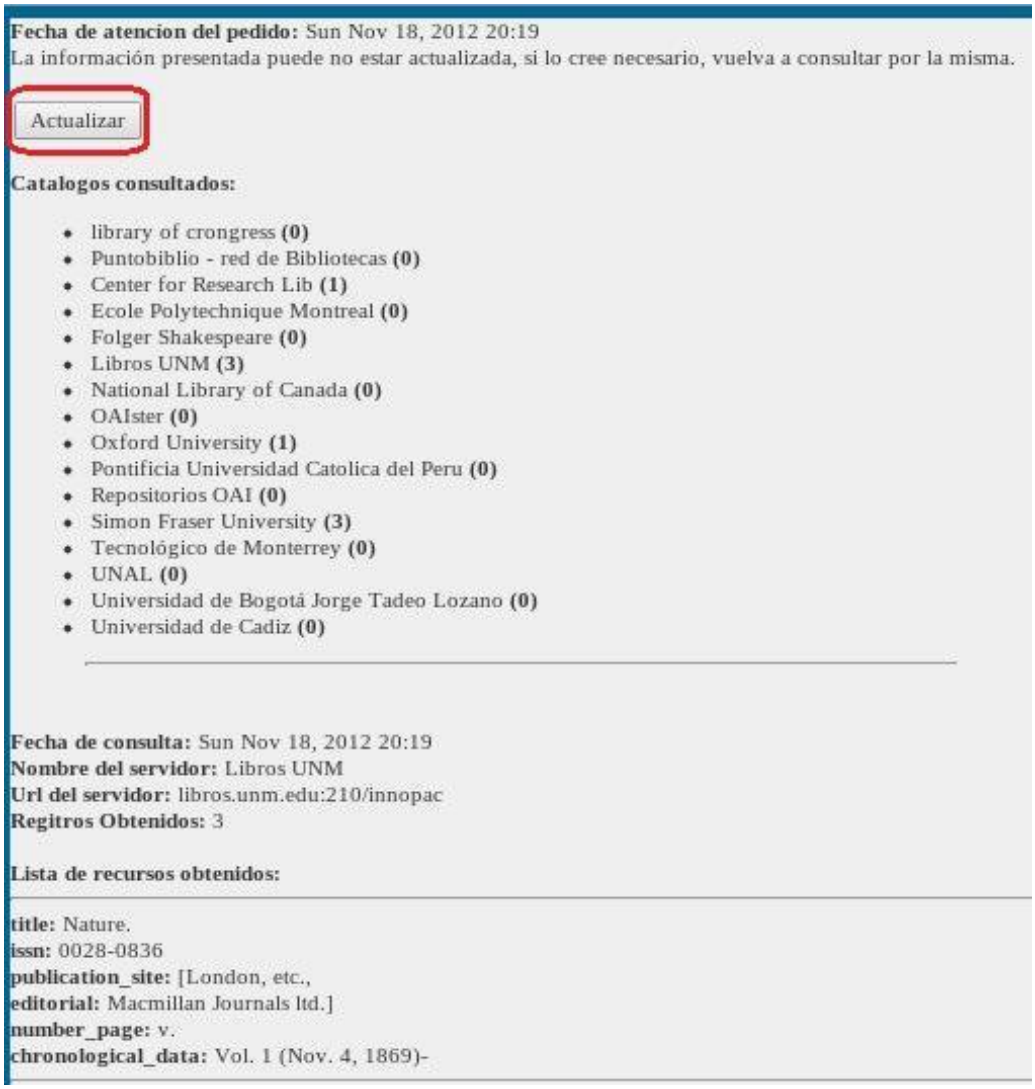


Figura 4.9 *Visualización de resultados producto de la metabúsqueda.*

Con el transcurso del tiempo y como producto de la inserción, modificación o eliminación de recursos en cada uno de los catálogos involucrados en el proceso de recolección de recursos, es posible que se pierda la validez de los datos que ya han sido recolectados. Si bien esta actividad escapa a los alcances de la herramienta, para contrarrestar dicho inconveniente se diseñó un mecanismo, que dependiendo de la fecha en la que se recolectaron los recursos, se determina la necesidad o no de actualizar dicho resultado. Esto permite que el administrador solicite la actualización de un resultado cuando el mismo pierda validez, manteniéndose así la consistencia de la información.

