Trabajo de diploma para optar por el título:

Ingeniero en Ciencias Informáticas

Módulo para la prestación de servicios según el estándar OAI-PMH en el SIGB.

Autor: Leonardo Alvarez Figueras

Tutores: Ing. Edisnel Carrazana Castro
Ing. Rolando Gual Pérez

Universidad de las Ciencias informáticas (UCI)
La Habana, 2013

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Leonardo Alvarez Figueras

_____

Firma del Autor.

Ing. Edisnel Carrazana Castro

_____

Firma del Tutor.

Ing. Rolando Gual Perez

_____

Firma del Tutor.

Agradecer a mis padres por siempre darme todo el apoyo y cariño del mundo, por estar presentes cuando lo he necesitado, por guiarme y alentar mis sueños y esperanzas.

A mi abuelo Fausto ya fallecido que fue como un padre para mí y a mi abuela Melba, que ha sido una madre para mí, no tengo como agradecerles por todo el amor y cariño que me han dado, los quiero.

A mis hermanos Carlos, Jeovani y en especial a mi hermano Alejandro, gracias por darme la dicha de contar con tu afecto y cariño.

Quiero también agradecer a mis tíos Pucho y Eduardo dos personas que he admirado toda mi vida, a mis tías abuelas Kira y Maguie,

A mis primos hermanos Karina, Ivan, Reinier, Frank Ernesto, Eidita y en especial a mi prima Daily.

También agradecer a Arturo, Ivonne y Arturito por ser como una segunda familia para mí, gracias por su apoyo incondicional.

A todo el equipo de trabajo que desarrolla el SIGB, en la UCI, en especial a los profesores Edisnel y Adnier, ejemplos de lo que debe ser un buen profesional.

Y a todos los amigos que he tenido el gusto de conocer en esta escuela, sin ustedes nada de esto hubiera sido posible, estarán en mi corazón para siempre.

Le dedico este trabajo a lo más grande que tengo en mi vida, a mi madre y mi padre, a ustedes les debo todo lo que soy hoy y a mis abuelos Melba y Fausto por todo el amor y cariño que me han dado.

Resumen

El presente trabajo describe las características referentes a una solución que permite al Sistema Integrado de Gestión Bibliotecaria (SIGB) de la biblioteca de la Universidad de las Ciencias Informáticas (UCI) incrementar su interoperabilidad. Debido al desarrollo actual de las tecnologías es indispensable que los sistemas de bibliotecas puedan comunicarse con sistemas afines para garantizar el acceso libre a la información. En el trabajo se analizan algunas de las iniciativas surgidas con el propósito de permitir la interoperabilidad entre los sistemas de información la cual permitirá la catalogación por copia y se realiza un estudio del Protocolo para la Recolección de Metadatos de la Iniciativa de Archivos Abiertos (OAI-PMH), señalando sus principales características y deficiencias y se selecciona un protocolo para su posterior implementación en el SIGB.

Palabras claves: interoperabilidad, metadatos, recolección, protocolo, SIGB.

Índice de contenidos

Índice de ilustraciones

Índice de tablas

Introducción

A lo largo de toda la historia de la humanidad el hombre siempre ha tenido la necesidad de conservar sus conocimientos y experiencias acumuladas para transmitirlas a las nuevas generaciones. Con el surgimiento de las primeras escrituras el hombre comenzó a plasmar sus ideas y pensamientos de manera física los cuales comenzó a almacenar o coleccionar en un determinado espacio para poder consultarlo cuando lo necesitara, lugar este que se conoce hoy como biblioteca.

Una biblioteca se puede describir como “Organización o parte de ella cuya principal función consiste en mantener una colección y facilitar mediante los servicios del personal, el uso de los documentos necesarios para satisfacer las necesidades de información, investigación, educación y ocio de sus lectores.” (1).

Debido al indetenible desarrollo tecnológico de la informática y de las comunicaciones se hicieron necesarias nuevas formas de acceder a la información por parte de los usuarios de manera rápida y eficiente, para satisfacer estas necesidades surgen las bibliotecas digitales. "El concepto de biblioteca digital no es únicamente el equivalente de repositorios digitalizados con métodos de gestión de la información. Es más bien, un entorno donde se encuentran 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" (2). Es importante mencionar que el entorno de la biblioteca digital no tiene por qué ser físico y que sus servicios se brindan de manera dinámica por la red.

Un aspecto a tener en cuenta en las bibliotecas digitales es la gran gama de servicios que aportan, uno de los más importantes sin duda es el intercambio de datos. Estos sistemas pueden establecer comunicación con otros sistemas posibilitando la transferencia e intercambio de información entre ellos. Este proceso se conoce con el nombre de interoperabilidad. Los sistemas de información interoperables deben implementar determinada norma o estándar que permita dicha comunicación de manera rápida y transparente.

En Cuba, en la (UCI) se desarrolla un Sistema Integrado de Gestión Bibliotecaria a cargo del proyecto “Sistema integrado de gestión bibliotecaria” del Centro Informatización de la Seguridad Ciudadana (ISEC), el cual desarrolla soluciones para bibliotecas que pretendan implantar un sistema con estas características: bajo licencias GPL de *software* libre y código abierto, la aplicación está basada en Koha, pionero dentro de los sistemas integrados para la gestión bibliotecaria libres, el cual es adaptado en

dependencia de las necesidades de la institución interesada. De esta manera, el proyecto presenta una solución adecuada para el funcionamiento de la biblioteca universitaria.

El SIGB de la Biblioteca de la UCI cuenta con varias funcionalidades y prestaciones pero en estos momentos solo es capaz de acceder a la información que suministran otros sistemas por la web mediante el protocolo Z39.50 quedando así muy pobre su interoperabilidad. En la UCI la iniciativa de acceso abierto va en incremento encontrándose sitios como el repositorio institucional que brindan sus datos a través del protocolo OAI-PMH.

El repositorio institucional de la UCI guarda toda la información científica que se genera en la universidad, esta información también debe almacenarse en el SIGB de la biblioteca mediante el proceso de catalogación ya que los documentos digitales como las tesis tienen una copia física útil para los servicios que brinda la biblioteca entre ellos el préstamo, la catalogación es de vital importancia ya que luego de realizarse por los bibliotecarios permite a los usuarios hacer búsquedas y obtener la copia en formato duro de los materiales digitales para el préstamo. Consiste en describir en el SIGB los registros bibliográficos de toda nueva adquisición que ingrese en la biblioteca tanto en forma física como electrónica utilizando el formato MARC21.

Actualmente el proceso de catalogación de los registros del repositorio institucional en el SIGB se realiza de forma manual por el bibliotecario interactuando con el sistema, es decir, el bibliotecario cataloga toda la documentación registro por registro ingresando los datos en el sistema. Esto trae como consecuencia una demora significativa en el proceso, esto unido al margen de error humano hacen que el proceso sea difícil y lento.

Dada la situación problemática anteriormente expuesta se plantea como **problema científico**: ¿Cómo facilitar el proceso de catalogación de los registros bibliográficos contenidos en el repositorio institucional en el SIGB?

Definiéndose como **objeto de estudio** de esta investigación la interoperabilidad entre aplicaciones web y como **campo de acción** los protocolos que se emplean para la comunicación en sistemas integrados de gestión bibliotecaria.

De acuerdo con la problemática planteada se propone como **objetivo general**: Desarrollar un módulo en el SIGB que permita importar registros bibliográficos desde el repositorio institucional para la catalogación por copia en la biblioteca de la UCI.

Con el propósito de cumplir con el objetivo trazado, se plantean las siguientes tareas de la investigación:

- Evaluación de las tendencias actuales de los Sistemas Integrados de Gestión Bibliotecaria.
- Estudio de las particularidades del protocolo OAI-PMH.
- Estudio del proceso de catalogación por copia.
- Descripción de los requisitos funcionales y no funcionales para el módulo a desarrollar.
- Elaboración del modelo de caso de uso del sistema.
- Diseño de las interfaces de usuario.
- Diseño de la base de datos.
- Implementación de los elementos de diseño.
- Implementación de los requisitos identificados.
- Implementación de las pruebas del desarrollador.
- Definición de las configuraciones del entorno de prueba.
- Ejecución de las pruebas del desarrollador.
- Integración con el sistema.

Para el desarrollo de las tareas de la investigación se utilizan los siguientes métodos científicos:

Métodos teóricos:

Análisis histórico-lógico: Permite estudiar el modo en que han evolucionado los estándares y normas existentes para facilitar el intercambio de datos entre sistemas de información.
Analítico-sintético: Posibilita descomponer y distinguir los principales elementos que intervienen en la comunicación que puede establecerse entre dos sistemas, así como el protocolo que se seleccione para el desarrollo del módulo a construir, con el objetivo de estudiar cada uno de estos elementos por separado y posteriormente integrarlos, de modo que se logre un mayor entendimiento y se pueda arribar a conclusiones que contribuyan a la solución del problema que dio origen a la investigación.

Método empírico:

Observación: Guía el estudio del estado del arte, permitiendo realizar un análisis sistémico, selectivo y objetivo de los principales sistemas que en la actualidad pueden realizar intercambios de información entre ellos.

El presente documento está compuesto por una introducción, cuatro capítulos de contenido, conclusiones generales, recomendaciones, referencias bibliográficas, bibliografía y anexos. Los capítulos están estructurados de la siguiente manera:

Capítulo 1. Fundamentos teóricos de la interoperabilidad y los protocolos para garantizarla: se expone un conjunto de conceptos y criterios fundamentales para lograr un mayor entendimiento de lo que es la interoperabilidad. Se plasma el resultado de un estudio realizado sobre los estándares que permiten la interoperabilidad entre sistemas de información, específicamente entre sistemas de gestión de documentos. Se analizan los principales proveedores de servicios existentes. Además, se exponen las distintas tecnologías a utilizar en la implementación del módulo, así como la metodología de desarrollo de *software* a emplear.

Capítulo 2. Características del módulo para garantizar la prestación de servicios según OAI-PMH para el SIGB: en este capítulo se realiza una descripción detallada de la solución que se propone para erradicar el problema que propició el origen de la presente investigación. Se representan los principales conceptos que se manejan en el contexto del sistema mediante un modelo de dominio. Se realiza una descripción de los requisitos funcionales y no funcionales. A partir de los requisitos obtenidos, se definen los casos de uso y los actores que se relacionan con cada uno de ellos, confeccionando el diagrama de casos de uso.

Capítulo 3: Análisis y diseño de la propuesta de solución. Se describe el desarrollo y construcción de la propuesta solución mediante el análisis y diseño reflejándose estos en la confección de los diagramas pertinentes.

Capítulo 4. Implementación y pruebas del módulo de prestación de servicios según el estándar OAI-PMH para el SIGB: se representan los elementos físicos necesarios para un correcto despliegue de la aplicación, empleando para ello un diagrama de despliegue. Se muestran componentes de la implementación. Se realiza la validación y prueba de la solución de acuerdo a los requisitos que debe cumplir para garantizar una calidad óptima, todo esto a través de pruebas funcionales.

Capítulo I. Fundamentación teórica

En este capítulo se explicaran los elementos conceptuales necesarios para comprender el ámbito del proceso de interoperabilidad para el desarrollo de la solución. Se abordará sobre que es la interoperabilidad, los protocolos que la desarrollan y los formatos que la soportan; así como las distintas herramientas existentes en la actualidad que sirven para la comunicación entre sistemas gestores de documentos para lograr la catalogación por copia. Se profundizará en las características fundamentales de los protocolos que garantizan la interoperabilidad. Por último se realizará un análisis y selección de la metodología, herramientas y tecnologías a utilizar para el diseño de la solución a desarrollar.

1.1 Conceptualización

Como profesionales de la información estamos sumergidos completamente en la llamada “explosión digital” ocasionada por el creciente desarrollo de la tecnología e Internet como medio masivo de acceso y socialización de la información. Todo esto ha creado nuevas oportunidades y perspectivas a la colaboración entre los profesionales de la información los cuales deben de enfocarse en crear y adaptar herramientas para un tratamiento más eficiente de la información en el contexto digital.

En este contexto dinámico que ofrece la tecnología digital, ha surgido la Iniciativa de Archivos Abiertos (OAI), esta se creó con la misión de desarrollar y promover estándares de interoperabilidad para facilitar la difusión eficiente de contenidos en Internet, como una alternativa para solucionar inconvenientes que muestran los sistemas de información inmersos, ya que, desarrolla y promueve la interoperabilidad de las normas que tienen por objeto facilitar la difusión eficaz de contenidos, constituyendo una herramienta que apuntala el Movimiento del Acceso Abierto impulsado por científicos y especialistas de la información con la finalidad de colocar al alcance de toda la sociedad los contenidos académicos y científicos (3).

Por lo anteriormente expuesto es de vital importancia estudiar el término interoperabilidad. “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...” (4). Podemos decir entonces que la interoperabilidad garantiza de manera eficiente, rápida y cómoda el intercambio de información, siendo así

una poderosa forma para vincular recursos distribuidos como los sistemas que administran los catálogos de las bibliotecas (5).

Según Gómez Dueñas, es posible definir interoperabilidad como la capacidad que presenta un sistema de información de comunicarse y compartir información efectivamente con otro mediante una interconexión libre y transparente (compartir metadatos, documentos y objetos digitales). En términos de OAI se busca crear una metodología simple, con una forma de entendimiento rápida, que sea de fácil desarrollo y aplicación a partir del uso extendido de los navegadores de Internet (6). A los sistemas de información existentes se les puede incluir o mejorar su capacidad interoperabilidad para así utilizarlas para lograr el intercambio de información mediante el metalenguaje XML, el cual es muy óptimo para el manejo de la información digital.

Entonces es de gran necesidad e importancia considerar la aplicación herramientas que permitan la interoperabilidad entre los sistemas y recursos que abogan por la organización y preservación del contenido digital académico-científico.

El concepto de metadatos tiene sus antecedentes en la catalogación tradicional; por eso, es común encontrarse con la afirmación de que una ficha catalográfica es un ejemplo ilustrativo del término metadatos y su alcance. Como señala Hillman los metadatos han estado presentes desde que los primeros bibliotecarios crearon listas de recursos de información (7). Caplan, quien documenta el nacimiento del término en las Ciencias de la Computación, establece que el prefijo “meta” se utiliza para explicar o hablar “acerca de”; así, por ejemplo, un metalenguaje es un lenguaje utilizado para describir otros lenguajes (8).

Los metadatos han estado con nosotros desde que el primer bibliotecario escribió una lista de los documentos que tenía en un estante, en un pergamino escrito a mano. Como mejor se puede comprender el término metadato es como “datos sobre datos” o “datos referentes a datos” para identificar archivos digitales de conjuntos de datos científicos, sociales y geoespaciales. Por ejemplo, un sistema de metadatos común entre los bibliotecarios, el catálogo de biblioteca, contiene un conjunto de registros de metadatos con elementos que describen un libro u otra publicación en una biblioteca: autor, título, fecha de creación o publicación, materia, y la asignatura topográfica especificando la localización de la publicación en el estante. Por lo que en Internet son la información electrónica dispersa y representan a la descripción bibliográfica de recursos electrónicos así como su ubicación.

La relación entre un registro de metadatos y el recurso al que describe puede darse de una de estas dos formas (9):

- Los elementos pueden estar en un registro separado del documento, como en el caso del registro de un catálogo de bibliotecas.
- Los metadatos pueden estar incluidos, incrustados, en el propio recurso.

Actualmente la aplicación de los metadatos va más allá de etiquetar objetos y recursos ya que posibilita calificar procesos o acciones; por lo que hasta representan una amenaza para el desarrollo de la práctica habitual de la catalogación y el uso del formato MARC (Registro Catalográfico Legible por Máquina) si se consideran sus ventajas frente a los tradicionales sistemas de catalogación. Según las afirmaciones de Ortiz-Repiso, estas ventajas están a la vista (3):

- Permiten establecer relaciones entre registros y ficheros diferentes.
- Son más flexibles en la indización y presentación del texto.
- Son menos complejos y permiten la realización de diferentes búsquedas.

En el mundo de la gestión bibliotecaria es de vital importancia obtener de forma rápida y eficiente los archivos necesarios para las bibliotecas, esto ocurre a través de la catalogación. Catalogar consiste en describir un documento en sus partes esenciales con el fin de identificarlo entre el resto del fondo bibliotecario y de conocer su ubicación en la biblioteca. Se realiza para identificar un documento y recuperarlo. El objetivo de la catalogación es identificar los documentos de forma unívoca (descripción bibliográfica) y agrupar la información para poder recuperar los documentos siguiendo los distintos criterios de selección (10). La catalogación se rige por normas para garantizar uniformidad y fácil comprensión.

El concepto mencionado anteriormente también se aplica a la catalogación digital con la particularidad que los datos quedan en soporte electrónico, es decir, se pueden consultar a través de la web. Cuando se quieren catalogar registros desde otra fuente se aplica lo que se conoce como catalogación por copia que es una nueva oportunidad abierta a las bibliotecas de todo el mundo para copiar la información que estas necesiten desde otras fuentes.

La catalogación por copia se encuentra muy unida a la OAI ya que la OAI le brinda la posibilidad de acceder de manera rápida y eficiente a diversas fuentes que contengan archivos a través de la web. La catalogación por copia brinda las siguientes ventajas:

- Ahorro de tiempo invertido en clasificar y catalogar.

- Mayor uniformidad en la clasificación y uniformidad.
- Mayor calidad en la clasificación y catalogación.

Como se ha visto la catalogación por copia brinda a las bibliotecas inmensas posibilidades de enriquecer su patrimonio, esto unido a la las posibilidades que le brinda la OAI de acceder de manera gratuita y segura a la información hacen de estos dos conceptos una poderosa herramienta en si para gestión bibliotecaria.

1.2 Protocolos para la transferencia y recuperación de información

Un protocolo puede ser definido como un "conjunto de reglas o estándares diseñados para permitir a las computadoras conectarse con otras e intercambiar información con el mínimo error posible" (5).

Son numerosos los protocolos desarrollados a través de los años con el fin de permitir la interoperabilidad entre Sistemas de Información Documental (SID) (11).

- Z39.50
- Simple Digital Library Interoperability Protocol (SDLIP)
- OAI-PMH (Open Archives Protocol)
- Guildford protocol
- Dienst protocol

Sin embargo, solo algunos de estos se han consolidado internacionalmente como estándares aceptados por los profesionales de la información, son usados de manera masiva y su desarrollo y mantención se encuentra avalado por iniciativas tanto públicas como privadas: el Protocolo Z39.50 y el Protocolo OAI-PMH (11). A continuación se realiza una pequeña reseña de alguno de ellos.

Protocolo Z39.50

El objetivo principal de Z39.50 consiste en permitir al usuario realizar búsquedas en bases de datos que cuenten con un servidor Z39.50, sin tener que conocer para ello las sintaxis de búsqueda que utilizan dichos sistemas.

Formalmente, facilita la interconexión entre los usuarios y las bases de datos donde se encuentra la información que necesitan a partir de una interfaz común y da fácil manejo, independientemente del lugar en que se encuentren las bases de datos así como la estructura y la forma de acceso de estas (11).

Lo que hace el protocolo Z39.50 es especificar un conjunto de reglas para gestionar las formas y procedimientos de interconexión remota de computadoras, con el propósito de buscar y recuperar información, aunque su aplicación actual es más amplia ya que incluye la consulta y el intercambio de datos bibliográficos, y la intercomunicación de índices y resúmenes, de información geoespacial, de documentos oficiales, de objetos digitales o de metadatos que describen los documentos de las bibliotecas electrónicas o digitales (11).

Este protocolo debido a su versatilidad facilita considerablemente el trabajo a los bibliotecarios por lo que es utilizado mayoritariamente en sistemas de gestión bibliotecaria.

Protocolo OAI-PMH (Open Archives Initiative - Protocol Metadata Harvesting)

El protocolo OAI-PMH (Open Archives Initiative - Protocol Metadata Harvesting o Iniciativa Abierta de Archivos – Protocolo de Recolección de Metadatos) es una herramienta de interoperabilidad que posibilita el intercambio de metadatos sobre cualquier material almacenado en soporte electrónico. Esta transferencia se realiza desde los proveedores de servicio a través de búsquedas hacia los repositorios de archivos (proveedores de datos) (11).