En este proceso de actualización de resultados, se permite que el administrador modifique los campos relacionados al pedido de búsqueda. Particularmente en esta situación, existe un problema relacionado a la normalización de datos, es decir, un único título de revista por ejemplo, puede ser mencionado de distintas maneras por diferentes personas. Si bien esto parece un detalle sin relevancia, al momento de realizar una búsqueda, la exactitud en estos tipos de campos resulta determinante para la correcta recolección de recursos. Es por esto, que un administrador del sitio Celsius podrá normalizar la información de la petición de búsqueda ingresada por el usuario, para rehacer la misma con los parámetros que considere correctos. Esta actividad de corrección de campos puede realizarse independientemente de si el resultado perdió o no la vigencia.

4.4 Conclusión

Los resultados obtenidos de cada uno de los casos de prueba de aplicación mencionados anteriormente, resultaron positivos y cumplieron con las expectativas planteadas.

La integración de la API resultó ser flexible y poco intrusiva para con las aplicaciones de terceros. Para el caso particular de Celsius, resultó de gran utilidad poder brindar al administrador un conjunto de resultados, sin la necesidad de la intervención del mismo en la costosa tarea de selección y recolección de los recursos. De este modo el administrador efectiviza en gran medida el proceso de atención de pedidos, ya que puede determinar de manera puntual cada una de las existencias del registro, o en el peor de los casos, descartar aquellos catálogos en los que no se encontró dicho recurso.

La aplicación web resultó de gran utilidad por el hecho de brindar un rápido y libre acceso al servicio. Este elevado grado de accesibilidad, es en gran medida producto de la independencia que posee la herramienta con respecto al hardware y software mediante el cual se realicen peticiones. La misma no presenta restricciones de uso, y posibilita una metabúsqueda de amplio espectro sobre varios catálogos disponibles en la web, ahorrando el tiempo y esfuerzo que llevaría la tarea de búsqueda en cada uno de los catálogos involucrados en el proceso de recolección.

Capítulo 5

Conclusiones y Trabajos Futuros

5.1.Conclusión

Es notable el crecimiento de recursos bibliográficos producido en Internet durante la última década. La gran dispersión generada entre éstos, y la poca relevancia al momento de organizar y estructurar los mismos, dificultó de manera significante la labor de aquellas personas interesadas en localizar dichos recursos. Si bien son evidentes los aportes brindados por las múltiples bibliotecas digitales y los repositorios institucionales disponibles en la web, se creyó conveniente generar una abstracción del distanciamiento entre cada instancia, permitiendo englobar múltiples catálogos a partir de una única búsqueda. Por este motivo, resultó necesario crear la herramienta MetaLib Service con el objetivo de agilizar los servicios de búsqueda bibliográfica y facilitar la labor de referencistas y bibliotecarios encargados de atender la creciente demanda de investigadores, docentes y alumnos de su institución.

A partir de las pruebas realizadas sobre la herramienta, podemos concluir que se logró efectivizar en gran medida el proceso de búsqueda y recolección de recursos específicos. Las diferentes pruebas que han sido realizadas a lo largo del desarrollo fueron exitosas, tanto en la aplicación Celsius como en la aplicación Web. La integración de la API resultó ser flexible y poco intrusiva para con la aplicación de terceros. Es evidente la gran utilidad brindada a los referencistas y bibliotecarios, el poder contar con un conjunto de resultados, sin la necesidad de la intervención de los mismos en la costosa tarea de búsqueda y selección de los recursos a través de los múltiples repositorios distribuidos en la web.

De esta forma se cumple uno de los principales objetivos de los repositorios y bibliotecas digitales, difundir no solo la producción científica generada por las instituciones académicas, sino también los recursos bibliográficos de diferente índole.

5.2 Trabajos futuros

Si bien la herramienta MetaLib Service respondió como se esperaba a cada una de las pruebas a las que se sometió, se debe ser cauto y esperar las posibles observaciones que puedan realizar tanto los referencistas de los sitios en donde se integró el servicio, como los usuarios de la aplicación web. Adicionalmente, el uso masivo y exhaustivo de esta herramienta generará requerimientos de cambios y/o mejoras que excedan las posibles previsiones realizadas al momento, y que como es de esperar, serán consideradas por los autores de este trabajo a fin de poder mejorar aún más a MetaLib Service.

A continuación, se detallan cada uno de los aspectos que serán considerados en un futuro en el proceso de optimización de la herramienta.

Interfaz Web de Búsqueda

En una primera etapa, se buscará optimizar la interfaz web de la herramienta para brindar una mejor visualización de los listados de pedidos obtenidos a través de la misma. Se sacará provecho a la estructura jerárquica de los registros involucrados en la respuesta, de modo tal de poder realizar una visualización de la información detallada y discriminada por catálogo, del mismo modo que se hizo la visualización de resultados en la integración con la aplicación Celsius.

Como se sabe que el proceso de búsqueda y recolección de recursos es costoso, por tal motivo, cada vez que se ejecuten acciones de búsqueda a partir de la interfaz web, se bloqueará la pantalla para evitar que el usuario siga interactuando con la herramienta mientras se recolectan los recursos.

Generación de Estadísticas

Posteriormente, se prevé agregar nuevas funciones encargadas de procesar y analizar cada una de las operaciones previamente persistidas en el sistema, con el fin de lograr la generación de múltiples estadísticas, ya sea tanto sobre el tipo y número de pedidos realizados a la aplicación, así como la cantidad y tipo de información que retorno cada catálogo para cada uno de estos pedidos. Esto permitirá conocer y estudiar la evolución de los distintos catálogos y repositorios en cuanto a los documentos que disponen y proveen a sus usuarios, y también

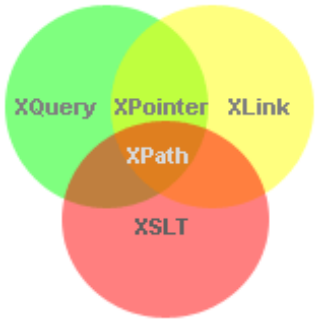
permitirá analizar cómo evolucionan las solicitudes de metabúsqueda de los usuarios de esta herramienta, ya sean usuarios finales a través de la interfaz web de búsqueda, o aplicaciones de terceros que se comunican por medio de la API. Todos estos procesos son posibles debido a que la herramienta lleva un historial de todas las transacciones realizadas, así como también de los resultados obtenidos para las mismas.

Paginación de Resultados

Un punto importante a desarrollar en un futuro, es permitir la paginación de los resultados recolectados por la metabúsqueda. El hecho de contar con respuestas de gran volumen en cuanto a la cantidad de registros retornados por cada uno de los catálogos y, si se considera además la posibilidad de que la cantidad de los servidores consultados se irá incrementando con el tiempo, es evidente la necesidad de idear un mecanismo que permita fraccionar el resultado, de modo tal que el mismo pueda ser entregado en su totalidad, pero bajo demanda del cliente. Lo mencionado anteriormente sumado al elevado consumo de recursos que la actividad de recolección de registros requiere, significó una importante limitación para el desarrollo de la herramienta, motivo por el cual, se decidió implementar una solución temporal al inconveniente, mediante la cual se acotó la cantidad de resultados recolectados de cada catálogo. Pero esto pone claramente un freno a las posibilidades que la herramienta brinda.

Luego de analizar varias posibles soluciones, se determinó que lo más adecuado y eficiente, será incorporar una base de datos xml con el objetivo de almacenar de forma completa los resultados obtenidos de cada recolección de recursos. Dicha base de datos podrá ser alguna de código abierto como dbXML, eXist, Xindice. Dado que estas bases de datos pueden gestionar millones de registros XML sin mayores inconvenientes, resultan una solución ideal para mantener los registros históricos (que pueden ser muy aprovechados por el módulo de generación de estadísticas arriba mencionado), y para procesar los datos que retornan los distintos servidores antes de enviarlos al usuario o aplicación final.

Además, estas bases de datos permiten utilizar el lenguaje XPath, mediante el cual es posible construir expresiones para recorrer y procesar un documento XML de manera simple y muy eficiente. Si bien el lenguaje no fue creado realmente para realizar búsquedas en bases de datos, se presenta como una gran opción al momento de realizar búsquedas en un documento determinado.



[18]

Figura 5.1 *Lenguajes de Consulta XML.*

La potencia de consulta y recuperación que brinda el lenguaje, sumado a la organización jerárquica mediante la cual se estructuran los resultados, nos permitirá discriminar recursos particulares por cada uno de los catálogos involucrados en la respuesta. De este modo, será posible para el usuario que ejecuta la búsqueda, solicitar para cada uno de los catálogos que componen la respuesta, un subconjunto determinado de recursos. Por ejemplo en una primera solicitud, se podrá solicitar el subrango (x; p) de recursos, luego los siguientes (x1; p1) y así sucesivamente hasta agotarse los mismos. El subrango de recursos (x; p) será flexible entre cada una de las solicitudes.