OAI-PMH usa transacciones HTTP (Hypertext Transfer Protocol o Protocolo de Transferencia de Hipertexto, que es el protocolo usado en cada transacción de la World Wide Web desde un cliente o recolector hacia un servidor o archivo en forma de preguntas (ver anexo 1). Ejemplo: el cliente emite una solicitud al servidor pidiéndole le envié metadatos con los datos que contenga, como respuesta el segundo devuelve un XML donde se incluyen los datos de la petición.

Esta estructura de interoperabilidad demuestra el enfoque de *harvesting* (recopilación), en la que la información es transmitida desde la fuente remota (servidor) al destino (cliente) donde a la información se le añadirá un valor agregado.

“El protocolo soporta múltiples formatos para expresar los metadatos, no obstante requiere que todos los servidores ofrezcan los registros utilizando Dublin Core codificado en XML. Además de este formato cada servidor es libre de ofrecer los registros en otros formatos adicionales (MARC por ejemplo). Un cliente puede pedir que los registros se le sirvan en cualquiera de los formatos soportados por el servidor, esto con el fin de que en el futuro las diferentes comunidades que utilicen el protocolo definan sus propios formatos que sean más precisos que el Dublin Core. Por ejemplo la comunidad de archivos de e-prints

está trabajando en un formato denominado AMF (Academic Metadata Format) que describirá todos los elementos que intervienen en el proceso de comunicación científica: documentos, autores, instituciones y canales de distribución de documentos.” (12).

La arquitectura de OAI-PMH se basa en clientes y servidores; los primeros son los archivos que proporcionan la información y los segundos son los recolectores o servicios que toman los datos, y los presentan a los usuarios finales. Para obtener un valor agregado.

El protocolo elegido para el desarrollo del trabajo es OAI-PMH debido a que ya el sistema cuenta con Z39.50 y OAI-PMH satisface todas las necesidades actuales del SIGB,

1.3 Flujo actual de los procesos de OAI-PMH

Como ya se explicó el protocolo OAI-PMH usa transacciones HTTP desde un cliente o recolector hacia un servidor en forma de preguntas. 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 (13):

- 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.
- 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.

Además del verbo, la petición puede estar compuesta por algunos argumentos que indican determinadas características que debe tener la respuesta.

- MetadataPrefix: con este atributo se especifica el formato de metadatos con que se requiere la respuesta. Es obligatorio para ListIdentifiers, ListRecords y GetRecord.
- Set: indica el conjunto del cual deben ser devueltos los metadatos. Se utiliza, solo si el repositorio posee una estructura de conjuntos, para los verbos ListIdentifiers y ListRecords, en ambos casos es opcional.
- From: es una fecha que indica que se solicitan solo los registros que han sido modificados o adicionados a partir de ella. Puede ser empleado de conjunto con los verbos ListIdentifiers y ListRecords.
- Until: es una fecha que indica que se solicitan solo los registros que han sido modificados o adicionados hasta ella. Puede ser empleado de conjunto con los verbos ListIdentifiers y ListRecords.
- ResumptionToken: se utiliza cuando se generan respuestas extensas y es necesario dividirlas en varias partes, indica las características de la búsqueda que se debe realizar y a partir de que registro debe comenzar la respuesta. Puede ser empleado de conjunto con los verbos ListSets, ListIdentifiers y ListRecords.

Existen aspectos que no son tratados por el protocolo, siendo las cuestiones de gestión o autorización para el acceso de los clientes un ejemplo de ello. El servidor deberá recurrir a métodos externos si desea limitar a los clientes en el sentido de la utilización que se le puede dar a los datos. Tampoco trata el tema de cómo los clientes pueden localizar aquellos servidores que contengan los datos que necesitan.

Otro aspecto importante que no es tomado en cuenta por el protocolo es la estandarización de la granularidad de las fechas, cada repositorio es libre de emplear el formato deseado, cuestión que provoca problemas de incompatibilidad a la hora de realizar recolecciones por parte de los proveedores de servicios. Además, el protocolo establece que el control del flujo de la transmisión de los metadatos es opcional, sin embargo este aspecto debería considerarse obligatorio para proveedores de datos que almacenan grandes volúmenes de información, de modo que se garantice el éxito de la transmisión.

En este sentido, el protocolo no especifica completamente como manejar la señal de reanudación, el tiempo de expiración de dicha señal varía en cada repositorio, por lo que se da el caso de que proveedores de servicios intentan continuar con una recolección que por algún motivo fue interrumpida

durante un determinado tiempo y el proveedor de datos no es capaz de responder correctamente, por lo que se hace necesario comenzar la recolección desde el principio.

1.4 Aplicaciones capaces de generar proveedores de servicios

A continuación se describe una serie de aplicaciones capaces de generar proveedores de servicios a partir de sus recolectores correspondientes. La mayor parte de las herramientas generadoras de proveedores de servicios y los paquetes de *software* recolectores han sido desarrolladas por los mismos creadores de aplicaciones para proveedores de datos (14).

- **ARC**: Es un servicio experimental creado con el objetivo de investigar temas relacionados con la recolección de metadatos siguiendo el protocolo OAI-PMH y cómo hacerlos disponibles a los usuarios. Más que un servicio en sí mismo es un *software* que podría ser utilizado por instituciones que quieran crear sus propios servicios. El código fuente está disponible en la red de forma gratuita. Ha sido desarrollado por el Digital Library Research Group de la Old Dominion University (14).
- **Open Harvester Systems (OHS)** es un sistema de indexación de metadatos gratuito desarrollado por el Public Knowledge Project a través de los esfuerzos financiados por el gobierno federal de los Estados Unidos para ampliar y mejorar el acceso a la investigación.

  OHS le permite crear un índice de búsqueda de los metadatos de la OAI compatibles con archivos, tales como los sitios que utilizan Open Journal Systems (OJS) o Open Conference Systems (OCS) (15).
- **Net::OAI::Harvester**

  Net::OAI::Harvester es una extensión en Perl para consultar repositorios OAI-PMH de forma sencilla y recolectar metadatos de estos. Desarrollado por los programadores Ed Summers y Martin Emmerich [Summers, 2004], se puede descargar su versión 1.0 (de julio de 2005) en el archivo de recursos en Perl CPAN (Comprehensive Perl Archive Network).

  En cuanto a las características específicas de Net::OAI::Harvester, destacan su empleo de un analizador o *parser* (XML::SAX) para dividir los resultados de una consulta OAI-PMH de lista con posible respuesta larga, como ListRecords. Al dividir el documento XML de la respuesta se crean una serie de objetos Perl (registros, cabeceras...), que se almacenan en el disco duro de forma seriada y facilitan que se utilice una menor memoria de procesamiento. Además, los filtros

XML::SAX permiten que cada implementador desarrolle sus propios paquetes para la indización de metadatos, de manera que acepten no solo metadatos en Dublin Core sin cualificar sino cualquier otro formato (14).

- **OAIHarvester2**

  El proyecto OAIHarvester2 es una aplicación en Java que proporciona un recolector de metadatos de repositorios que cumplen OAI-PMH v.1.1 y v2.0. Es un desarrollo de *software* libre de la OCLC63, que se distribuye de forma gratuita mediante licencia Apache versión 2.

  En palabras de su desarrollador, Jeffrey Young, OAIHarvester2 resulta más ligera que su predecesora. Puede correr como una aplicación independiente o integrarse en aplicaciones existentes. Mientras que la antigua aplicación tenía un interfaz en Java que permitía el procesamiento de los registros recolectados sobre la marcha, la nueva versión simplemente agrupa las respuestas OAI en un archivo XML para su posterior procesamiento (p. ej. con XSLT) (14).

Es importante mencionar que todas las aplicaciones descritas anteriormente son de *software* libre. Como se puede apreciar son muy diversas las herramientas para desarrollar proveedores de servicios, las misma abarcan un gran número de lenguajes y tecnologías. Para generar un proveedor de servicios se deben de considerar factores como las funcionalidades que se desean implementar, el lenguaje sobre el cual se desarrollará, la disponibilidad y fortaleza de la aplicación, su experiencia en uso, etc.

1.5 Sistemas que implementan OAI-PMH

Desde 2001 se ha mantenido constante el crecimiento del número de archivos que han implementado el protocolo OAI-PMH, hasta llegar a más de 45 archivos de variadas disciplinas. Se pueden destacar el CERN (informes y publicaciones en Física), Citebase, y los archivos del área de Bibliotecología y Documentación, tales como @rchiveSIC, DLIST (Digital Library of Information Science and Technology) y E-LIS (Eprints in Library and Information Science) (11).

De la misma forma los proveedores de servicios han crecido, siendo los más reconocidos a nivel mundial:

- ARC http://dlib.cs.odu.edu/ARC.html
- OAIster http://www.oclc.org/oaister/
- Perseus. http://www.perseus.tufts.edu/hopper/
- Cyclades. http://www.ercim.eu/cyclades/

- Scirus http://www.scirus.com/

Según la especificación de OAI-PMH un proveedor de servicios se define de forma escueta como aquel sistema que utiliza los metadatos recolectados a través del protocolo OAI-PMH para construir servicios de valor añadido (ver anexo 2). Todos estos servicios actualmente son registrados aunque esto no es obligatorio por lo que no se conoce con certeza el número que existen.

Ahora se muestra un listado de los repositorios cubanos que poseen este tipo de protocolos:

- INFOMED (Centro Nacional de Información de Ciencias Médicas), Cuba
- CIGETCAV
- Instituto cubano de radio y televisión
- Scielo Cuba

Solamente existe en la actualidad un repositorio de información cubano registrado en el Directorio de Repositorios de Acceso Abierto. Se trata del proyecto Scientific Electronic Library Online-Cuba (Scielo-Cuba**)** que contiene una amplia colección de revistas médicas cubanas.

El Ministerio del Poder Popular para la Educación Universitaria en Venezuela se encuentra desarrollando el proceso de municipalización de la Educación Superior, en apoyo a este proceso se creó la Biblioteca Digital “Alma Mater”, como espacio para acceder a los repositorios de universidades y a los artículos de revistas científicas de interés para los distintos programas de formación.

En el año 2012 en la UCI en el Centro de Informatización Universitaria (CENIA), se desarrolló un sistema de indexación de documentos para la biblioteca venezolana. El sistema de indexación de documentos para la Biblioteca Digital "Alma Mater" se basa en el protocolo OAI-PMH y le permite a dicha biblioteca recolectar de forma rápida documentos disponibles en fuentes externas específicamente en canales de información de acceso abierto (16).

El sistema de indexación de documentos para la biblioteca digital ¨Alma Mater¨, se creó como un módulo para el sistema de gestión de contenidos Drupal, en su versión 6.17, el módulo llamado *dataprovider_reader* se basó en el protocolo OAI-PMH, específicamente empleó el verbo *identify* para comprobar que la fuente de la que se van a indexar documentos es real y confiable. Se usó el *ListMetadataFormats* para verificar que se emplea Dublin Core como formatos de metadatos y

ListRecords para recuperar la información de todos los documentos que se indexan en la biblioteca (ver anexo_4).

En la Universidad de las Ciencias Informáticas también se implantó el Open Harvester Systems (OHS) (ver anexo_3), y se diseñó un buscador que realiza búsquedas sobre la base de datos indexada por el OHS (ver anexo_5).

En resumen, los proveedores de servicios están proliferando y cada vez están proporcionando servicios más sofisticados. Se puede decir que existe un mercado donde los servicios pueden competir.

1.6 Estándares de metadatos

El concepto de metadato existe mucho antes del de Internet y la Web, actualmente hay un creciente interés internacional por las normas y prácticas de los mismos, esto se debe al incremental crecimiento de las publicaciones electrónicas y las bibliotecas digitales, ocasionando esto lo que se conoce hoy como "sobrecarga de información" por la inmensa cantidad de información digital disponible en línea.

Dicho interés constituye una parte fundamental del desarrollo de la Web semántica, por adoptar a gran escala estándares y prácticas descriptivas para los recursos electrónicos porque ello contribuirá a mejorar la recuperación de recursos relevantes en cualquier contexto. Como señalan Weibel y Lagoze, dos líderes en el campo del desarrollo de metadatos:

"La asociación de metadatos descriptivos normalizados de los objetos de la red tiene el potencial para mejorar sustancialmente las capacidades de localización y recuperación, facilitando búsquedas basadas en campos (p. ej. autor, título), permitiendo la indización de objetos no textuales, y facilitando el acceso al contenido sustituido/referenciado que es distinto del acceso al contenido del propio recurso" (17).

Hoy se cuentan con varias iniciativas de estandarización de metadatos según el área donde se vayan a aplicar, ya que es imposible contar con un solo estándar porque no podría recoger todos los requisitos específicos. Por ejemplo, los primeros campos en los que se empezó a investigar el uso de metadatos fueron en humanidades y a través de TEI (Text Encoding Initiative), biología, y en las ciencias geoespaciales.

Para el desarrollo del trabajo de diploma se utilizará Dublin Core por ser el formato requerido por el protocolo y MARC21 ya que es el que utiliza el sistema.

DCMI (Dublin Core Metadata Initiative*)* es una organización internacional y virtual albergada por OCLC (Dublin Ohio, EE.UU) que se financia a través de proyectos y subvenciones y en la que el trabajo se realiza por voluntarios de todo el mundo.

Constituye un mecanismo básico de descripción que puede usarse en todos los dominios, para todo tipo de recursos, sencillo pero potente, que puede extenderse fácilmente y puede trabajar conjuntamente con otras soluciones específicas (18). Sus objetivos son:

- Desarrollar estándares de metadatos para la recuperación de información en Internet entre diversos dominios informativos.
- Definir marcos de trabajo para la interoperabilidad entre conjuntos de metadatos.
- Facilitar conjuntos de metadatos específicos para una comunidad o disciplina, acordes a los apartados 1 y 2.

Los principales temas de trabajo se concretan en:

- Expresar metadatos Dublin Core en XML, RDF/XML, etc
- Extensión y uso de Dublin Core a dominios específicos de información (Perfiles de aplicación)
- El Registro DCMI (diccionario).
- Citación, Agentes, Colecciones
- Soporte para el desarrollo de herramientas

Elementos de Dublin Core.

Dublin Core cuenta con los siguientes campos: Título, Creador, Colaborador, Editor, Fecha de publicación, Identificador (URI), Materia, Descripción, Cobertura, Tipo de recurso, Idioma, Formato, Fuente, Relación con otros documentos, Derechos.

Los mismos pueden calificarse en tres categorías.

- Sobre el contenido del recurso.

Etiquetas DC	Descripción
DC.Title	Título. El nombre dado al recurso.
DC.Subject	Materias y palabras clave. El tema del contenido del recurso.
DC.Description	Descripción del contenido del recurso. Puede incluir un resumen, una tabla de

	contenidos, etc.
DC.Source	Fuente. Referencia al recurso del que deriva el documento actual.
DC.Languaje	Lenguaje. El idioma del contenido del recurso.
DC.Relation	Relación. Una referencia a un recurso relacionado con el contenido.
DC.Coverage	Cobertura. Ámbito del contenido del recurso. Puede tratarse de una especificación geográfica, temporal o legal.

Tabla 1. Contenido del recurso Dublin Core.

- Sobre la propiedad intelectual del recurso.

Etiquetas DC	Descripción
DC.Creator	Autor. Responsable de la creación del contenido. Puede ser una entidad, una persona o un servicio.
DC.Publisher	Editor. Responsable de que el recurso se encuentre disponible.
DC.Contributor	Colaborador. Responsable de hacer colaboraciones al contenido del recurso.
DC.Rights	Derechos. Información sobre los derechos de la propiedad intelectual del recurso, como por ejemplo el *copyright*.

Tabla 2. Propiedad intelectual Dublin Core.

- Sobre la instancia del recurso.

Etiquetas DC	Descripción
DC.Date	Fecha. Fecha asociada a la creación o modificación del recurso. Se suele seguir la notación AAAA-MM-DD.
DC.Type	El tipo o categoría del contenido. Palabras clave de un vocabulario que describen la naturaleza del recurso.
DC.Format	Formato. Descripción física del recurso, como su tamaño, duración, dimensiones, etc. si son aplicables. Se suelen usar tipos MIME.
DC.Identifier	Identificación. Referencia unívoca para el contenido del recurso. Por ejemplo una URL o un ISBN.

Tabla 3. Instancia Dublin Core.

Dublin Core puede, por tanto, verse como un “pequeño lenguaje para realizar una clase particular de declaraciones sobre recursos". En este lenguaje, hay dos clases de términos: elementos (nombres) y

calificadores (adjetivos), que pueden ordenarse en un patrón simple de una sentencia o declaración. Los sujetos, implícitos o sobreentendidos, de este lenguaje, son los propios recursos.

Elementos de MARC21

MARC es el acrónimo de Machine Readable Catalogue o Cataloging. Esta descripción parece implicar que el MARC sea un tipo de catálogo o un método de catalogación automatizada. Con más propiedad, MARC se puede describir como un grupo de formatos que utilizan un conjunto de convenciones para la identificación y el procesamiento de datos (actualmente no sólo bibliográficos) que un ordenador puede manejar, procesar e intercambiar (19).

Un registro MARC está formado por tres elementos: la estructura del registro, la designación de contenido y los datos contenidos en el registro. Estructura del registro: constituye la realización de la American National Standard for Information Interchange (ANSI/NISO Z39.2) o de la norma equivalente ISO 2709. La estructura de un registro MARC, que constituye su semántica, está compuesta por tres componentes principales (19):

- Cabecera: contiene elemento de datos que proporcionan información de control para el procesamiento del registro. Los datos son números o valores codificados según la posición relativa del caracter. La Cabecera tiene una longitud fija de 24 posiciones y constituye el primer campo de cualquier registro MARC.
- Directorio: contiene tantos bloques de 12 caracteres como campos de longitud variable tiene el registro MARC. Cada bloque incluye la etiqueta, la longitud y la posición de origen de cada campo dentro del registro. Actúa a modo de índice del registro.
- Campos variables: Los datos en un registro MARC están organizados en campos variables, cada uno de los cuales está identificado por una etiqueta de tres caracteres numéricos que está almacenada en el directorio. El registro MARC termina siempre con un caracter separador de registro.

Designación de contenido: etiquetas, códigos y convenciones establecidas explícitamente para identificar y caracterizar posteriormente los elementos de datos contenidos en el registro y para facilitar su manipulación. La designación de contenido, que constituye la sintaxis de un registro MARC, es propia de cada formato y es lo que identifica y distingue un formato MARC del resto de los formatos. Por citar solo

un par de ejemplos, el campo 008 en MARC 21 corresponde al campo 100 de UNIMARC, o el campo 100 en MARC 21 corresponde al campo 700 en UNIMARC. Existen dos tipos de campos variables (19):

- Campos de control variables: En MARC 21 corresponde a los campos 00X. Los campos de control variables son estructuralmente diferentes a los campos de datos variables. Estos campos no contienen ni indicadores ni códigos de subcampo. Pueden contener un solo dato.
- Campos de datos variables: en MARC 21 corresponden a los campos etiquetados entre 01X y 8XX.