El usuario solicita los registros que se encuentran dentro del rango (1,4)

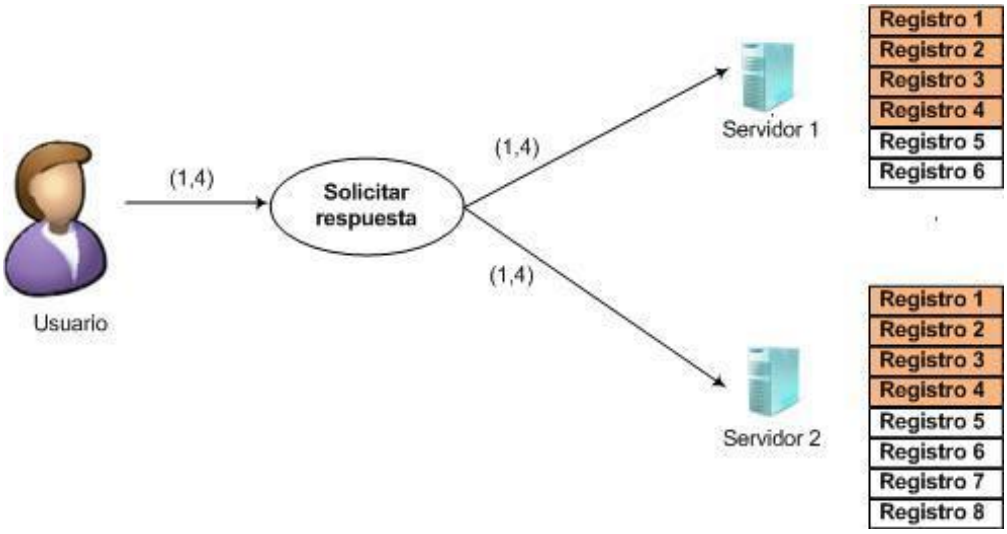


Figura 5.2 *Primer* solicitud de resultados XML

[18] Imagen tomada de: http://www.w3schools.com/xpath/xpath_intro.asp

El usuario realiza una segunda solicitud sobre el mismo resultado, pero con un rango diferente (5,7)

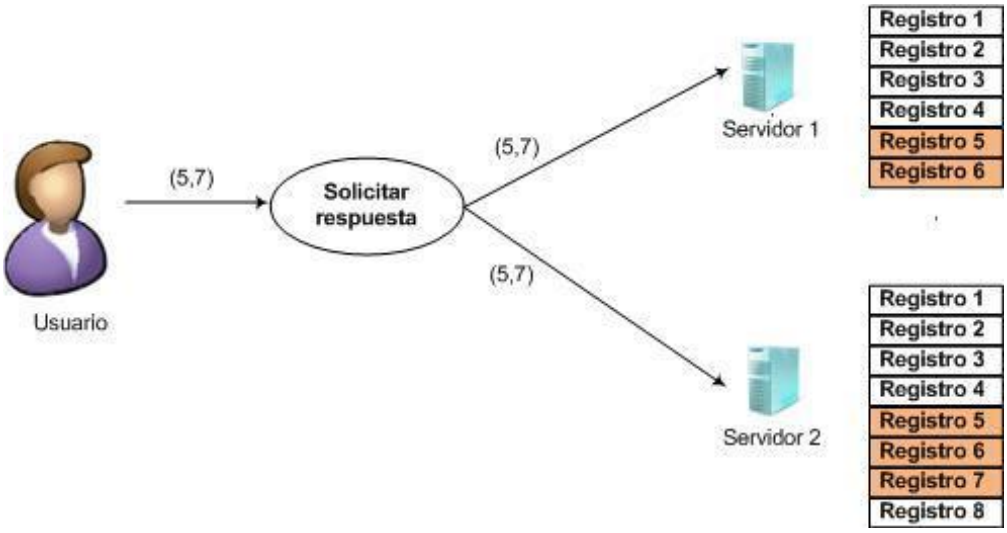


Figura 5.4 *Segunda* solicitud de resultados XML.

Es importante notar que para una determinada solicitud, el subrango o cantidad de recursos obtenidos, se mantiene igual para cada uno de los servidores. A continuación y mediante un ejemplo, se mostrará un conjunto reducido de operaciones brindadas por XPath, sobre un resultado tipo.