En los campos de datos variables se utiliza los siguientes tipos de designadores de contenido:

- Indicadores: Las dos primeras posiciones al inicio de cada campo de datos variable contienen valores que permiten interpretar o completar los datos que se encuentran grabados en el campo a partir del segundo byte del primer subcampo. Los valores de los indicadores se interpretan por separado, según la posición primera o segunda, y pueden ser dígitos, caracteres alfabéticos o espacios en blanco. Por ejemplo, en el campo 245 el segundo indicador indica al sistema que debe de ignorar determinado número de caracteres a efectos de ordenación del título propio.
- Subcampos: Los códigos de subcampo preceden de cada elemento de datos dentro del campo que precisa un tratamiento separado. Un código de subcampo consiste en un delimitador seguido de un identificador del elemento de datos. Los identificadores de elementos de datos pueden ser representados por caracteres alfabéticos en minúsculas o numéricos. Los códigos de subcampos están definidos de forma independiente para cada campo, aunque el formato intenta preservar el mismo código para significados similares donde es posible. El objetivo de los códigos de subcampos es identificar elementos de datos. El formato no establece el orden de los subcampos, que viene dado por la norma prevista para el contenido de datos, como por ejemplo unas reglas de catalogación.
- Contenido de datos: los datos contenidos en un registro MARC suelen estar definidos por normas que no forman parte del formato, como las ISBD, las AACR2, los LCSH o cualquier otra regla de catalogación, lista de encabezamientos o clasificación que utiliza la agencia que crear el registro. Solo determinados elementos de datos codificados están definidos por el mismo formato MARC, como por ejemplo ciertas posiciones de la cabecera.

Después del estudio ambos formatos (Dublin Core y MARC21) se llegó al establecimiento de los campos equivalentes entre ambos.

Dublin Core	Marc21
Contributor	720 ##$
Coverage	500$a
Creator	720 ##$a
Date	260 ##$c
Description	520 ##$a
Format	856 ##$q
Identifier	856 40 $u
Language	546 ##$a
Publisher	260 ##$b
Rights	540 ##$a
Source	786 0#$n
Subject	653 ##$a
Title	245 00$a
Type	655 #7$a

Tabla 4. Equivalencia Dublin Core y Marc21.

Esta correspondencia permite la conversión de los registros de Dublin Core a MARC21 necesaria a la hora del proceso de catalogación por copia.

1.7 Análisis de las soluciones encontradas

El estudio de las soluciones anteriormente mencionadas permitió recopilar elementos necesarios para la elaboración de una propuesta de solución al problema planteado más completa. De cada uno de estos sistemas fueron tomados los aspectos positivos pues presentan similitudes con el sistema a desarrollar. Aunque el SIGB implementa Z39.50 el repositorio institucional brinda sus datos bajo el protocolo OAI-PMH razón por la cual se escogió este protocolo para la elaboración del módulo.

A continuación se muestra una tabla con las características más importantes a tener en cuenta para la elección de una de estas herramientas.

Aplicación	Funcionalidades OAI-PMH	Lenguaje	Disponibilidad	Experiencia	Integración con el SIGB
ARC	Todas	Java	Alta	Alta	Media
Open Harvester Systems (OHS)	Todas	PHP	Alta	Alta	Media
Net::OAI::Harvester	Todas	Perl	Alta	Media	Alta
OAIHarvester2	Todas	Java	Media	Media	Media

Tabla 5. Analizas de las soluciones encontradas.

De todas las posibles soluciones se seleccionó Net::OAI::Harvester, debido a que está desarrollada en el lenguaje del sistema (Perl), su integración con el SIGB es muy sencilla ya que se encuentra en forma de módulo, además, se encuentra disponible en la página de Perl (CPAN) y cumple con todas las funcionalidades necesarias para la correcta implementación del proveedor de servicios, esto unido a que con la herramienta Net::OAI::Harvester se puede desarrollar un módulo en el mismo SIGB lo cual le serviría cuando fuese a implantarse en otras instituciones. Se descartó la solución Open Harvester Systems (OHS) que aunque es muy usado en la actualidad su uso implicaría la instalación de un sistema externo ya que esta desarrollado en otro lenguaje de programación y sería difícil su integración en el e SIGB.

Actualmente SIGB puede recolectar metadatos de proveedores de datos solo por el protocolo Z39.50. Teniendo en cuenta este elemento y los expuestos anteriormente se concluye que es necesaria la implementación de un módulo con la utilización del módulo Net::OAI::Harvester que le permita a SIGB comportarse como un proveedor de servicios según lo establecido por el protocolo OAI-PMH.

1.8 Tecnologías necesarias para la implementación del módulo

Las tecnologías que se describen a continuación tienen en común que no son privativas, las misma son las definidas por el proyecto para el desarrollo de la solución de *software* a desarrollar.

1.8.1 Herramienta de modelado: Visual Paradigm

Esta herramienta da soporte al modelado visual que permite UML ofreciendo un entorno de creación de diagramas. Permite el diseño centrado en casos de uso y enfocado al negocio posibilitando la generación de un *software* de gran calidad. Cuenta con un lenguaje estándar y común para todo el equipo de desarrollo que hace posible una mejor comunicación. Proporciona la ventaja de realizar ingeniería directa e inversa. El modelo y el código permanecen sincronizados en todo el ciclo de desarrollo. En el Visual Paradigm en su versión 8.0, se puede construir diferentes tipos de diagramas que permiten ver el sistema desde diferentes perspectivas, entre ellos se encuentran los de casos de uso, de clase, actividad, estado, componentes, secuencia, entre otros. Otra de sus ventajas es que permite hacer paquetes de trabajo que proveen de un mecanismo de organización de los modelos agrupando elementos de modelado, siendo esto de gran ayuda a la hora de desarrollar sistemas de gran envergadura y complejidad.

Considerando todas las facilidades brindadas por Visual Paradigm y que está disponible en la universidad una licencia para su uso, permite llegar a la conclusión, que es la mejor opción para realizar el modelado de la solución y para ello se utiliza en la versión Visual Paradigm for UML 8.0 que se caracteriza por:

- *Software* libre.
- Disponibilidad en múltiples plataformas (Windows, Linux).
- Diseño centrado en casos de uso y enfocado al negocio que generan *software* con mayor calidad.
- Capacidades de ingeniería directa e inversa.
- Soporta aplicaciones web.
- Diagramas de flujo de datos.

1.8.2 Entorno de desarrollo integrado: Eclipse

Eclipse es un entorno de desarrollo integrado (sigla IDE en inglés de Integrated Development Environment), principalmente es una plataforma de programación, usada para crear proyectos de desarrollo. Eclipse comenzó como un proyecto de IBM Canadá. Fue desarrollado por el Objeto Internacional de Tecnología (sigla OTI en inglés de Object Technology International) como reemplazo de

Visual Age también desarrollado por OTI. En noviembre del 2001, se formó un consorcio para el desarrollo futuro de Eclipse como código abierto.

Eclipse emplea módulos (en inglés *plug-in*) para proporcionar todas sus funcionalidades como una plataforma de cliente enriquecido, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, ya sea que el usuario las necesite o no.

Este mecanismo de módulos es una plataforma ligera para componentes de *software*, adicionalmente le permite al Eclipse extenderse usando otros lenguajes de programación como son C/C++, Java, PHP, Perl, Python y otros (20). Uno de los módulos destinado y que posibilita la integración para la programación en el lenguaje Perl es el EPIC.

1.8.3 Lenguaje unificado de modelado: UML

Por sus siglas en inglés, Unified Modeling Language, es el lenguaje de modelado de sistemas de *software* más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de *software*. UML es un lenguaje expresivo, claro y uniforme, que no garantiza el éxito de los proyectos pero si mejora sustancialmente el desarrollo de los mismos, al permitir una nueva y fuerte integración entre las herramientas, los procesos y los dominios. Es independiente del proceso, aunque para utilizarlo óptimamente se debe usar en un proceso que sea dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental. Es importante resaltar que UML es un lenguaje para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de *software*, para detallar los artefactos en el sistema y para documentar y construir (21).

1.8.4 Lenguaje de programación: Perl

Es un lenguaje de programación diseñado por Larry Wall y creado en 1987. Perl fue originalmente desarrollado para la manipulación de texto y en la actualidad es ampliamente utilizado en la administración de sistemas y en el desarrollo web. Perl toma características de C y del lenguaje interpretado *bash*. La estructura completa de Perl deriva del lenguaje C, Perl es un lenguaje imperativo, con variables, expresiones, asignaciones y bloques de códigos delimitados por llaves. Perl tiene una gran potencia en la manipulación de textos debido a que incluye expresiones regulares que facilitan el trabajo con textos. Perl soporta la implementación de módulos que permiten en gran medida la separación de las funcionalidades

del sistema y la reutilización de código. Entre las características que hacen que Perl, sea el lenguaje utilizado para el desarrollo del sistema se encuentran:

- Es un lenguaje libre, por lo cual es posible estudiar, modificar o distribuir cualquier código de Perl sin tener que pagar un centavo, ni estar sujeto a demandas.
- Es multiplataforma, o sea, puede funcionar lo mismo en un servidor web que tenga instalado un sistema operativo Windows o Linux.
- Es un lenguaje de programación utilizado para construir aplicaciones web con ayuda del módulo CGI.
- Muy práctico para extraer información de archivos de texto y generar informes a partir del contendido de los ficheros

Junto a lo antes mencionado cabe mencionar que es lenguaje en el cual se encuentra desarrollado el Koha y por consecuencia el SIGB.

Módulos de Perl

Un módulo es un conjunto de operaciones que pueden ser utilizadas en cualquier *script* en Perl, simplemente se llama módulo al que se hace referencia y las operaciones que de él nos interesa utilizar. Es evidente que dentro de un *script* se pueden llamar diversos módulos según lo que efectivamente le sirva a quien esté programando. En el sistema se utilizan varios módulos de Perl, unos escritos por la comunidad que soporta al lenguaje certificados internacionalmente, y otros escritos por la comunidad de desarrollo de Koha (22).

Todos los módulos mencionados a continuación se utilizaron debido a que son los empleados por el sistema, en el cual han demostrado su eficiencia y funcionalidad.

Módulo para el soporte de CGI

Este módulo utiliza objetos de Perl para que sea más fácil llenar los formularios web y analizar su contenido. Este paquete define objetos CGI, entidades que contienen la cadena de consulta actual y otras variables de estado. Si se utilizan los métodos de un objeto CGI, se pueden examinar las palabras claves y los parámetros pasados a un *script* y crear formularios, cuyos valores iniciales son tomados de la consulta actual, lo que da la preservación de la información de estado. El módulo ofrece funciones de acceso directo que producen lenguaje HTML, reduciendo la mecanografía y lo errores de codificación.

También proporciona algunas de las funcionalidades más avanzadas de los CGI e incluye soporte a la carga de archivos, *cookies*, hojas de estilo en cascada y marcos. Existen dos formas de programación con el módulo CGI: uno es el estilo orientado a objetos y el otro es el estilo funcional u orientado a funciones (23).

Módulo DBI

DBI (Interfaz de bases de datos independiente) es un módulo de acceso a bases de datos para el lenguaje de programación Perl. Este define un conjunto de funciones, variables y convenciones que ofrecen una interfaz de base de datos consistente e independiente de la base de datos que se esté utilizando, es decir, DBI es la interfaz independiente para base de datos de Perl, lo cual no significa que no haya otros, pero normalmente todo lo que se puede hacer con un módulo que no use DBI para acceso a bases de datos, se puede hacer con este, de forma más fácil y portable (22).

Es importante destacar que DBI es una capa de "pegamento" entre una aplicación y uno o más módulos controladores de base de datos. Es el módulo controlador el que hace mayor parte de la verdadera labor.

Módulo DBD

DBD (Driver de Base de Datos, del inglés Data Base Driver) es el *driver* de la base de datos, es decir, se encarga de llevar a cabo lo que se pide que se haga en el *script* Perl (usando DBI) en una base de datos específica. Existe un módulo DBD para cada tipo de base de datos, dicho módulo se encarga de pasar las peticiones que se realizan en DBI a peticiones a la base de datos sobre la que se esté trabajando. Para trabajar con una base de datos determinada hace falta tener controlado el módulo DBD correspondiente, por ejemplo, para trabajar con la base de datos MySQL hace falta el módulo DBD-MySQL (22).

Módulo Net::OAI::Harvester

Net::OAI::Harvester es una extensión en Perl para consultar repositorios OAI-PMH de forma sencilla y recolectar metadatos de estos. Desarrollado por los programadores Ed Summers y Martin Emmerich, se puede descargar su versión 1.0 (de julio de 2005) en el archivo de recursos en Perl CPAN (Comprehensive Perl Archive Network) (14).

Se utilizó este módulo ya que esta desarrollado en Perl, (el lenguaje del sistema) además el mismo satisface todas las necesidades funcionales necesarias para la recolección por el protocolo OAI-PMH. Se trabajó con la versión OAI-Harvester-1.15., disponible en el sitio oficial de Perl (CPAN).

1.8.5 Lenguaje de Marcado Extensible (XML)

Formato simple de texto muy flexible derivado de SGML (ISO 8879). Originalmente diseñado para afrontar los retos de las grandes publicaciones electrónicas. XML es un lenguaje que permite jerarquizar y estructurar la información y describir los contenidos dentro del propio documento, así como la reutilización de partes del mismo. La información estructurada presenta varios contenidos (texto, imágenes, audio, etcétera) (24).

Ofrece un formato para la descripción de datos estructurados, lo que facilita que las declaraciones de contenido sean más precisas y los resultados de búsquedas sean más significativos. Proporciona interoperabilidad mediante un formato basado en estándares flexibles y abiertos, con formas nuevas de acceso a las bases de datos existentes y de entregar datos a clientes de la Web.

Este leguaje se utilizó debido a que las respuestas de las peticiones del protocolo OAI-PMH se envían en XML.

1.8.6 Lenguaje de programación del lado del cliente: JavaScript

JavaScript es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes. Este lenguaje cuenta con varias características, es un lenguaje basado en acciones que tiene menos restricciones. Además, gran parte de la programación en este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del *mouse*, aperturas, utilización de teclas, cargas de páginas, entre otros. Para el desarrollo de la solución se utiliza javascript en su versión 1.8, esto se determina según el navegador utilizado que soporte algunas de las versiones existentes. Fue necesario hacer uso de este lenguaje para crear diferentes efectos e interactuar con los usuarios a la hora de realizar algunas acciones no válidas que puedan atentar contra la seguridad del sistema como en la entrada de datos en los formularios.

1.8.7 Lenguaje de Marcas de Hipertexto: HTML

El HTML, Hyper Text Markup Language (Lenguaje de Marcas de Hipertexto) es un lenguaje muy sencillo que permite describir hipertexto, es decir, texto presentado de forma estructurada, agradable, con enlaces que conducen a otros documentos o fuentes de información relacionadas y con inserciones multimedia; es el lenguaje que se utiliza para presentar información en la World Wide Web (WWW) (25). Se utilizó la versión 4.0 para el desarrollo de la solución.

La descripción se basa en especificar en el texto la estructura lógica del contenido (títulos, párrafos de texto normal, enumeraciones, definiciones y citas), así como los diferentes efectos que se quieren proporcionar (cursiva, negrita, o un gráfico determinado), para que la presentación final la realice un programa especializado como los navegadores Internet Explorer o Mozilla Firefox (22).

Este lenguaje indica al navegador dónde colocar el texto, las imágenes y los videos, atendiendo a la disposición deseada para colocarlos en la página. No es más que una serie de etiquetas que se utilizan para definir la forma o estilo que se quiera aplicar al documento.

1.8.8 Sistema gestor de bases de datos: MySQL

El sistema gestor de bases de datos MYSQL, fue se seleccionado por disímiles características, primeramente se integra muy bien al lenguaje Perl y al servidor Apache. Para realizar proyectos sencillos, donde no se requiere almacenar gran cantidad de información, ha demostrado ser muy rápido. Aunque actualmente posee licencia propietaria debido a que fue comprada por la Sun Microsystems, las versiones de MySQL que hayan sido liberadas en fechas anteriores a esa adquisición, serán consideradas con licencia libre, por lo cual la versión 14.14 de MySQL que se usa en este trabajo es libre. Esto supone una gran ventaja si se tiene en cuenta los altos precios que hay que pagar para obtener un gestor de base de datos como Oracle o SQL Server. Otra de las ventajas es que la misma es multiplataforma, por lo cual funciona lo mismo en servidores con sistema operativo Linux y Windows (20).

1.8.9 Servidor web: Apache

Es un servidor altamente configurable, es muy sencillo ampliar las capacidades del servidor web apache. Cualquiera que posea una experiencia mínima en la programación con C o Perl puede escribir un módulo para realizar una función determinada. Esto significa que hay una gran cantidad de módulos Apache disponibles para su utilización.

Es un sistema multiplataforma que funciona a la perfección en sistemas operativos como Windows y Linux, en los cuales brinda un gran rendimiento y escalabilidad. Incluso en el sistema operativo Windows funciona mucho mejor que su otro competidor, el servidor IIS, es muy frecuente ver que muchos servidores Windows utilicen Apache en vez de IIS.

Debido a las disímiles ventajas que tiene este servidor, respecto a sus competidores, se selecciona como servidor web porque es el que se utiliza con el SIGB. Entre las características principales que se encuentran son:

- Es de licencia libre, se pueden realizar múltiples cambios para mejorar su rendimiento sin tener que pagar.
- Es un servidor potente, que tiene buena seguridad y robustez.

1.8.10 Metodología de desarrollo de software RUP

En la realización de proyectos de *software*, es necesario basarse en una metodología de desarrollo de *software* que ayude a organizar y planificar todo el proceso de desarrollo de *software* para poder obtener un producto de óptima calidad y los clientes se sientan satisfechos con el resultado. Un proceso de desarrollo de *software* es la definición del conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de artefactos que explica los pasos necesarios para terminar el proyecto, tiene la misión de transformar los requerimientos del usuario en un producto de *software*. Un proceso define "quién" está haciendo "qué", "cuándo" y "cómo" para alcanzar un determinado objetivo (20).

Entre las metodologías se encuentran las llamadas "ágiles", las cuales permiten desarrollar y obtener rápidamente aplicaciones de menor envergadura y tiempo de desarrollo donde el cliente participa en todo momento a lo largo de todo el proceso, algunos ejemplos son: XP (Programación extrema), Scrum, DSDM (desarrollo de *software* dirigido por modelos) (26); mientras que las "tradicionales" son utilizadas para dirigir procesos más grandes y con un tiempo mayor de duración, es el caso de RUP (Proceso Racional Unificado, por sus siglas en inglés Rational Unified Process).

RUP es una metodología flexible, y fácil de adaptarse a las necesidades de cualquier proyecto, además esta le brinda al equipo de desarrollo guías consistentes y personalizadas del proceso. El mismo en conjunto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP utiliza UML para definir los modelos de *software* y puede definirse como un modelo que es dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental. Se divide en cuatro fases: inicio, elaboración, construcción y transición.

Para darle solución al problema planteado en el presente trabajo se optó por RUP como metodología de desarrollo de *software*, pues cuando surgió el proyecto Gestión Bibliotecaria encargado de desarrollar el

SIGB de la biblioteca de la UCI, en este se estableció usar la metodología RUP por el tamaño y complejidad del sistema, por lo que la solución a desarrollar continúa el uso de la misma para elaborar la documentación en correspondencia con la existente.

1.9 Conclusiones parciales del capítulo

En la actualidad gran parte de los sistemas de gestión de documentos tienen la capacidad de interoperar con otros sistemas. El protocolo para la interoperabilidad que se adapta a las características del SIGB es OAI-PMH. Se decide utilizar el módulo Net::OAI::Harvester por ajustarse a las necesidades planteadas. El protocolo OAI-PMH permitirá importar registros desde el repositorio institucional de la UCI para hacer catalogación por copia hacia el SIGB. Las tecnologías seleccionadas para el desarrollo de la solución están acordes a los requisitos y a las políticas del centro.

Capítulo II. Características del subsistema

En el presente capítulo se describe la propuesta de solución para el subsistema, sus características, funcionalidades y la descripción de la arquitectura utilizada. Además, para el desarrollo de la solución se detallan los dos primeros flujos de trabajo de la metodología propuesta en el capítulo anterior. Estos flujos de trabajos, incluyen de forma general en la elaboración de modelos, diagramas y el planteamiento de los requisitos funcionales y no funcionales que requiere el módulo del sistema, siendo esta una actividad de vital importancia para la implementación.

2.1. Descripción de la propuesta de solución

Con el desarrollo de un módulo que garantice que el SIGB se comporte como un proveedor de servicios según el protocolo OAI-PMH, se le proporciona al sistema una nueva funcionalidad que le permitirá recolectar información de otros sistemas que implementen el protocolo la cual facilitará el proceso de catalogación.