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
  <search_date>Sun Nov 18, 2012 20:19</search_date>
  <servers>
    <entry>
        <name>Center for Research Lib</name>
        <url>catalog.crl.edu:210/INNOPAC</url>
        <records>
           <entry>
              <title>Nature</title>
              <isbn>0028-0836</isbn>
              <publication_site>Washington</publication_site>
              <editorial>Macmillan Journals Ltd.</editorial>
              <number_page>197</number_page>
           </entry>
          <entry>
           . . .
           . . .
```

```
            . . .
          </entry>
        </records>
      </entry>
     <entry>
     . . .
     . . .
     . . .
     </entry>
  </servers>
</result>
```

Dada una respuesta XML, es posible seleccionar todos los catálogos encontrados en el archivo respuesta a partir de la siguiente expresión:

```
result/servers//entry
```

Entonces, es posible iterar el contenido de cada uno de los catálogos de la siguiente forma:

```
for $catalog in result/servers//entry
```

Y a partir del subrango (1; 50), podremos recuperar un subconjunto de registros pertenecientes a cada uno de los catálogos.

```
for $catalog in result/servers//entry
      $catalog/records//entry[(position()>1) and (position()<50)]
```

La idea de esto no es hacer una demostración detallada del funcionamiento y las operaciones brindadas por XPath, sino más bien, demostrar con qué facilidad se puede procesar un resultado XML. Obviamente esta tarea requerirá un proceso de estudio y análisis en profundidad, tanto de las herramientas nombradas anteriormente, así como de las prestaciones que ofrecen las mismas.

Anexo 1

Integración y uso de la API

1.1. Introducción

A lo largo de éste anexo se explica detalladamente el proceso de instalación y el uso correcto de la API que acompaña a MetaLib Service. El servicio brindado por la API permitirá integrar a MetaLib Service a una aplicación específica, con el objetivo de consultar catálogos de forma remota y, de ésta manera obtener registros de manera sencilla y transparente .

1.2. Requisitos

Para la integración correcta de la API a una aplicación, se deberá cumplir con los siguientes requisitos y/o necesidades:

- Identificador Único (API KEY).
- Version PHP 5 en adelante
- Libreria cURL

1.3. Instalacion

El proceso de instalación, está compuesto por dos etapas básicas:

- Obtención del Identificador Único (API KEY).
- Incorporación de los fuentes al proyecto.

Identificador Único (API KEY)

Como primer paso para comenzar a utilizar la librería, deberá adquirirse por medio del sitio oficial la clave de identificación (identificationKey, comunmente conocida como API KEY), mediante la cual podrá identificarse como usuario registrado en el sistema en cada una de las transacciones realizadas.

Una vez que cuente con el identificador mencionado anteriormente, deberá editar el archivo `path_de_descarga/IntegrationLibrary/Constant.php`, seteando el valor de la constante `'_CONSUMER_KEY'` con el valor brindado por el administrador del sitio.

```
/**Identificador único de usuario**/
define('_CONSUMER_KEY', 'API KEY');
```

Incorporación de los fuentes

Para comenzar a interactuar con el servidor, será necesario incorporar la librería al proyecto que consumirá el servicio de metabúsqueda, siendo posible a partir de este momento, instanciar cada una de las acciones existentes.

```
/**Incluimos la librería dentro de la aplicación**/
require('/path/ObjectConnection.php');
```

1.4. Uso

Para lograr el uso correcto de la API, se deberá seguir detalladamente cada una de las siguientes etapas:

- Creación de la Conexión.
- Creación y ejecución de un pedido de búsqueda.
- Estado de la respuesta.
- Acciones sobre uno o varios pedidos.

Creación de la Conexión

Luego de setear el API KEY e incorporar la librería al proyecto que consumirá el servicio, se está en condiciones de crear el objeto a partir del cual se definirán características particulares de la conexión y cada una de las transacciones.

```
/**Instanciamos la Clase para gestionar las conexiones y peticiones al servidor**/
$http = new ObjectConnection();
```

La API funciona mediante el protocolo HTTP. Además, se utiliza una librería adicional llamada cURL, la cual tiene una serie de funciones y procedimientos para acceder al contenido de URLs.

```
/**Inicializa el objeto curl con las opciones por defecto**/
$http->init();
```

Creación y ejecución de un pedido de búsqueda

La creación y ejecución de un pedido de búsqueda esta compuesto por varias acciones de menor magnitud, a continuación identificaremos y explicaremos cada una de estas.

Tipo de acción de consulta

Como primer paso será necesario definir el tipo de acción mediante el cual queremos realizar la búsqueda. Las posibles alternativas son síncrona y asíncrona, en caso de no indicarse ningún tipo, por defecto, la búsqueda será de tipo síncrona.

Síncrona

Al utilizar este tipo de acción, los resultados se obtendrán de forma inmediata.