El SIGB podrá realizar peticiones a diferentes proveedores de datos a través del protocolo OAI-PMH. Las peticiones se emitirán utilizando los métodos GET o POST del protocolo HTTP las mismas se realizarán de acuerdo a los seis verbos (*GetRecord, ListRecords, ListIdentifiers, ListSets, ListMetadataFormats e Identify*). Formulada la solicitud será enviada al proveedor de datos, el proveedor de datos deberá analizar la solicitud y elaborar una respuesta codificada en sintaxis XML la cual enviará a través de uno de los métodos anteriormente mencionados al SIGB.

Cuando el SIGB obtenga la respuesta del proveedor de datos (XML) con la ayuda del módulo la analizará y almacenará en su base de datos para su posterior catalogación.

El mismo administrador del sistema será el encargado de definir hacia que proveedor de datos desea recolectar y que información será recolectada, para que después el bibliotecario pueda realizar la catalogación.

2.2. Modelo de dominio

“El modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las “cosas” que existen o los eventos que suceden en el entorno en el que trabaja el sistema” (27). Por tanto el modelo de dominio puede ser tomado como punto inicial para el

diseño del sistema, este puede utilizarse para capturar y expresar los conceptos más importantes del contexto del sistema. Además es una representación visual de los conceptos u objetos del mundo real, significativos para un problema o área de interés. El modelo de dominio es utilizado por el analista como un medio para comprender los procesos de negocios donde el sistema va ser utilizado.

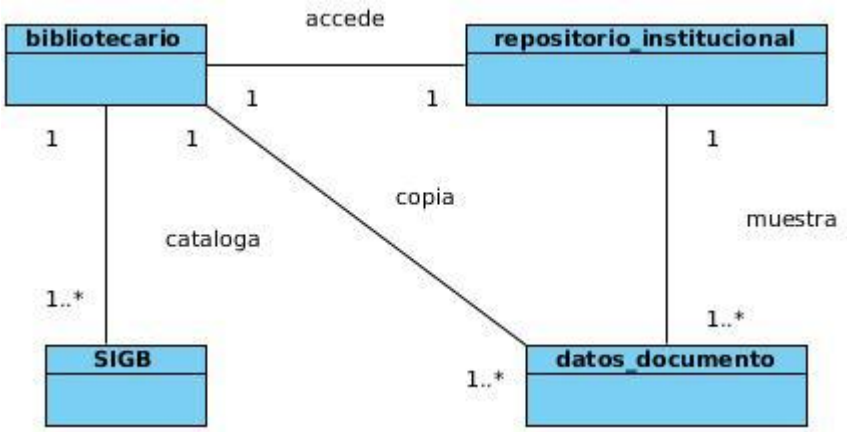


Ilustración 1. Diagrama de clases del modelo de dominio.

Actualmente el proceso de catalogación de los datos del repositorio institucional hacia el SIGB de la UCI ocurre de la siguiente forma, el bibliotecario accede por un navegador web al repositorio institucional donde busca el documento que desea catalogar, de ahí copia los datos del recurso deseado con los cuales cataloga el documento en el SIGB.

2.2.1 Glosario de términos del dominio.

Bibliotecario: persona encargada de catalogar la documentación de la biblioteca.
SIGB: es el sistema integrado donde se gestiona todo el contenido y la información disponible para la biblioteca.
Repositorio Institucional: sistema donde se encuentra toda la información científica de la UCI.
Datos_documento: se trata de los datos de los archivos contenidos por el sistema.

2.3 Especificación de los requisitos de software

Los requerimientos de *software* son capacidades o condiciones que tienen que ser alcanzadas o poseídas por un sistema o componente de un sistema para satisfacer un contrato, estándar u otro documento impuesto formalmente. Define las funciones de qué es lo que el sistema será capaz de realizar, para lo

cual se identifican las funcionalidades requeridas y las restricciones que se imponen. Son el conjunto de propiedades que debe cumplir el *software* para ser exitoso en el entorno en el cual se usará. Estos deben ser comprensibles por clientes, usuarios y desarrolladores, deben tener una sola interpretación y estar definidos en forma medible y verificable (27).

2.3.1 Requerimientos funcionales

Los requisitos funcionales son capacidades o condiciones con las cual el sistema debe cumplir e indican su comportamiento, es decir son las cosas que el sistema puede hacer. Cuando se expresan como requerimientos del usuario, habitualmente se describen de forma general mientras que los requerimientos funcionales del sistema describen con detalle la función de este, sus entradas y salidas, excepciones. A continuación se listan y describen los requisitos funcionales que debe cumplir el sistema para la solución planteada.

- RF1: Agregar proveedor.
  Descripción: Permite al administrador agregar un proveedor de datos.
- RF2: Eliminar proveedor.
  Descripción: Permite al administrador eliminar un proveedor de datos.
- RF3: Modificar proveedor.
  Descripción: Permite al administrador modificar un proveedor de datos.
- RF4 Listar proveedores
  Descripción: Permite al administrador ver los proveedores de datos.
- RF5: Recolectar metadatos de repositorio completo.
  Descripción: Permite al administrador recolectar todos los datos de un repositorio, permite especificar rangos de fecha.
- RF6: Recolectar metadatos por colección.
  Descripción: Permite al administrador recolectar los datos de una colección dada, permite especificar rangos de fecha.
- RF7: Obtener información de proveedor de datos
  Descripción: Permite al bibliotecario ver los datos del repositorio que seleccione.
- RF8: Mostrar las colecciones de un repositorio.
  Descripción: Permite al bibliotecario ver las colecciones que posee un repositorio determinado.

- RF9: Buscar registros a catalogar.
  Descripción: Permite al bibliotecario buscar los registros recolectados a catalogar.
- RF10: Catalogar registros recolectados.
  Descripción: Permite al bibliotecario catalogar los registros recolectados.

2.3.2 Técnicas de validación de requisitos

La validación de requisitos tiene como misión, demostrar que la definición de los requisitos es lo que realmente el usuario necesita o el cliente desea. Luego de realizar la captura de requisitos se procede a la validación de estos. La validación de requisitos tiene gran importancia, ya que los errores en el documento de requisitos pueden conducir a grandes costos, como repetir el trabajo cuando son descubiertos durante el desarrollo o después de que el sistema esté en uso.

Para la validar los requisitos de este trabajo, se utilizó la técnica de generación de casos de prueba en ella los requerimientos deben poder probarse. Si las pruebas para estos se conciben como parte del proceso de validación, a menudo revela problemas en los requerimientos. Para esta investigación, fueron realizados diseños de casos de pruebas a todos los requerimientos del sistema. Los cuales arrojan diferentes tipos de errores; como de validación, ortográficos y de interfaz.

2.3.3 Requerimientos no funcionales

“Los requisitos no funcionales son los requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de este, como la fiabilidad y el tiempo de respuesta” (27). Es decir los requisitos no funcionales representan aquellos atributos que debe poseer el sistema pero que no son funcionalidad específica. Los requerimientos no funcionales que debe tener el módulo son:

Usabilidad:

- RNF1: El sistema podrá ser utilizado con el fin de cosechar metadatos desde cualquier proveedor de datos que cumpla con las especificaciones que establece el protocolo OAI-PMH.

Soporte

- RNF2: Se debe implementar el sistema siguiendo el estándar de codificación.
- RNF3: Los componentes de *software* que integran la solución se organizarán de forma modular.

Seguridad

- RNF4: Para acceder al módulo los usuarios deben de estar autenticados.

Portabilidad

- RNF5: El sistema debe ser capaz de ejecutarse sobre plataforma Linux.

Legales

- RNF6: Las herramientas seleccionadas para el desarrollo del módulo están respaldadas por licencias libres o GPL.

2.3.3 Listado de casos de uso del sistema

- CU_1 Gestionar proveedores.
- CU_2 Recolectar metadatos de repositorio completo.
- CU_3 Recolectar metadatos por colección.
- CU_4 Obtener información del proveedor de datos.
- CU_5 Comprobar proveedor de datos.
- CU_6 Buscar registros a catalogar.
- CU_7 Catalogar registros recolectados.

2.4. Modelo de casos de uso del sistema

El diagrama de casos de uso es un modelo del sistema que contiene actores, casos de uso y sus relaciones. El diagrama de casos de uso describe lo que hace el sistema para cada tipo de usuario y permite que los desarrolladores del sistema y los clientes lleguen a un acuerdo sobre los requisitos, es decir, sobre las condiciones y posibilidades que debe cumplir el sistema (27).

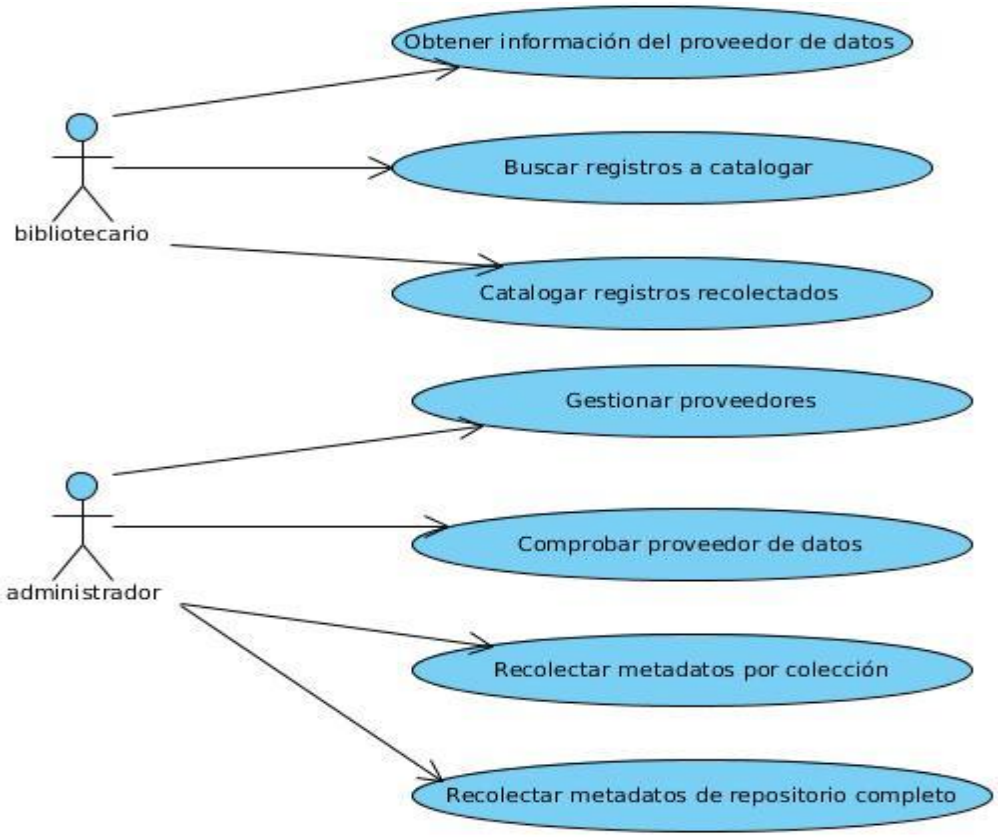


Ilustración 2. Diagrama de casos de uso del sistema.

2.4.1 Patrón de casos de uso

La experiencia en la utilización de casos de uso ha evolucionado en un conjunto de patrones que permiten con más precisión reflejar los requisitos reales, haciendo más fácil el trabajo con los sistemas y mucho más simple su mantenimiento. Dado un contexto y un problema a resolver, estas técnicas han mostrado ser la solución adoptada en la comunidad del desarrollo de *software*. Se presentan a modo de herramientas que permiten resolver los problemas que se les planteen a los desarrolladores de una forma ágil y sistemática. Estos patrones se enfocan hacia el diseño y las técnicas utilizadas en modelos de alta calidad y no en cómo modelar usos específicos. Utilizando estos patrones, arquitectos, analistas, ingenieros, y gerentes pueden lograr mejores resultados de forma más rápida.

Uno de estos patrones es CRUD este está presente en los casos donde se quiere realizar inserciones, eliminaciones, modificaciones y consultas a alguna entidad del sistema, su nombre es un acrónimo de las palabras en inglés *Create, Read, Update, Delete*. El patrón CRUD Completo consiste en un caso de uso

para administrar la información, nos permite modelar las diferentes operaciones para administrar una entidad de información, tales como crear, leer, cambiar y eliminar o dar de baja.

A continuación se muestra el caso de uso gestionar repositorio OAI-PMH donde se evidencia la utilización del de este patrón en la propuesta de solución.

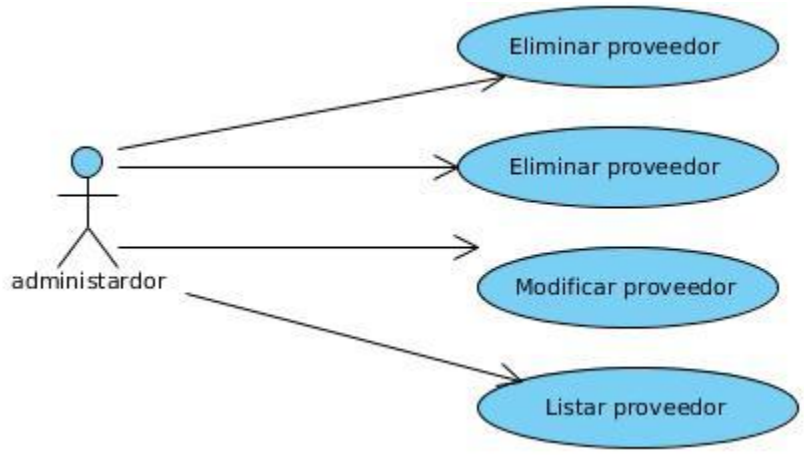


Ilustración 3. Diagrama de caso de uso gestionar proveedores antes de aplicar el CRUD completo.

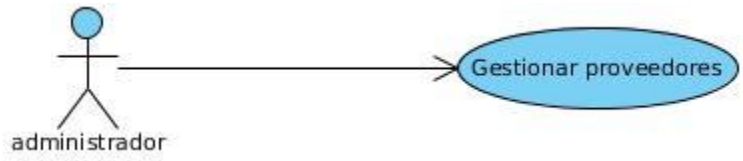


Ilustración 4. Diagrama de casos de uso gestionar proveedores después de aplicar el CRUD completo.

2.4.2 Actores del sistema

Los actores del sistema representan entidades externas que interactúan directamente con el sistema (personas, máquinas u otros sistemas) (27). A continuación se muestran los actores que interactúan con el sistema:

Actor	Descripción
Administrador	Es la persona que gestiona los proveedores de datos y realiza las peticiones OAI.
Bibliotecario	Se encarga de catalogar los registros recolectados.

Tabla 6. Actores del sistema.

2.4.3 Descripción de los casos de uso

Para entender la funcionalidad asociada a cada caso de uso no es suficiente con la representación gráfica del diagrama de casos de uso, por lo que es necesario realizar una descripción de cada uno de estos. La descripción puede ser elaborada de forma breve o extendida y debe ir acompañada del prototipo de interfaz de usuario respectivo. Los casos de uso son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario.

A continuación se realiza la descripción del caso de uso gestionar proveedores de prioridad media y la descripción de los casos de uso de recolección de prioridad crítica.

Caso de uso	Gestionar proveedores.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador selecciona la opción servidores OAI-PMHH y elige adicionar, modificar, comprobar o eliminar un proveedor de datos. El sistema solicita los datos en dependencia de la opción seleccionada, el administrador inserta un nuevo proveedor, realiza las modificaciones deseadas, elimina el proveedor y puede ver los proveedores existentes. El sistema valida y guarda los datos, finalizando así el caso de uso.
Precondiciones	El administrador debe estar autenticado.
Referencias	RF1, RF2, RF3, RF4.
Prioridad	Media.
Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. Selecciona la opción servidores OAI-PMH.	2. El sistema muestra los datos de los proveedores de datos almacenados en una tabla y

	las opciones: nuevo proveedor, editar y eliminar. a) Si selecciona la opción nuevo proveedor (ir a la sección “Nuevo proveedor OAI-PMH”). b) Si selecciona la opción editar (ir a la sección “Editar”). c) Si selecciona la opción eliminar (ir a la sección “Borrar”). d) Si selecciona la opción eliminar (ir a la sección “Comprobar”).
Ver anexo 6	
Flujo Normal de Eventos	
Sección Nuevo proveedor OAI-PMH	
Acción del actor	**Respuesta del sistema**
1.Selecciona la opción Nuevo proveedor de datos	2 Muestra una interfaz donde pide los datos.
3. Introduce los datos del nuevo proveedor.	
4 Selecciona la opción Guardar.	5. Valida los datos.
	6. Informa que se guardó el nuevo proveedor.
Ver anexo 7 y anexo 8	
Flujo alterno	
Acción del actor	**Respuesta del sistema**
	5.1 Muestra un mensaje indicando los campos errados para su corrección (Ir a la acción 2).
Sección Editar	

Acción del actor	**Respuesta del sistema**
1. Selecciona la opción Editar.	2 Muestra una interfaz donde el administrador puede cambiar los datos.
3. Edita los datos del proveedor.	
4 Selecciona la opción Guardar.	5. Valida los datos.
	6. Informa que se guardó el nuevo proveedor.
Ver anexo 9	
Flujo alterno	
Acción del actor	**Respuesta del sistema**
	5.1 Muestra un mensaje indicando los campos errados para su corrección (Ir a la acción 2).
Sección Eliminar	
Acción del actor	**Respuesta del sistema**
1. Selecciona la opción Eliminar.	2 Muestra una interfaz donde muestra el nombre del servidor y pide que confirme si desea eliminarlo.
3. Selecciona Si deseo eliminar el servidor.	4. Elimina el servidor.
	5 Muestra una interfaz informando que se eliminó el servidor.
Ver anexo 10 y anexo 11	

Sección Mostrar	
Acción del actor	**Respuesta del sistema**
	Muestra los datos de los proveedores guardados en el sistema.

Tabla 7. Descripción del caso de uso gestionar proveedores de datos.

Caso de uso	Recolectar metadatos de repositorio completo.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador decide recolectar los datos un repositorio.
Precondiciones	El administrador debe estar autenticado.
Referencias	RF5.
Prioridad	Crítica.
Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1 Selecciona el proveedor de datos.	2. El sistema lista las colecciones correspondientes al servidor seleccionado.
3. (Opcional) Introduce los rangos de fecha (desde, hasta).	5. Valida los datos.
4. Selecciona la opción Recolectar.	6. El sistema informa el éxito de la recolección.
Ver anexo 12	
Flujo alterno error de fecha	
Acción del actor	**Respuesta del sistema**
3.1 Introduce la fecha desde (mayor) a la fecha hasta.	5.1 Informa que la fecha de desde debe ser

	anterior a la fecha hasta.
3.2 Introduce la fecha hasta (menor) a la fecha desde.	5.2 Informa que la fecha de hasta debe ser mayor a la fecha desde.
Flujo alterno	
Acción del actor	**Respuesta del sistema**
	6.1 El sistema informa que no se pudo realizar la recolección (Ir a la acción 2).

Tabla 8. Descripción del caso de uso recolectar metadatos de repositorio completo.

Caso de uso	Recolectar metadatos por colección.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador decide recolectar los datos de una colección específica de un repositorio.
Precondiciones	El administrador debe estar autenticado.
Referencias	RF6.
Prioridad	Crítica.
Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. Selecciona el proveedor de datos.	2. El sistema lista las colecciones correspondientes al servidor seleccionado.
3. Selecciona la colección que desea recolectar.	
4. (Opcional) Introduce los rangos de fecha (desde, hasta).	6. Valida los datos.
5. Selecciona la opción Recolectar.	7. el sistema realiza la petición al servidor.
	8. El sistema informa el éxito de la recolección.
Ver anexo 13	

Flujo alterno error de fecha	
Acción del actor	**Respuesta del sistema**
3.1 Introduce la fecha dese (mayor) a la fecha hasta.	5.1 Informa que la fecha de desde debe ser anterior a la fecha hasta.
3.2 Introduce la fecha hasta (menor) a la fecha desde.	5.2 Informa que la fecha de hasta debe ser mayor a la fecha desde.
Flujo alterno	
Acción del actor	**Respuesta del sistema**
	8.1 El sistema informa que no se pudo realizar la recolección (Ir a la acción 2).

Tabla 9. Descripción del caso de uso recolectar metadatos por colección.

2.5 Conclusiones del capítulo

En este capítulo se especificaron las clases del dominio para lograr entender los procesos que se van llevar a cabo en la solución, se identificó al bibliotecario y al administrador como actores, además se identificaron 9 requisitos funcionales y 6 requisitos no funcionales que se utilizarán en el módulo para elaborar el proveedor de servicios en el SIGB de la UCI, los cuales son agrupados y representados mediante un diagrama de caso de uso, presentado la descripción breve de los mismos, quedando las bases sentadas para avanzar al próximo flujo de trabajo que es análisis y diseño.

Capítulo III. Análisis y diseño de la propuesta de solución

En el presente capítulo se abordará el flujo de trabajo de análisis y diseño, exponiendo a través de un grupo de artefactos cómo será llevada la solución del módulo del sistema que se propone, para su modelado se utilizan los diagramas de clases del análisis, de interacción, el diseño de clases y el diseño de la base de datos que se necesita para el almacenamiento de la información persistente.

3.1 Análisis del sistema

El análisis consiste en obtener una visión del sistema, sin tomar en cuenta el lenguaje de programación o la plataforma en que se ejecute la aplicación. Se preocupa solo de ver que hace el sistema, interesándose nada más por los requisitos funcionales, permitiendo estructurar los requisitos de manera que nos facilite su comprensión (28).

A continuación se muestran los diagramas de clases del análisis y los diagramas de colaboración para el caso de uso descrito anteriormente.

3.2 Diagrama de clases del análisis

Uno de los principales artefactos del análisis es el diagrama de clases de análisis, en él se representan los conceptos en un dominio del problema, además se representan las clases de análisis (clase interfaz, clase controladora y clase entidad) y sus relaciones entre sí (27).

Los tipos de clases utilizados en el modelo de análisis son:

- Clase Interfaz (CI): modelan las formas de interacción entre los actores y el sistema.
- Clase Controladora (CC): encapsulan el comportamiento de cada caso de uso y coordinan el trabajo de las clases interfaz y entidad.
- Clase Entidad (CE): modelan toda la información del sistema que posee una vida larga y que puede ser persistente.

A continuación se muestra el diagrama de clases del análisis definido para el caso de uso antes descrito:

Ilustración 5. Diagrama de clase del análisis: CU Gestionar proveedores de datos.

3.3 Diagrama de colaboración

Un diagrama de colaboración muestra una interacción organizada basándose en los objetos que forman parte en la interacción y los enlaces entre los mismos (en cuanto a la interacción se refiere). Proporcionan la representación principal de un escenario, ya que las colaboraciones se organizan entorno a los enlaces de unos objetos con otros (28).

A continuación se muestran los diagramas de colaboración definidos para el caso de uso gestionar proveedores.

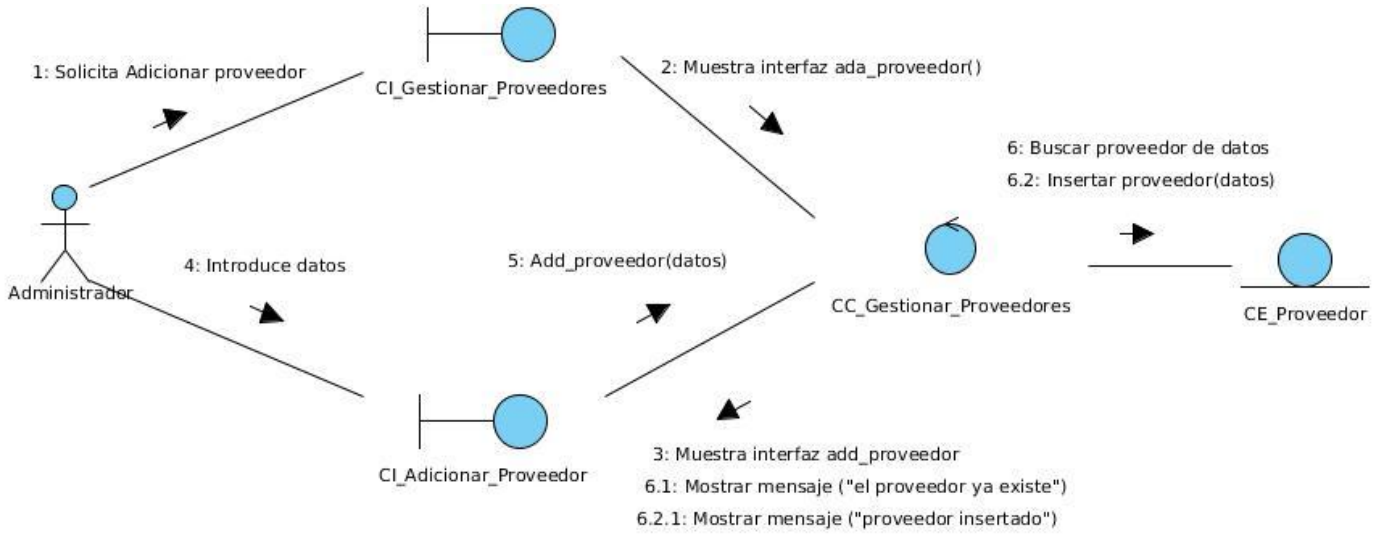


Ilustración 6. Diagrama de colaboración: Adicionar nuevo proveedor.

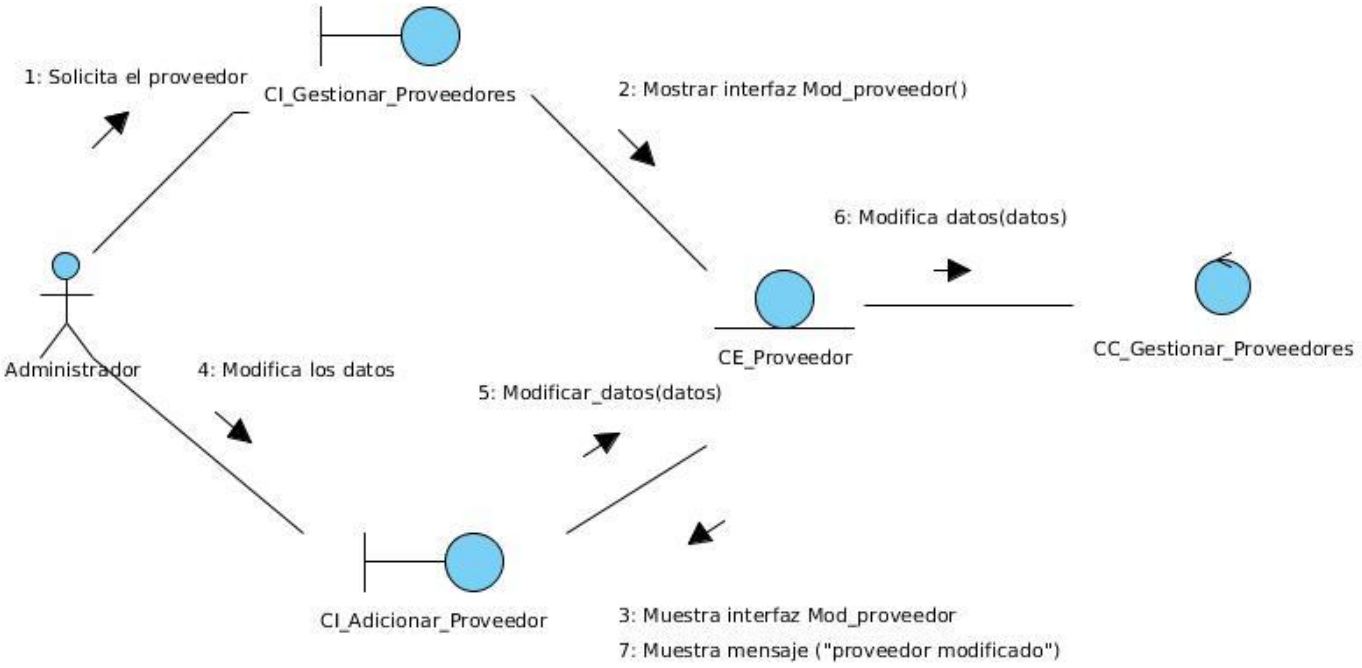


Ilustración 7. Diagrama de colaboración: Editar proveedor de datos.

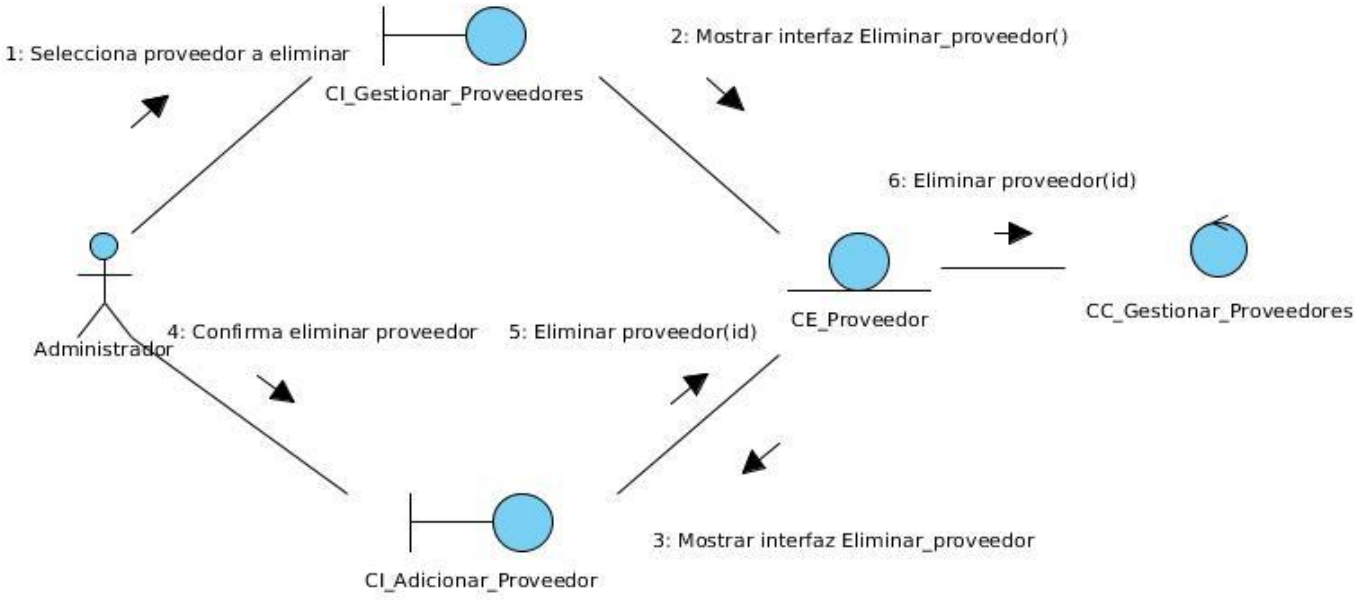


Ilustración 8. Diagrama de colaboración: Eliminar proveedor de datos.

3.4 Diseño del sistema

El modelo de diseño es un refinamiento del análisis. Un modelo de objetos que describe la realización física de los casos de uso. Se centra en los impactos que producen en el sistema a desarrollar los requerimientos funcionales y no funcionales. Es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. En este modelo los casos de uso son realizados por las clases de diseño y sus objetos (28).

3.5 Diagrama de Clases del Diseño

Los diagramas de clases del diseño describen gráficamente las especificaciones de las clases de *software* y de las interfaces en una aplicación. Normalmente, contienen clases, asociaciones y atributos, interfaces, con sus operaciones y constantes, métodos, información sobre los tipos de los atributos, navegabilidad y dependencias (27).

Los tipos de clases que se utilizarán en el diseño del presente trabajo son:

- CP_<Nombre de la página>: son las páginas que van a funcionar como interfaz a los usuarios. Se construirán dinámicamente para ser visualizadas en el explorador de los usuarios.
- Fr_<Nombre del formulario>: son los formularios que se utilizan para obtener los datos introducidos por el usuario en cada una de las actividades que se realizan durante el procesamiento de un documento.
- SP_<Nombre de la página>: son las páginas servidoras que construyen a las páginas clientes y tienen toda la lógica de presentación. Invocan todos los métodos necesarios de la capa lógica a través de las clases de servicio.

A continuación se muestran los diagramas de clases del diseño definido para el caso de uso gestionar proveedores.

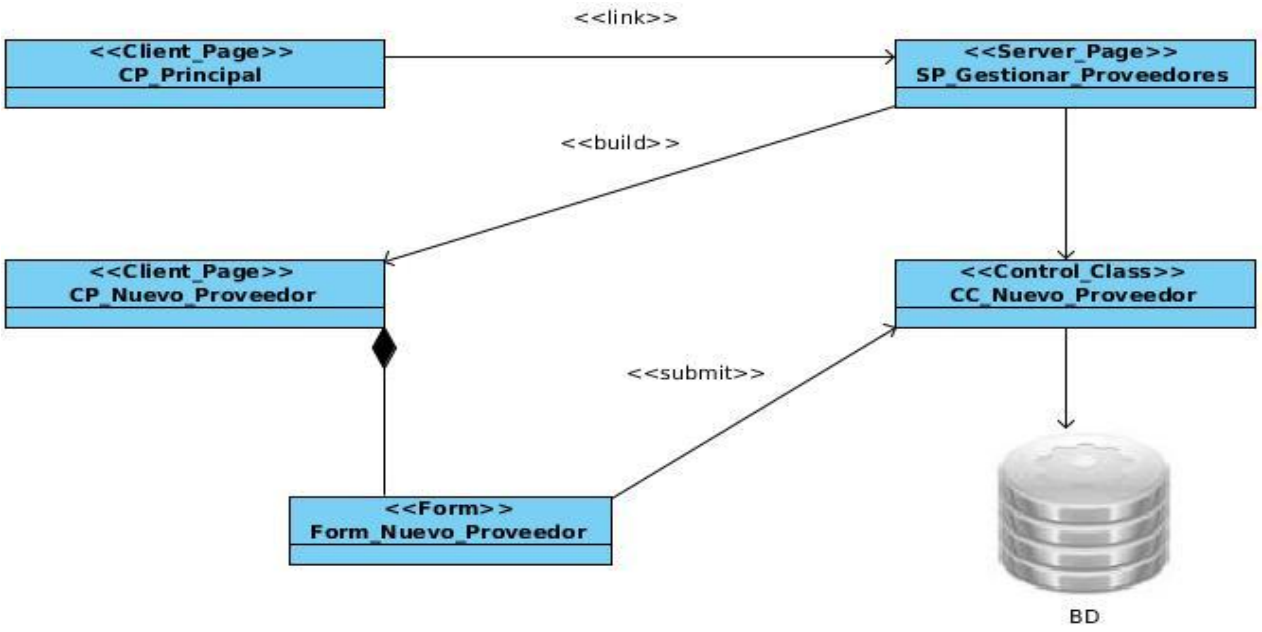
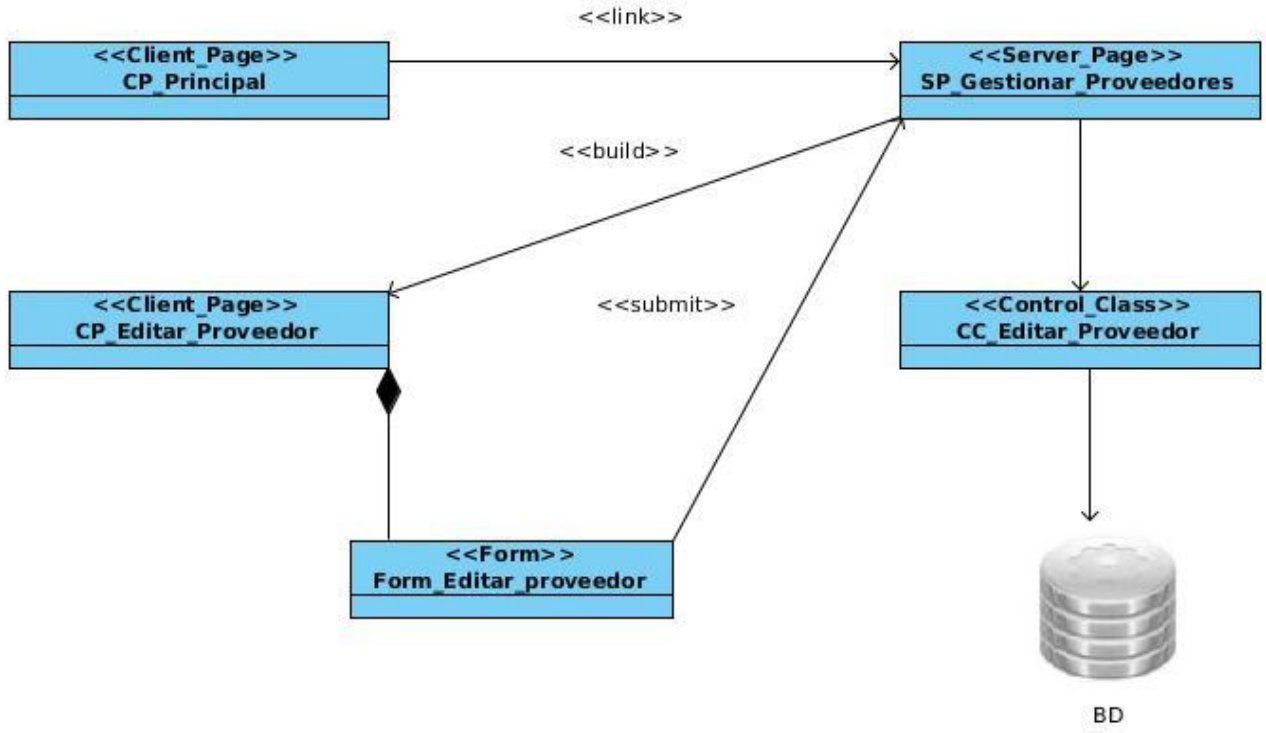


Ilustración 9. Diagrama de clases del diseño: Nuevo proveedor



.

Ilustración 10. Diagrama de clases del diseño: Editar proveedor.

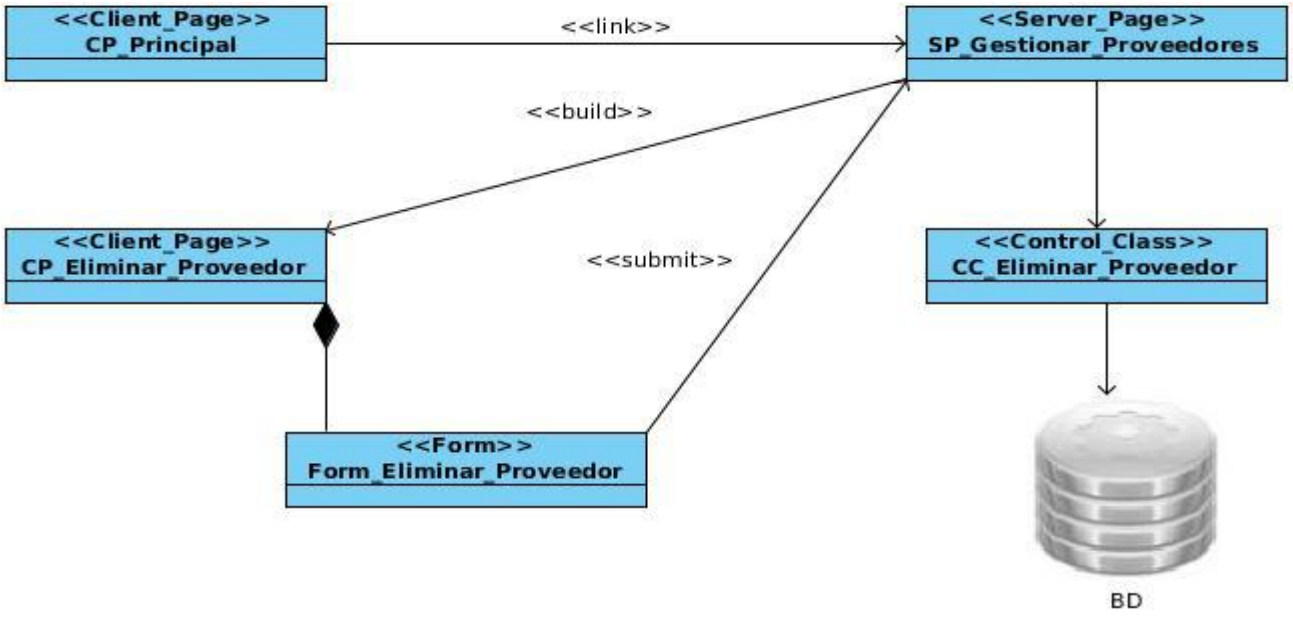


Ilustración 11. Diagrama de clases del diseño: Eliminar proveedor.