```
$http->setSyncroTypeAction();
```

Asíncrona

En este caso, se cargará el pedido en estado pendiente. Este método retorna el identificador del pedido creado, de modo tal que el usuario no pierda la referencia al mismo, permitiéndole posteriores acciones sobre dicho pedido.

```
$http->setAsyncroTypeAction();
```

Tipo de Resultado

Es necesario indicar el tipo de datos en el cual se desea obtener el resultado. Se puede optar tanto por el formato XML como por el formato JSON, en caso de no indicarse ningún tipo, por defecto, el tipo será JSON.

Para indicar esto contamos con dos tipos de métodos:

XML

```
$http->setXMLTypeResult();
```

Ejemplo resultado en formato XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
   <search_date>Sun Nov 18, 2012 20:19</search_date>
   <servers>
      <entry>
           <name>Center for Research Lib</name>
           <url>catalog.crl.edu:210/INNOPAC</url>
           <records>
              <entry>
                 <title>Nature.</title>
                 <issn>0028-0836</issn>
                 <publication_site>Washington></publication_site>
                 <editorial>Macmillan Journals Ltd.</editorial>
                 <number_page>--</number_page>
                 <chronological_data>Began in Nov 1869</chronological_data>
              </entry>
           </records>
      </entry>
   </servers>
</result>
```

JSON

```
$http->setJSONTypeResult();
```

Ejemplo resultado en formato JSON:

```
{"search_date":"Sun Nov 18, 2012 20:19",
"servers":[{
      "name":"Center for Research Lib",
      "url":"catalog.crl.edu:210\/INNOPAC",
      "records":[{
            "title": "Nature.",
            "issn": "0028-0836",
```

```
            "publication_site": "Washington :",
            "editorial":"Macmillan Journals Ltd.,", "number_page":"v.:",
            "chronological_data":"Began in Nov 1869."
            }]
        }]
}
```

En ambos casos el ejemplo es acotado, pero se podrían multiplicar tanto la cantidad de registros obtenidos por cada servidor, así como también el número de servidores involucrados en la dicha respuesta.

Creación de la consulta

Para definir cada uno de los valores de los campos que se utilizarán para crear la consulta que será ejecutada sobre los catálogos, será necesario crear una instancia del objeto encargado de almacenar estos valores.

Esta acción se llevará a cabo por medio del siguiente método:

```
        /**Instancia de la Clase ObjectQuery**/
        $query = new ObjectQuery();
```

A partir de aquí, se podra de setear tantos campos como sean necesarios y permitidos por la herramienta. Cabe destacar que los campos permitidos son: título, autor, año de publicación, issn, isbn y el campo comodín, todos estos sin excepción alguna, deberán ser de tipo string.

```
/**Se agrega un título**/
        $query->addTitle('php');

/**Se agrega un autor**/
        $query->addAuthor('garcia marquez');

/**Se agrega un año de publicación**/
        $query->addYear('1998');

/**Se agrega un isbn**/
        $query->addIsbn();
```

```
/**Se agrega un issn**/
       $query->addIssn('0886-0572');

/**Se agrega un campo comodin**/
       $query->addAllFields('php');
```

El caso de autor, puede ser tomado como un caso particular, debido a que es posible agregar múltiples valores del mismo para un único objeto consulta:

```
/**Se agrega un Author**/
       $query->addAuthor('antonio puig');
       $query->addAuthor('David lopez');
```

Ejecución de una búsqueda

Finalmente se podrá realizar la ejecución de la búsqueda y obtener el resultado:

```
       $response = $http->executeSearchPost($query);
```

La respuesta obtenida dependerá del tipo de acción elegido para ejecutar la consulta, en caso de tratarse de una petición síncrona, obtendremos directamente el resultado producto de la búsqueda a través de los catálogos, para una petición asíncrona, lo que obtendremos será el identificador del pedido creado.

Estado de la respuesta

Una vez ejecutada la petición de búsqueda, la variable `$response` contendrá el resultado producto de la ejecución de la búsqueda. Para obtener los datos concretos de la misma, será necesario realizar la llamada al siguiente método:

```
       $response->getResult();
```

La forma en que se debe iterar el resultado una vez obtenido la respuesta, dependerá del tipo de resultado elegido previamente.

Consulta de error

A través de la respuesta, será posible determinar si ha ocurrido algún error o excepción durante el proceso de búsqueda. Para esto se diseñó un objeto encargado de encapsular dicho evento

anómalo, el cual permitirá al usuario conocer e identificar el mismo. Dicha acción es posible debido a que la aplicación cuenta con un estándar de errores o excepciones en el cual se determinan múltiples pares (código, valor), encargados de describir cada una de las situaciones anómalas que puedan ocurrir. De este modo es posible, a partir del valor devuelto por el método de consulta de estado de la respuesta, conocer si una búsqueda se realizó de forma exitosa, true, o caso contrario, ocurrió algún evento que impidió el fin de su realización, false.

```
/**Si no ocurre ningun evento anomalo**/
if( $response->getState() )
{
       /**Obtengo los datos**/
       $response->getResult();
}
/**Si ocurre algun evento anomalo**/
else
{
    /**Obtengo el error**/
   echo 'Codigo: '.$response->getExceptionCode();
   echo '<br>';
   echo $response->getHumanReadableException();
   echo '<br>';
}
```

En caso de existir un evento que impida la normal ejecución, será posible consultar el codigo y descripcion del mismo.

Estándar de errores definidos

Código de Excepción	Detalle
4000	El tipo de acción elegido, no es permitido para su perfil.
4001	La identificación de usuario es inexistente.
4003	El pedido aun no fue atendido.
4004	No existen pedidos pendientes.
4005	La identificación de pedido no existe o no es correcta.
4007	El pedido se actualizo correctamente
4008	El pedido aún se encuentra vigente.
4010	Se produjo un error al consultar el catálogo.
4011	El proceso de creación de la respuesta fallo.

Anexo 1.1 *Códigos de excepción producidos por la Herramienta*

Acciones sobre uno o varios pedidos

La aplicación permite realizar un conjunto determinado de operaciones sobre uno o varios pedidos ya realizados por el usuario. Desde la API se podrá realizar la invocación a dichas operaciones por medio de las siguientes funciones:

- Conocer el estado de un pedido particular: en este método, se deberá indicar el identificador del pedido en el cual se tiene interés. El resultado dependerá de si el pedido fue o no atendido. Pudiendo obtener entonces, tanto la respuesta efectiva, así como también el mensaje relacionado al código 4003, indicando que el pedido aún no fue atendido.

```
$response = $http->getPendingRequestSearch(identificador);
```

- Obtener todos los pedidos pendientes que tiene un usuario.

```
$response = $http->getAllPendingRequestSearch();
```

- Exigir la atención de un pedido asíncrono: se podrá exigir la atención de un pedido asíncrono que aún no fue atendido, es decir, que se encuentra en estado pendiente. Esto provoca que se genere la inmediata atención del pedido, y posibilita de este modo la posterior recuperación del resultado:

```
/**Exijo atención de una búsqueda**/
$response = $http->demandAtentionRequestSearch(identificador);

/**Recupero resultado de la búsqueda**/
$response = $http->getPendingRequestSearch(identificador);
```

- Al pasar cierto tiempo desde la creación de un pedido, debido a las actualizaciones que sufren cada uno de los catálogos en línea involucrados en la generación de la respuesta, la misma pierde cierta validez. A través de este método, se podrá solicitar la actualización de un resultado obtenido a partir de una búsqueda realizada previamente. Para esto, se deberá indicar el identificador del pedido, así como el query del mismo. Esto brinda la posibilidad de modificar ciertos parámetros para la nueva búsqueda. Obtendremos de forma inmediata el resultado efectivo de la búsqueda.

```
$response = $http->updateResponseRequest($identificar, $query);
```

Referencias

[1] Weibel, S., Kunze, J., & Lagoze, C. (1997, February 9) “Dublin Core Metadata for simple resource description”

[2] Manuel Ortega Cantero, José Bravo Rodríguez. “Sistemas de interacción Persona-Computador”

[3] Crescencio Bravo Santos, Miguel Ángel Redondo Duque. “Sistemas interactivos y colaborativos en la web”

[4] AKEROYD, J. La Administración del Cambio en las Bibliotecas Electrónicas.

[5] *Lic. Marlery Sánchez Díaz y Dr. Juan Carlos Vega Valdés.* Bibliotecas electrónicas, digitales y virtuales: tres entidades por definir. Acimed Vol 10 06 2002

[6] Bernal Campos, C.A., & Bernal, E. SRU-SRW como estándar para buscar y recuperar información en ambientes URL y de Servicios Web, 2007. [Preprint].

[7] Aldo Guajardo Salinas. “Z39.50 y OAI-PMH: Protocolos de Transferencia y Recuperación de Información”. XV CONFERENCIA INTERNACIONAL DE BIBLIOTECOLOGIA

[8] BARRUECO, J. M.; SUBIRATS COLL, I. OAI-PMH: protocolo para la transmisión de contenidos en internet. El profesional de la información, 2003, marzo-abril, v. 12, n. 2, pp. 99-106.