3.6 Diagrama entidad relación de base de datos

Un modelo de datos no es más que la representación de un fenómeno de la realidad objetiva a través de los objetos, sus propiedades y las relaciones que se establecen entre ellos. A continuación se explican los elementos del modelo.

- oai_servers: Almacena toda la información necesaria de los servidores (proveedores de datos).
- oai_colecciones: Almacena las colecciones con las cuenta cada proveedor.
- oai_registros: Almacena los registros de cada colección.

Para mejor comprensión el modelo de datos cuanta con los servidores (oai_servers), los cuales tienen colecciones (oai_colecciones) y las colecciones tienen registros (oai_registros).

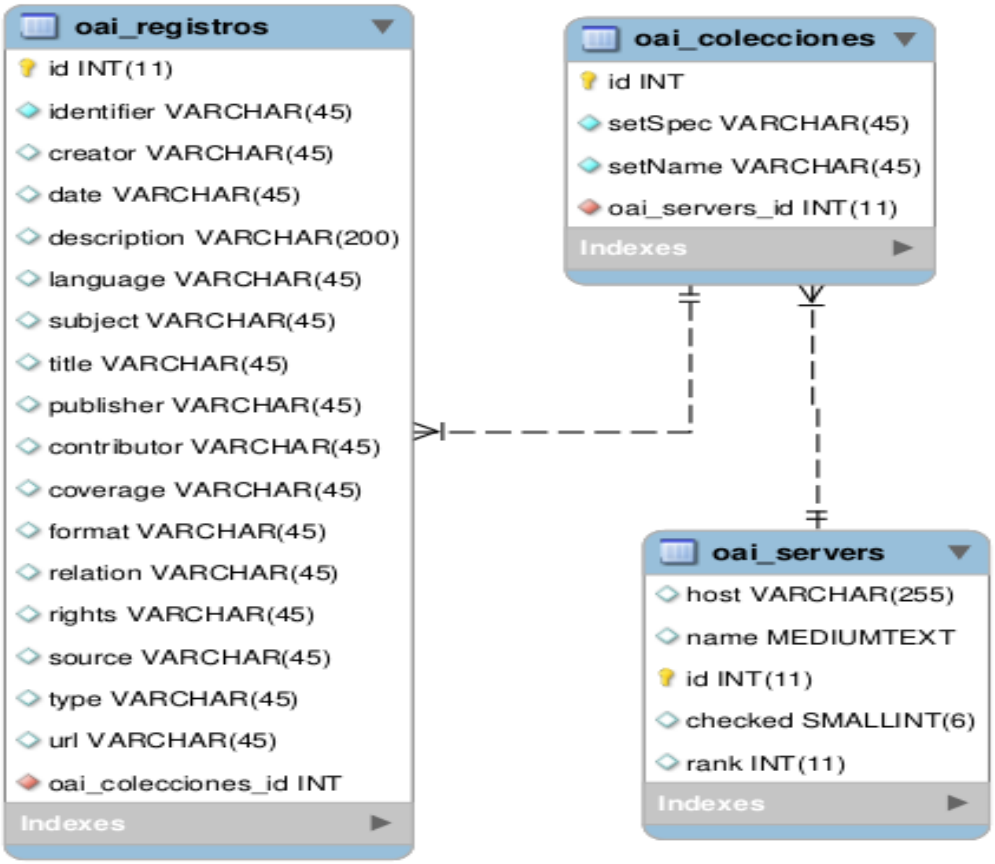


Ilustración 12. Diagrama entidad relación.

3.7 Descripción de la arquitectura

Las técnicas metodológicas desarrolladas con el fin de facilitar la programación se engloban dentro de la llamada arquitectura de *software* o arquitectura lógica. Se refiere a un grupo de abstracciones y patrones que nos brindan un esquema de referencia útil para guiar el desarrollo de *software* dentro de un sistema informático. Así, los programadores, diseñadores, ingenieros y analistas pueden trabajar bajo una línea común que les posibilite la compatibilidad necesaria para lograr el objetivo deseado (27).

Algunos tipos de arquitectura son más recomendables que otras para ciertas tecnologías, en este caso se utiliza el patrón de arquitectura Modelo Vista Controlador (MVC), que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Este patrón de arquitectura se

ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página.

3.8 Procesamiento del patrón de arquitectura MVC

El procesamiento del patrón de arquitectura MVC en el sistema se lleva a cabo entre sus tres componentes (Modelo - Vista - Controlador) de la siguiente manera:

El controlador es un fichero *script* de Perl (PL) que recibe una orden solicitada (-1-) por el cliente, este decide e invoca (-2-) quién la lleva a cabo en el modelo que es otro fichero *script* de Perl (PM). Una vez que el modelo (la lógica de negocio), hace un pedido (-3-) a la base de datos esta le retorna (-3-) los datos para que termine sus operaciones, y la modelo devuelve y envía (-4-) el flujo al controlador que procesa y envía (-5-) el resultado a la vista o capa de presentación (plantilla TMPL), que es la entregada (-6-) al cliente. Este proceso se puede evidenciar más claro en la siguiente figura:

Ilustración 13. Patrón de arquitectura.

3.8 Conclusiones del capítulo

En este capítulo se abordaron los aspectos relacionados con el análisis del módulo, desarrollándose los diagramas de clases del análisis y los diagramas de colaboración correspondientes a cada caso de uso del sistema. Se modelaron los diagramas de clases del diseño, mostrando así una propuesta para el

desarrollo del módulo. Se proporcionaron las bases necesarias para la implementación del sistema propuesto.

Capítulo IV. Implementación y prueba

Este capítulo expone lo referente a los flujos de trabajo implementación y prueba, los cuales son determinantes en el proceso de desarrollo de *software*, se muestran los diferentes artefactos que se generan en los mismo como es el modelo de diagrama de componentes, haciendo una representación de la implementación de las clases de diseño en términos de componente y cómo estos se organizan de acuerdo con los nodos específicos en el modelo de despliegue. Además, se define el método de prueba y la técnica utilizada, que se presenta y realiza en el análisis de los casos de prueba, teniendo en cuenta los datos de entrada, resultados esperados y condiciones que deben cumplirse mientras se ejecuta el caso de prueba, con el objetivo de comprobar los errores que puede tener el sistema, corregirlos y lograr obtener un óptimo funcionamiento.

4.1 Diagrama de despliegue

El diagrama de despliegue se utiliza para modelar la configuración de los elementos de procesado en tiempo de ejecución y de los componentes, procesos y objetos de *software* que viven en ellos. Se modelan los nodos físicos y las asociaciones de comunicación que existen entre ellos (27).

El diagrama de despliegue nos informa de los recursos necesarios para realizar el despliegue óptimo del sistema. A continuación se describen características de los nodos:

- Nodo PC Cliente: Representa las computadoras que utilizarán los usuarios para interactuar con la aplicación. Se comunica con el Servidor de Aplicación a través del protocolo HTTP.
- Nodo Servidor de Aplicación: Representa el servidor Apache donde se encuentra instalado el sistema.
- Nodo Servidor de Base de Datos: Es el servidor MySQL donde se almacena la base de datos el sistema.
- Nodo Servidor de LDAP: Es el servidor de dominio donde se encuentra almacenado en una base de datos todos los usuarios de la universidad.

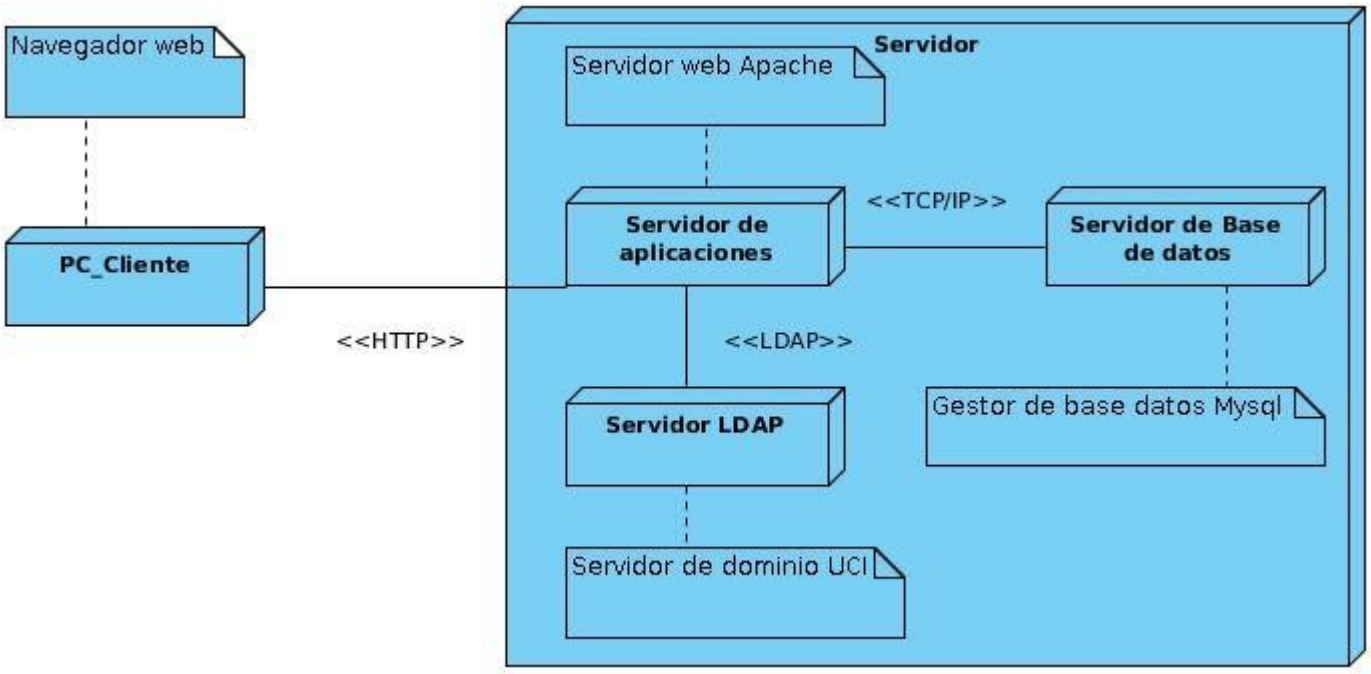


Ilustración 14. Diagrama de despliegue del sistema.

4.2 Diagrama de componentes

Los diagramas de componentes describen los elementos físicos del sistema y su relación, mostrando las dependencias lógicas entre los componentes de *software*. El diagrama de componentes hace parte de la vista física de un sistema, la cual modela la estructura de implementación de la aplicación por sí misma, su organización en componentes y su despliegue en nodos de ejecución. La vista de implementación se representa con los diagramas de componentes (27).

El subsistema de la solución para el trabajo está dividido en tres subsistemas de implementación fundamentales: el subsistema de las Vistas, el subsistema de las Controladoras y el subsistema de los Modelos, estructurados de forma tal que se agrupan los *scripts* de acuerdo con el papel que desempeñan dentro del patrón arquitectónico Modelo - Vista - Controlador.

- El subsistema de las Vistas contiene los componentes necesarios para la interacción del usuario con el sistema, los cuales son manejados por el subsistema de las Controladoras.
- El subsistema de las Controladoras, es el rector de las actividades de la aplicación, este contiene los ficheros de código fuente, los cuales interactúan con los demás subsistemas coordinando las acciones del *software*.

- El subsistema de los Modelos es el encargado de la interacción con la base de datos, para de esta forma gestionar la información con la que trabaja el sistema.

A continuación se muestra el diagrama de componentes:

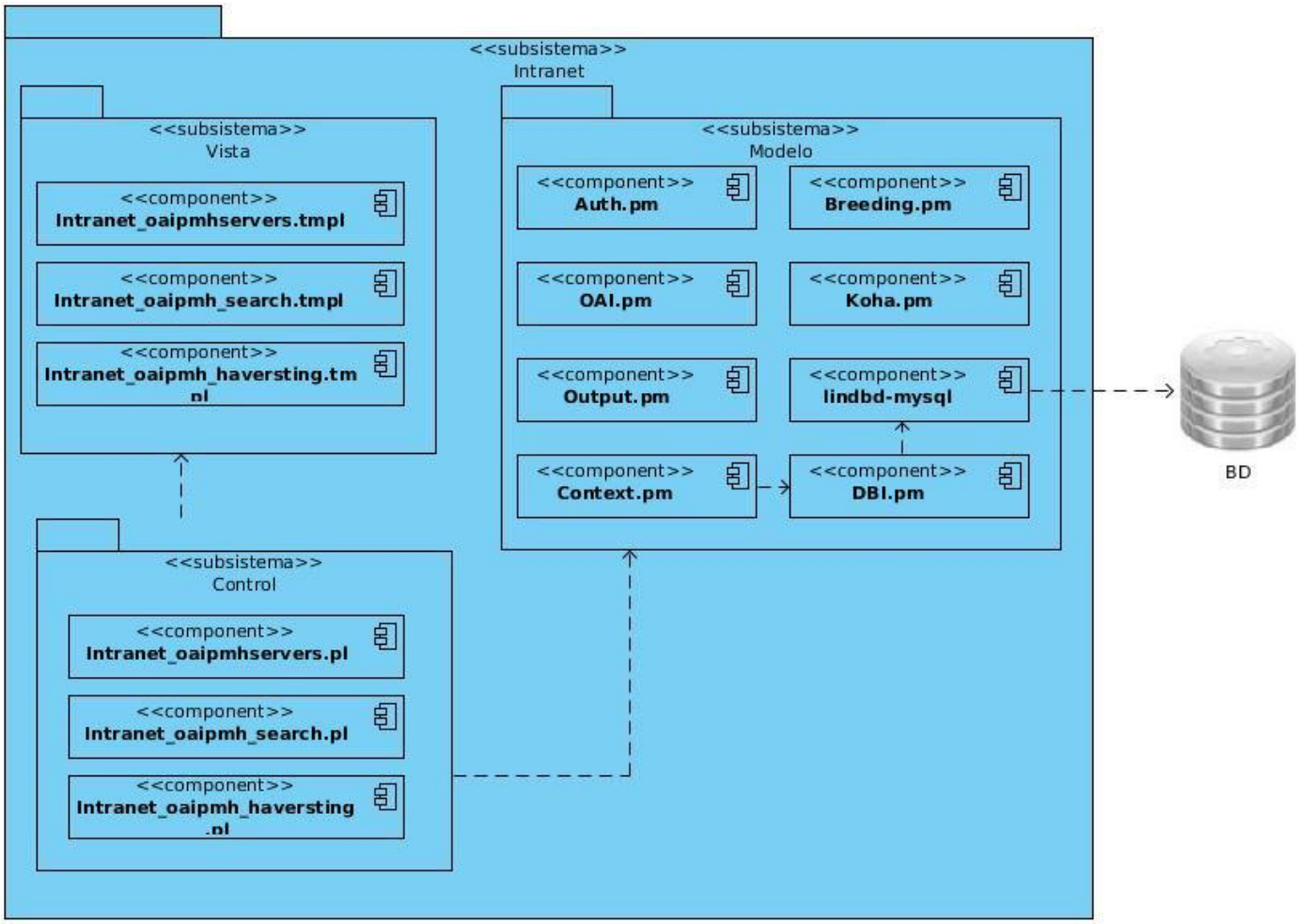


Ilustración 15. Diagrama de componentes.

4.3 Estándares de codificación

Se definen estándares de codificación porque es un estilo de programación homogéneo, esto en un proyecto permite que todos los participantes lo puedan entender en menos tiempo y que el código en consecuencia sea más entendible (29).

Lo anteriormente expuesto demuestra la necesidad de la aplicación de estándares de codificación en los proyectos de gran tamaño como es el caso del SIGB, como el SIGB está basado en Koha e implementado en Perl para la realización del módulo se siguieron los estándares ya utilizados en la aplicación, los mismos son.

Comentarios de implementación.

En el mismo se debe especificar el año en que se realizó el fichero, junto con una breve descripción de SIGB.

Comentarios de documentación

Estos comentarios le brindan al desarrollador una reseña de para que se utiliza el módulo, en el mismo se especifican aspectos importantes de la implementación.

Sentencias

Cada línea debe de tener una sola sentencia.

```
$searchstring=~ s/\'/\\\'/g;
```

Encamellado

Los nombres de las funciones deben de escribirse corridos sin dejar espacios, donde la primera letra de cada palabra debe de ser mayúscula.

```
sub CrearRecolector
```

Asignación de nombres

Cada tipo de elemento debe nombrarse con una serie de reglas determinadas.
Los PM: Deben de escribirse con mayúscula con un nombre que identifique claramente su función.
Los PL y TMPL: Deben de nombrarse con minúsculas y los que se relacionen tengan nombres afines.
admin-home.tmpl y admin-home.pl

4.4 Pruebas de software

La calidad de los sistemas informáticos se ha convertido en uno de los principales objetivos estratégicos de las organizaciones debido a que su éxito depende en gran medida de esta. Para hablar del aseguramiento de la calidad se debe también hablar de control de calidad, que son procedimientos que se establecen para poder verificar y medir la amplitud de un producto, servicio o proceso. Se puede definir

entonces que el aseguramiento de la calidad es la labor continua de garantizar que el control se aplica y el resultado es un producto de calidad (30).

- El método de pruebas de caja negra, se llevan a cabo sobre la interfaz del *software*, por lo que los casos de prueba pretenden demostrar que las funciones del *software* son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba se centra principalmente en los requisitos funcionales y examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del *software*.
- El método de pruebas de caja blanca, en este se comprueban los caminos lógicos del *software* proponiendo casos de prueba donde se ejerciten conjuntos específicos de condiciones y/o *bucles*. Además, se puede examinar el estado del programa en varios puntos para determinar si el estado real coinciden con lo esperado o mencionado.

Para comprobar la calidad de la solución planteada se le realizó las pruebas funcionales, empleándose el método de prueba de caja negra y aplicándose la técnica de partición de equivalente, siendo esta efectiva en este caso, pues permite examinar los valores válidos e inválidos de las entradas existentes en el *software*, además se descubre de forma inmediata los errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico, y con la utilización de esta técnica se permite reducir el número de clases de prueba que se deben desarrollar.

Además se realizaron pruebas de integración las cuales se realizan para comprobar que un sistema trabaja correctamente en operaciones conjuntas y es posible aplicarlas en la actual solución para verificar que el nuevo subsistema se integra con el sistema correctamente. Estas pruebas identifican problemas de interfaces entre unidades, falta de coherencia entre lo que se espera de una unidad y lo que se ofrece y hacen énfasis en la interacción y no en el funcionamiento individual. Para aplicar este tipo de pruebas fue necesario obtener los distintos valores que se envían desde la interfaz del módulo al sistema, verificar si eran correctos y comprobar la extracción satisfactoria de los datos que se manejan y que son vitales para el trabajo del nuevo subsistema.