[9] Meredith Ringel Morris, Jaime Teevan, Steve Bush. (2008) Enhancing collaborative web search with personalization: groupization, smart splitting, and group hit-highlighting. En ACM 2008 conference on Computer supported cooperative work. San Diego, CA, USA. P. 481-484.

[10] Lynch, C. (2003). Institutional repositories: Essential infrastructure for scholarship in the digital age. Technical Report 226.

[11] Smiraglia, R. P. (1992). Authority control and the extent of derivative bibliographic relationships. Tesis, Doctor of Philosophy, Faculty of the Graduate Library School, University of Chicago.

[11] Cutter, Charles A. (1876). Rules for a printed dictionary catalog. Washington: Government Printing Office.

[10] De Giusti Marisa R., Marmonti Emiliano, Lira Ariel J., Sobrado Ariel, Villarreal Gonzalo L. (2006) Interconexión entre portales de referencia a través de WebServices. En 4to Simposio Internacional de Bibliotecas Digitales, Málaga, España.

[11] De Giusti, Marisa Raquel; Villarreal, Gonzalo; Lira, Ariel Jorge; Sobrado, Ariel. Características CSCW en Celsius Network. III Conferencia Internacional de Biblioteca Digital y Educación a Distancia.

[12] Barry, Damián; Buckle, Carlos; Tinetti, Fernando Gustavo; Jaramillo, Carlos; Samec, Gustavo; Pacheco, Cristian; Real, Ignacio; Aita, Ignacio; Cortez, Juan Manue. "Técnicas de recuperación de información en grandes volúmenes de datos heterogéneos con bases de datos NOSQL"

[13] Gil Costa, Graciela Verónica; Printista, Alicia Marcela. (2008) Motores de búsquedas síncronos/asíncronos. X Workshop de Investigadores en Ciencias de la Computación.

[14] De Giusti, Marisa Raquel; Marmonti, Emiliano Horacio; Vila, María Marta; Sobrado, Ariel; Villarreal, Gonzalo. Tecnologías para propagar los contenidos de una Biblioteca Digital. IV Simposio Internacional de Bibliotecas Digitales, Málaga, España.

[15] César Augusto Bernal C.; Eloisa Bernal. SRU-SRW COMO ESTANDAR PARA BUSCAR Y RECUPERAR INFORMACIÓN EN AMBIENTES URL Y DE SERVICIOS WEB

[16] Ibai Sistemas.(marzo, 2012) Informe de adaptación de DSpace a Europeana. Fase Danubio: Europeana Data Model (EDM).

[17] YAZ [http://www.indexdata.com/yaz]. Accedido en Febrero 2012.

[18] YAZ [http://www.php.net/manual/es/book.yaz.php]. Accedido en Marzo 2012.

[19] MARC [http://www.loc.gov/marc/]. Accedido en Marzo 2012.

[20] Dublin Core [http://www.dublincore.org/]. Accedido en Mayo 2012.

[21] cURL [http://www.php.net/manual/es/book.curl.php]. Accedido en Agosto 2012.

[22] OpenSearch [http://www.opensearch.org/Home]. Accedido en Septiembre 2012.

[23] DSpace [http://www.dspace.org/]. Accedido en Septiembre 2012.

[24] SOAP [http://www.w3schools.com/soap/soap_intro.asp]. Accedido en Septiembre 2012.

[25] OAI-PMH [http://www.openarchives.org/pmh/]. Accedido en Junio 2012.

[26] Solr [http://lucene.apache.org/solr/]. Accedido en Mayo 2012.

[27] Blog SeDiCi [http://sedici.unlp.edu.ar/blog/2011/08/05/indexadores-de-texto-los-nuevos-motores-para-repositorios-digitales/]. Accedido en Octubre 2012.

[28] APIs [http://www.linuxjournal.com/content/apis?page=0,0]. Accedido en Octubre 2012.

[29] XPath [http://www.w3schools.com/xpath/xpath_intro.asp]. Accedido en Septiembre 2012.

[30] CLL [http://www.indexdata.com/yaz/doc/tools.html#CCL]. Accedido en Junio 2012.

[31] Z39.50 [http://www.loc.gov/z3950/]. Accedido en Mayo 2012.

[32] Función Levenshtein [http://www.php.net/manual/es/function.levenshtein.php] Accedido en Octubre 2012.

[33] Función Similar-text [http://www.php.net/manual/es/function.similar-text.php] Accedido en Octubre 2012.