La estrategia ascendente dentro de las pruebas de integración fue la utilizada. Esta permite ver el sistema como un árbol, se prueba desde las hojas hacia la raíz, donde las hojas son las funcionalidades básicas que se ejecutan y se requieren sus salidas para que trabaje la raíz que constituye al sistema. De la forma en que el módulo trabaja es posible realizar pruebas de integración fácilmente, los datos mostrados en la

interfaz del usuario son extraídos de la base de datos donde ya fueron guardados por otros subsistemas, por lo tanto, solo se muestra información que coincide con la entrada por el usuario, no se recogen datos nuevos para persistirlos.

A continuación se presentan y describen los casos de prueba desarrollados para cada caso de uso definido, especificando la información de entrada, los resultados obtenidos una vez ejecutado el caso de prueba y las condiciones que deben cumplirse para que este se ejecute.

Nombre de caso de prueba	Gestionar proveedores	
Entrada	**Resultados**	**Condiciones**
El administrador selecciona la opción "Nuevo servidor OAI-PMH".	El sistema muestra un formulario para introducir los datos del nuevo servidor.	
El usuario ingresa los datos dejando un campo vacio.	El sistema muestra un mensaje pidiendo que llene los campos.	El mensaje se muestra hasta que el usuario complete el campo requerido.
El usuario completa todos los campos y selecciona la opción Guardar.	El sistema muestra un mensaje informando que ha sido agregado el nuevo proveedor	
El usuario selecciona la opción Editar.	El sistema muestra un formulario para introducir los datos a modificar del servidor.	
El usuario modifica el campo "url" y selecciona la opción Guardar.	El sistema muestra un mensaje Informando que ha sido modificado el proveedor.	
El usuario selecciona la opción Eliminar.	El sistema muestra un mensaje Informando si desea eliminar el proveedor.	
El usuario confirma que desea eliminar el servidor.	El sistema muestra un mensaje Informando que ha sido eliminado el proveedor.	

Tabla 10. Caso de prueba del caso de uso gestionar proveedor.

Nombre de caso de prueba	Recolectar metadatos de repositorio completo.	
Entrada	**Resultados**	**Condiciones**
El usuario selecciona el proveedor y oprime el botón Recolectar.	El sistema muestra un mensaje informando si la recolección ha sido satisfactoria.	
El usuario oprime el botón Recolectar.	El sistema muestra un mensaje informando que debe seleccionar el proveedor.	El mensaje se muestra hasta que el usuario seleccione el campo requerido.
El usuario selecciona el proveedor y entra los parámetros de fecha (desde [opcional], hasta [opcional]).	El sistema muestra un mensaje informando si la recolección ha sido satisfactoria.	
El usuario selecciona el proveedor y entra los parámetros de fecha (desde [opcional] = > hasta, hasta [opcional]).	El sistema muestra un mensaje informando que debe ingresar las fechas correctamente.	El mensaje se muestra hasta que el usuario seleccione el campo requerido.
El usuario selecciona el proveedor y entra los parámetros de fecha (desde [opcional], hasta [opcional] = < desde).	El sistema muestra un mensaje informando que debe ingresar las fechas correctamente.	El mensaje se muestra hasta que el usuario seleccione el campo requerido.

Tabla 11. Caso de prueba del caso de uso recolectar metadatos de repositorio completo.

Nombre de caso de prueba	Recolectar metadatos por colección.	
Entrada	**Resultados**	**Condiciones**
El usuario selecciona el proveedor, la colección y oprime el botón Recolectar.	El sistema muestra un mensaje informando si la recolección ha sido satisfactoria.	
El usuario selecciona la colección y oprime el botón Recolectar.	El sistema muestra un mensaje informando que debe seleccionar el proveedor.	El mensaje se muestra hasta que el usuario seleccione el campo requerido.

El usuario oprime el botón Recolectar.	El sistema muestra un mensaje informando que debe seleccionar los criterios de búsqueda.	El mensaje se muestra hasta que el usuario seleccione el campo requerido.
El usuario selecciona el proveedor, la colección y entra los parámetros de fecha (desde [opcional], hasta [opcional]).	El sistema muestra un mensaje informando si la recolección ha sido satisfactoria.	
El usuario selecciona el proveedor, la colección y entra los parámetros de fecha (desde [opcional] = > hasta, hasta [opcional]).	El sistema muestra un mensaje informando que debe ingresar las fechas correctamente.	El mensaje se muestra hasta que el usuario seleccione el campo requerido.
El usuario selecciona el proveedor, la colección y entra los parámetros de fecha (desde [opcional], hasta [opcional] = < desde).	El sistema muestra un mensaje informando que debe ingresar las fechas correctamente.	El mensaje se muestra hasta que el usuario seleccione el campo requerido.

Tabla 12. Caso de prueba del caso de uso recolectar metadatos por colección.

Nombre de caso de prueba	Buscar registros a catalogar.	
Entrada	**Resultados**	**Condiciones**
El usuario selecciona el proveedor y oprime el botón Buscar.	El sistema muestra los registros del proveedor seleccionado.	
El usuario selecciona el proveedor, la colección y oprime el botón Buscar.	El sistema muestra los registros de la colección seleccionado.	
El usuario selecciona, la colección y oprime el botón Buscar.	El sistema muestra un mensaje informando que debe seleccionar el proveedor.	El mensaje se muestra hasta que el usuario seleccione el campo requerido.

El usuario selecciona el proveedor, e inserta el autor y oprime el botón Buscar.	El sistema muestra el registro del proveedor seleccionado.	

Tabla 13. Caso de prueba del caso de uso buscar registros a catalogar.

Nombre de caso de prueba	Catalogar registros recolectados	
Entrada	**Resultados**	**Condiciones**
El bibliotecario selecciona la opción catalogar del registro.	EL sistema muestra la vista de la catalogación con los datos del registro escritos en los campos correspondientes.	
El bibliotecario cataloga el registro.	El sistema informa que se catalogo el registro.	

Tabla 14. Caso de prueba del caso de uso catalogar registros recolectados.

Resultados de las pruebas de caja negra

Los siete casos de prueba diseñados fueron aplicados en tres iteración por parte del propio equipo de desarrollo, obteniendo un total de 9 no conformidades, de ellas tres significativas, cinco no significativas y una recomendación. Todas las no conformidades encontradas fueron corregidas y posteriormente se realizó una segunda iteración en la cual se encontraron 3 no conformidades, las cuales fueron resueltas para luego realizarse una tercera que no arrojó no conformidades. Los resultados se muestran en la siguiente matriz de trazabilidad.

Iteración	**RF1**	**RF2**	**RF3**	**RF4**	**RF5**	**RF6**	**RF7**	**RF8**	**RF9**	**RF10**
1ra iteración	x	x	x	x						
2da iteración			x		x	x	x	x		
3ra iteración					x	x			x	x

Tabla 15. Matriz de trazabilidad de los requisitos funcionales.

4.5 Conclusiones del capítulo

En este capítulo se abordaron los aspectos fundamentales de los flujos de trabajos implementación y prueba, donde se realizó y presentó la modelación del diagrama de despliegue, especificándose la distribución física de los nodos del sistema; además se presentó la modelación del diagrama de componentes representando las relaciones entre ellos. Por último en el flujo de prueba se realizaron las descripciones de los casos de prueba pertenecientes a los casos de uso documentados en el trabajo para comprobar las funcionalidades del subsistema; las pruebas realizadas se llevaron a cabo en tres iteraciones, en la primera iteración se detectaron 9 no conformidades (NC) las cuales fueron resueltas tres de ellas y dos se tomaron que no proceden; en la segunda iteración se detectaron tres NC y fueron resueltas y finalmente en la tercera no se presentaron NC, por lo tanto se puede concluir que las pruebas de funcionalidad en el subsistema fueron realizadas satisfactoriamente obteniéndose un módulo que cumple con las expectativas propuestas para que el SIGB de la UCI se comporte como un proveedor de servicios bajo el protocolo OAI-PMH.

Conclusiones

La presente investigación se centró en el desarrollo del módulo para la prestación de servicios según el estándar OAI-PMH en el SIGB, con la finalidad de facilitar el proceso de catalogación que se lleva a cabo hoy de manera casi manual en el SIGB. Durante el avance de la misma se cumplimentaron los objetivos propuestos y se arribó a las siguientes conclusiones:

- Se realizó un estudio de las posibles soluciones que permitió extraer las funcionalidades necesarias a incorporar al módulo.
- El estudio realizado como parte de la investigación sirvió de apoyo en la toma de decisiones con vista a un desarrollo eficiente del módulo.
- El módulo implementado le permite al SIGB comportarse como un proveedor de servicios bajo el protocolo OAI-PMH aumentando así su interoperabilidad.
- La implementación del módulo permite ahora al SIGB de la UCI realizar la catalogación por copia de forma rápida, eficiente y sencilla, reduciendo el tiempo y el esfuerzo físico en el proceso.
- Se demostró a partir de las diferentes iteraciones de pruebas practicadas al *software*, que este satisface los requisitos funcionales y no funcionales del módulo.

Recomendaciones

Una vez concluido el trabajo se proponen las siguientes recomendaciones:

- Permitir a usuarios que no sean de la biblioteca el acceso al módulo.
- Recolectar metadatos de repositorios reconocidos a nivel mundial y nacional.
- Incentivar a otros proyectos en la universidad a la implementación del protocolo.
- Continuar enriqueciendo la interoperabilidad del SIGB con otros protocolos.

Glosario de términos

Biblioteca: Local donde se tiene considerable número de libros ordenados para la lectura. Internet: red mundial de ordenadores interconectados basada en el protocolo TCP/IP. Ofrece distintos servicios: *e-mail*, foros, FTP, chat, compartir ficheros, videoconferencia, etc.

Código abierto: significa que todas las personas pueden acceder al código fuente, es decir, al código de la programación.

Dublin Core: Dublin Core es un modelo de metadatos elaborado y auspiciado por la DCMI (Dublin Core Metadata Initiative).

Estándar: Es lo más habitual o corriente, o que reúne las características comunes a la mayoría. Puede ser conceptualizado como la definición clara de un modelo, criterio, regla de medida o de los requisitos mínimos aceptables para la operación de procesos específicos.

HTTP (por sus siglas en inglés, Hypertext Transfer Protocol): Es un protocolo de nivel de aplicación para sistemas distribuidos. HTTP ha estado en uso por la iniciativa de la World Wide Web mundial de la información desde 1990.

IDE: Entorno de desarrollo integrado (inglés: Integrated Development Environment) es un programa compuesto por un conjunto de herramientas para un programador desde el que se pueden editar programas, compilarlos y depurarlos.

Interoperabilidad: 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.

Koha: Es un Sistema Integrado de Gestión Bibliotecaria (SIGB), que permite automatizar todos los procesos de un centro de información y documentación, como una biblioteca.

MARC21: Es un registro catalográfico legible por máquina (Machine Readable Cataloging) "Legible por máquina" significa que un tipo particular de máquina, una computadora, puede leer e interpretar los datos contenidos en un registro catalográfico.

Metadatos: Datos sobre datos o datos referentes a datos para identificar archivos digitales de conjuntos de datos científicos.

SIGB: Aplicación informática destinada a automatizar los sistemas y entornos bibliotecarios. Un Sistema Integral de Gestión Bibliotecaria se puede aplicar a las funciones y servicios propios de cualquier tipo de biblioteca, tanto públicos como de carácter técnico.

OAI (por sus siglas en inglés, Open Archives Initiative): Organización dedicada a desarrollar y promover estándares de interoperabilidad que faciliten la diseminación de contenido a través de Internet.

OAI-PMH (Open Archives Iniciative Protocol for Metadata Harvesting): framework de interoperabilidad, basado en XML, para la compartición de información entre repositorios digitales.

Protocolo: Un protocolo es un conjunto de reglas o estándares diseñados para permitir a las computadoras conectarse con otras e intercambiar información con el mínimo de error posible.

Proveedor de datos: Sistema que brinda su información bajo cierto estándar o protocolo.

Proveedor de servicio: Sistema que se conecta a los proveedores de datos para recolectar cierta información.

Referencias bibliográficas

1. **2789, Norma UNE-EN ISO.** *ISO 2789-1991.* 2012.

2. **Atkins, Daniel E.** *Santa Fe Workshop on Distributed Knowledge Work Environments.* 1997.

3. **García, Lic. Nelida Elba.** *Metadatos : necesidad e importancia de integrar estándares.*

4. **Equihua, Saúl Martínez.** *Biblioteca Digital: Conceptos, Recursos y Estándares.* 2007.

5. **McGraw-Hill.** *Diccionario de Informática e Internet de Microsoft.* 2000.

6. **Dueñas, Laureano Felipe Gómez.** *La Iniciativa de Archivos Abiertos (OAI): Un nuevo paradigma en la comunicación científica y el intercambio de información.* 2006.

7. **Hillman, Diane.** *Using Dublin Core.* 2003.

8. **Caplan, Priscilla.** *Metadata Fundamentals.* 2003.

9. **García, Lic. Nelida Elba.** 4.1. METADATOS.

10. **principales, El mundo de la Catalogación y sus reglas.** ALQUIBLA. [En línea] http://bibliotecas1978.wordpress.com/.

11. **Salinas, Aldo Guajardo.** *Z39.50 y OAI-PMH: Protocolos de Transferencia y Recuperación de.* 2010.

12. **José Manuel Barrueco, Thomas Krichel.** *Acceso a prepublicaciones en Economía RePEc.* 1999.

13. **Initiative, Open Archives.** Open Archives Initiative. [En línea] 2012. http://www.openarchives.org/.

14. **Mateo, Gema Bueno de la Fuente y David Rodríguez.** *HERRAMIENTAS DE SOFTWARE PARA OAI-PMH.* 2007.

15. **Project, Publick knowledge.** Publick knowledge Project. [En línea] 2013. http://pkp.sfu.ca/harvester/.

16. **Riopedre, Yurelkys de los Ángeles Carreras.** *Protocolos para el intercambio de información entre bibliotecas.* 2012.

17. **S. Weibel, J. Kunze y C. Lagoze.** *Dublin Core Metadata for simple resource description.* 1997.

18. **Maks, Dekkers y.** *Metadatos.* 2003.

19. **Gavilán, César Martín.** *El formato MARC: variedades geográficas y de aplicación. MARC 21.* 2008.

20. **Arcia, Heydi Alonso Martínez y Daneidys Soler.** *Desarrollo del Módulo de Reportes del Sistema Integrado de Gestión Bibliotecaria Koha para la Biblioteca Nacional de Cuba José Martí.* 2010.

21. **paradigm, Visual.** Visual paradigm. [En línea] 2012. http://www.visual-paradigm.com/.

22. **Pérez, Yusdanis Feus.** Adaptación del Módulo de Circulación del Sistema Integrado de Gestión Bibliotecaria Koha a la Biblioteca Nacional de Cuba José Martí". [En línea] 2009. http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_2380_09/1/TD_2380_09.pdf.

23. **Van Der Henst, Christian S.** Qué es el CGI. [En línea] http://www.maestrosdelweb.com/editorial/cgiintro/.

24. **Lamarca, Lapuente.** *Hipertexto, el nuevo concepto de documento en la cultura de la imagen.* 2011.

25. **Alvarez, Miguel Angel.** Qué es HTML. [En línea] 2001. http://www.desarrolloweb.com/articulos/que-es-html.html.

26. **Domínguez, Iván Vega Tabares y Yoandri Aroche.** *Estrategia de selección de Metodología de Software ágil o robusta.* 2010.

27. **Presman, Roger S.** *Ingeniería del software: un enfoque práctico.* 1998.

28. **Grady Booch, Ivar Jacobson y James Rumbaugh.** *El Proceso Unificado de Desarrollo de Software.* 2000.

29. **Calleja, Manuel Arias.** *Estándares de codificación.*

30. **Hernandez, Luna.** La importancia del aseguramiento de la calidad en los sistemas de información. *Ingeniería Primero.* [En línea] 2010. http://www.tec.url.edu.gt/boletin.

31. **Crow, Raym.** *A Guide to Institutional Repositiry Software.* 2004.

Bibliografía

ARGUETA, M. Á. G. M. LA INTEROPERABILIDAD Y EL INTERCAMBIO DE METADATOS EN LA RED 2012.

CARPERTER, LEONA. Implementando OAI-PMH. [En línea] [Disponible en: http://travesia.mcu.es/portalnb/jspui/html/10421/1823/page4.htm].

CLARA LÓPEZ GUZMÁN, F. J. G. P. *Los repositorios digitales en el ámbito universitario. Comprehensive Perl Archive Network*. Disponible en: http://www.cpan.org/.

Dublin Core/MARC/GILS Crosswalk. Disponible en: http://www.loc.gov/marc/dccross_20010312.html.

ES TRADA., S. S. *Curso de Programación en Perl.* 2007.

Explotación cooperativa de recursos educativos. Disponible en: www.relpe.org.

FUSHIMI, M. *La biblioteca como espacio de construcción y difusión de la producción científica de las instituciones académicas.*

FERNÁNDEZ, D. A. *Nuevos requisitos y funcionalidades para el Sistema Integrado de Gestión Bibliotecaria Koha-Kobli.*

FERNÁNDEZ, T. F. *DUBLIN CORE CUALIFICADO DOCUMENTO DE TRABAJO.* Salamanca.

GAVILÁN, C. M. El formato MARC: variedades geográficas y de aplicación. MARC 21. 2008.

GEMA BUENO DE LA FUENTE, D. R. M. *HERRAMIENTAS DE SOFTWARE PARA OAi-PMH.* Publicado el: 2007 de última actualización: 2007. Disponible en: http://www.rojaseberhard.com.co/rojaseberhard/bibliotecologia/lib_oai.html.

INITIATIVE, O. A. Disponible en: http://www.openarchives.org/.

JORGE, R. A. *Las iniciativas para el acceso abierto a la información científica en el contexto de la web semántica.*

MAYERLY PÉREZ VELANDIA, L. F. S. *COMO FUNCIONA EL PROTOCOLO OAI – PMH EN LA RECUPERACION DE INFORMACION.*

MSC. RICARDO CASATE FERNÁNDEZ, L. B. P. P., MSC. NÉSTOR MENA DÍAZ. *Desarrollo de un portal de recursos electrónicos de acceso abierto a partir de la base de datos bibliográfica CUBACIENCIA.*

SIZA RAMÍREZ, JUAN PABLO. Estudio, análisis, evaluación e implementación de un protocolo de intercambio de contenidos sobre internet para la interoperabilidad de repositorios de la Biblioteca Digital en la Universidad Nacional de Colombia con un proveedor de servicios de contenido. 2010.

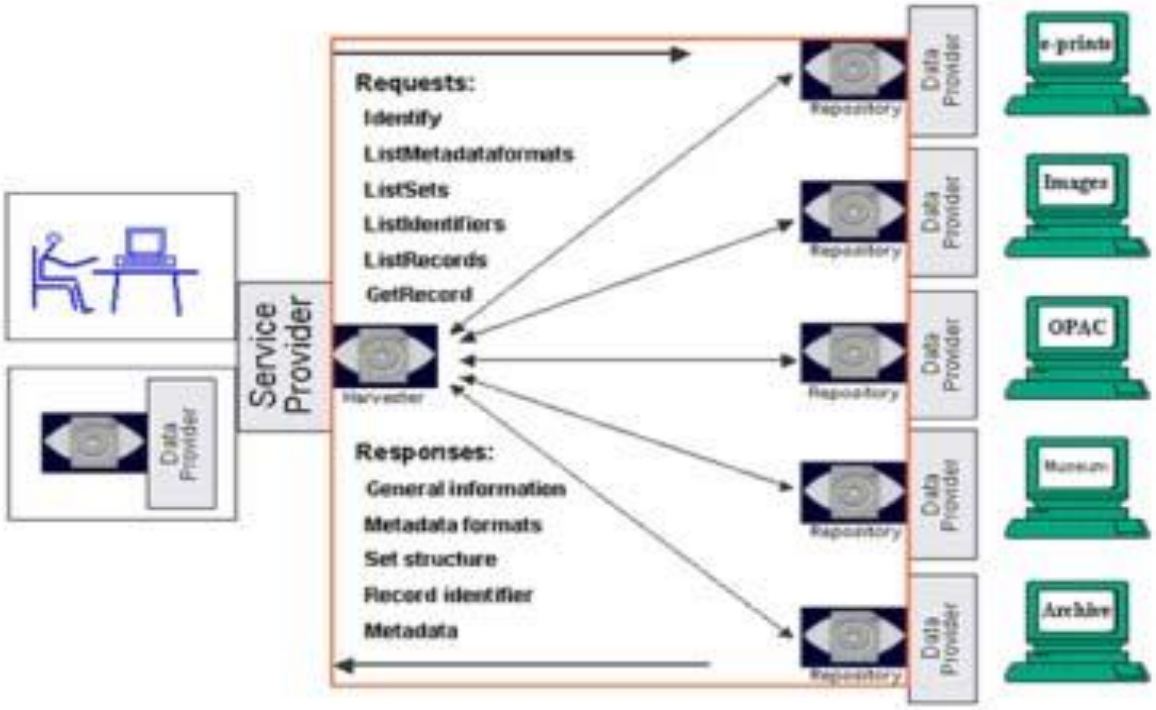
PEMAU ALONSO, JULIO y BARROSO CORROTO, JOSE ANGEL. INTRODUCCIÓN A OAI-PMH Y SU MPLANTACIÓN EN EL PORTAL E-REVISTAS.

YURELKYS DE LOS ÁNGELES CARRERAS RIOPEDRE, J. M. R. Á. Protocolos para el intercambio de información entre bibliotecas digitales. 2012.

Van de Sompel, Herbert y Lagoze, Carl. The Santa Fe Convention of the Open Archives Initiative. [En línea] 2000. [Disponible en: http://www.dlib.org/dlib/february00/vandesompel-oai/02vandesompel-oai.html].

Anexos

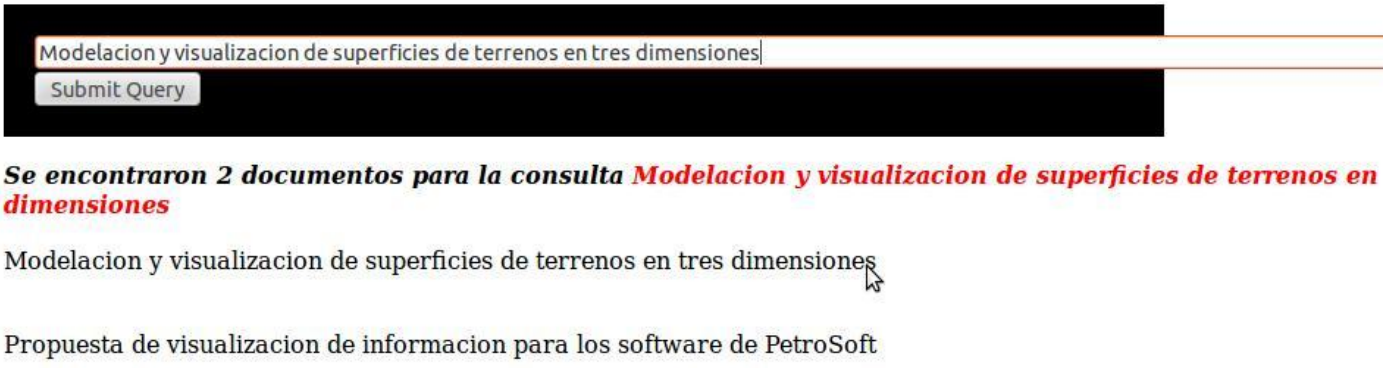
Anexo 1



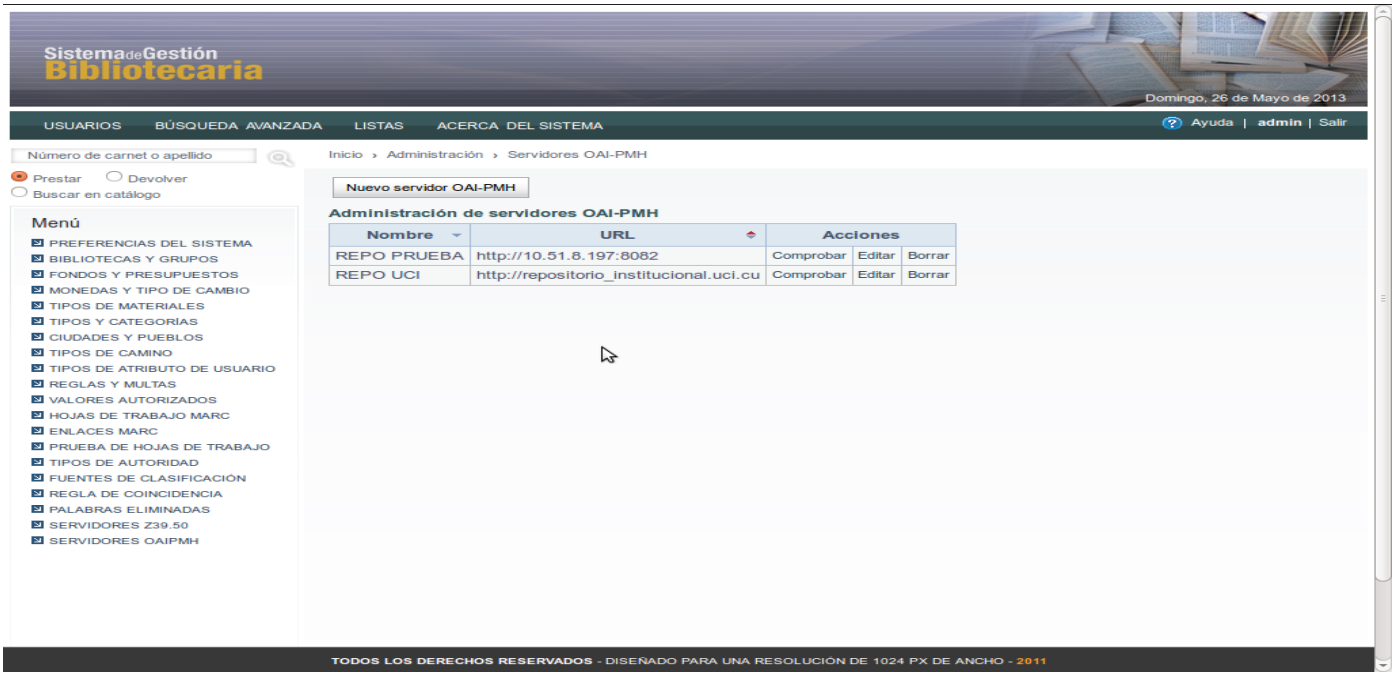
Anexo 2

Anexo 3

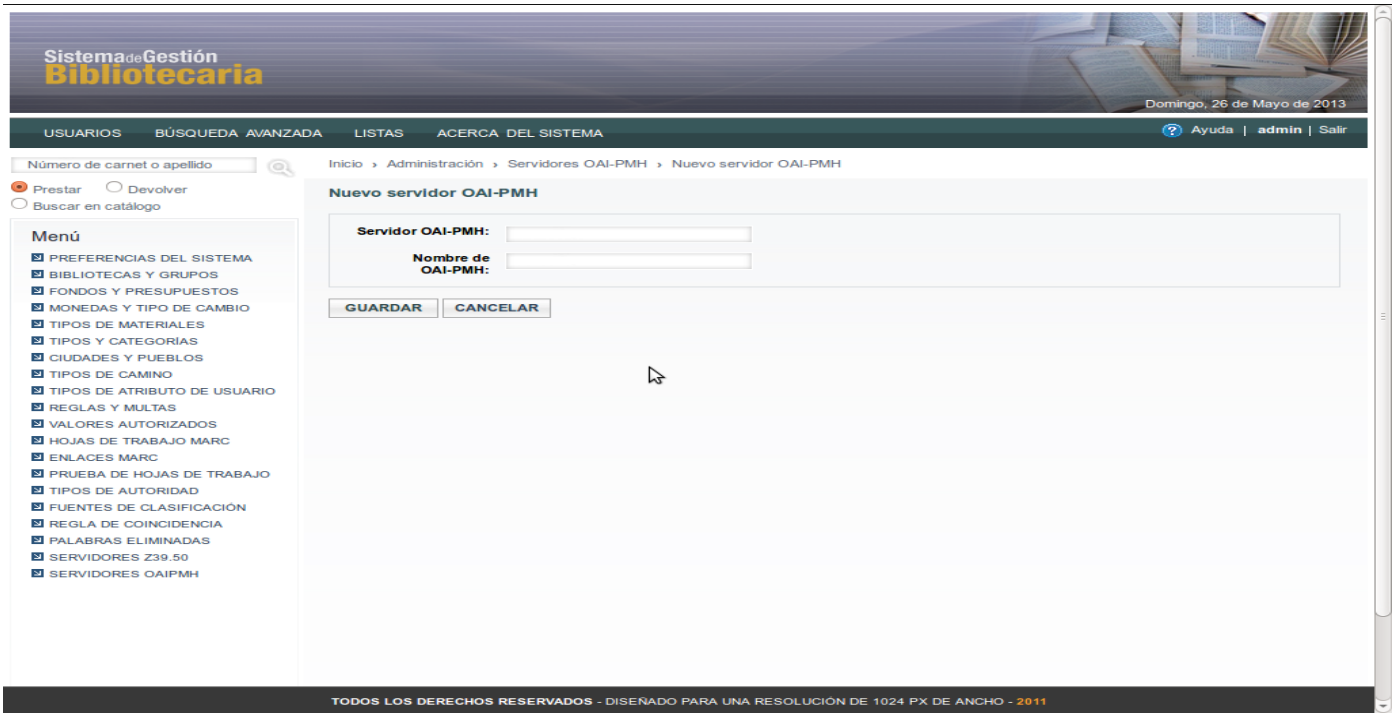
Anexo 4



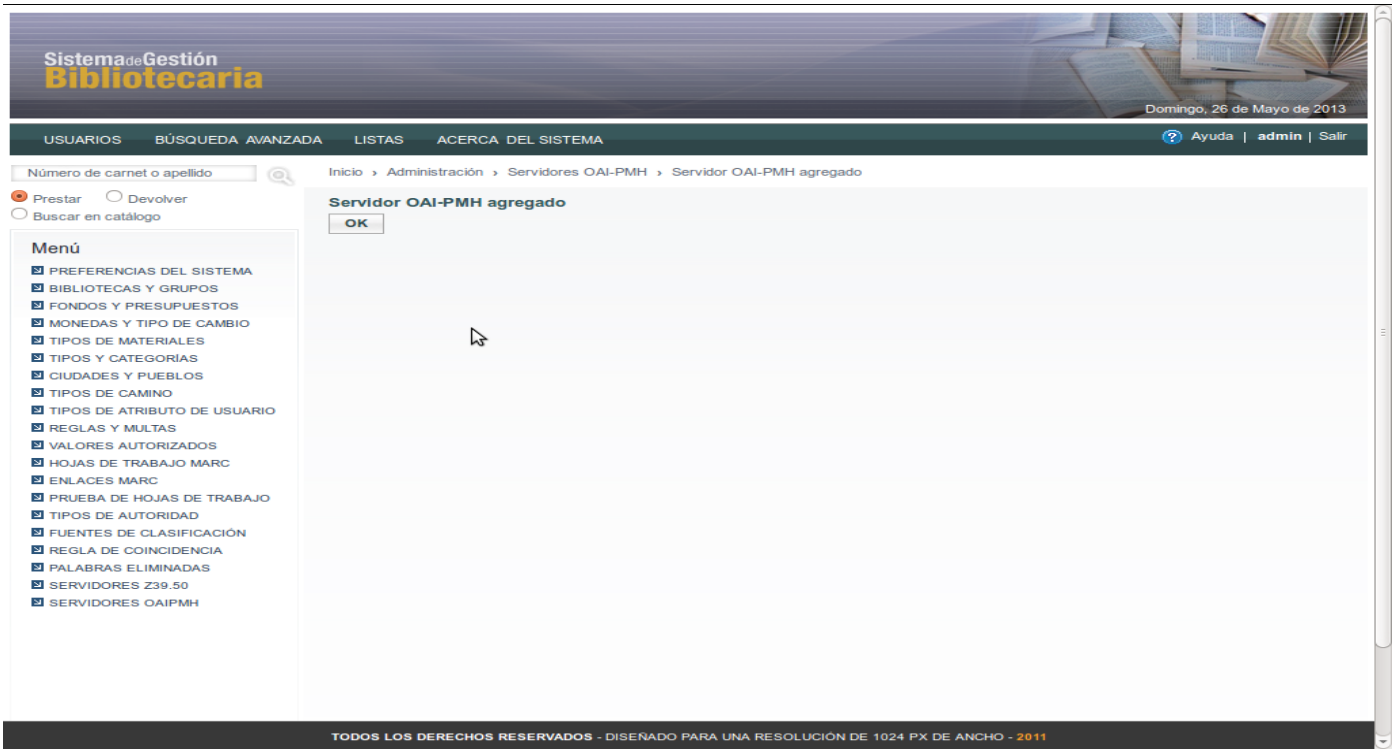
Anexo 5



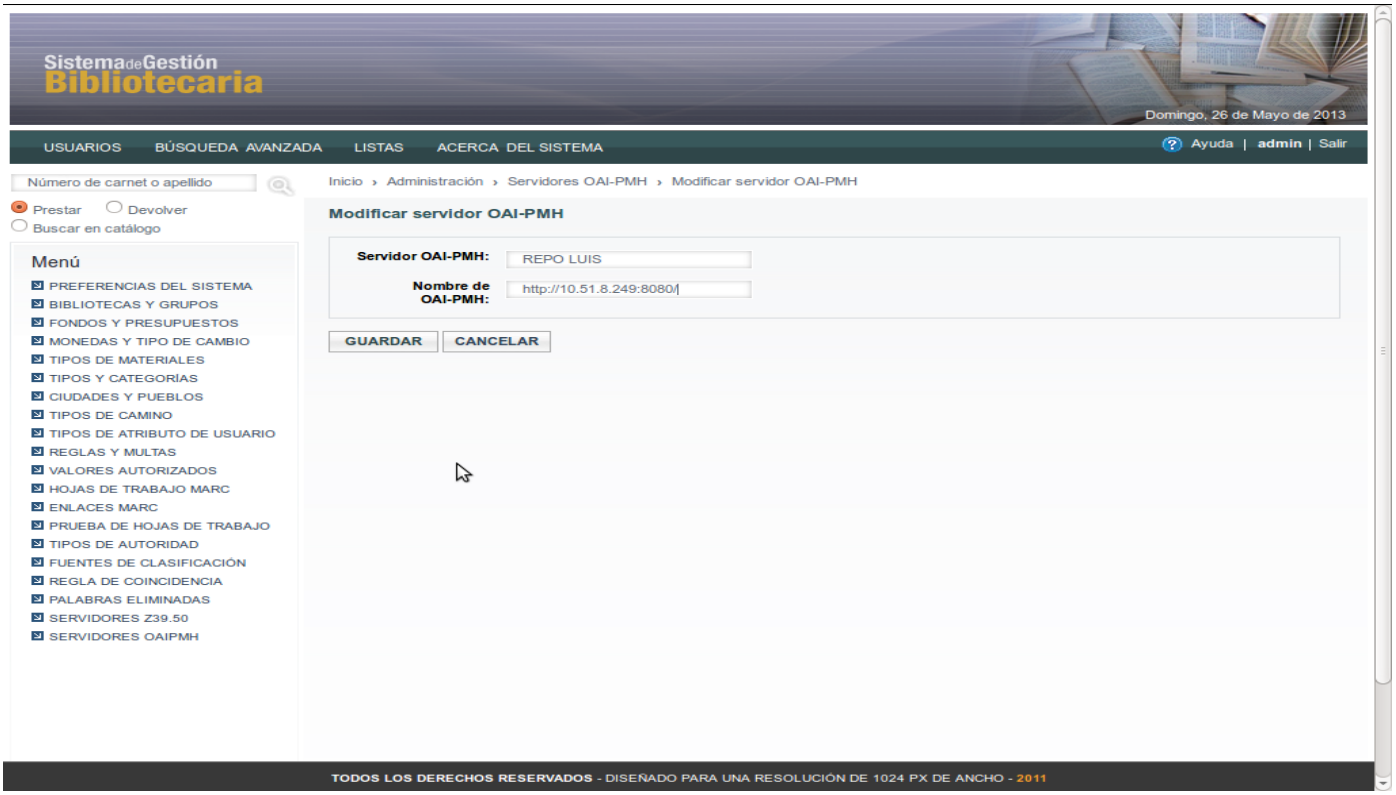
Anexo 6



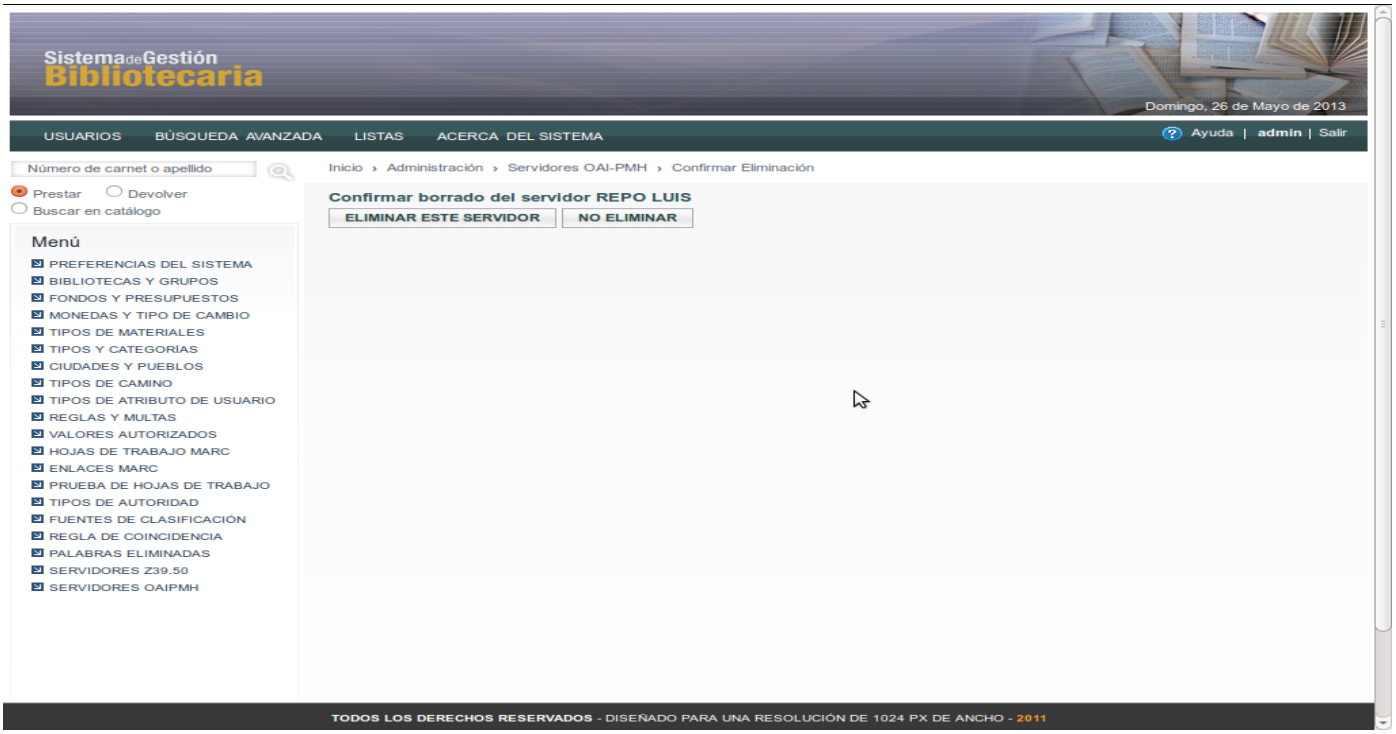
Anexo 7



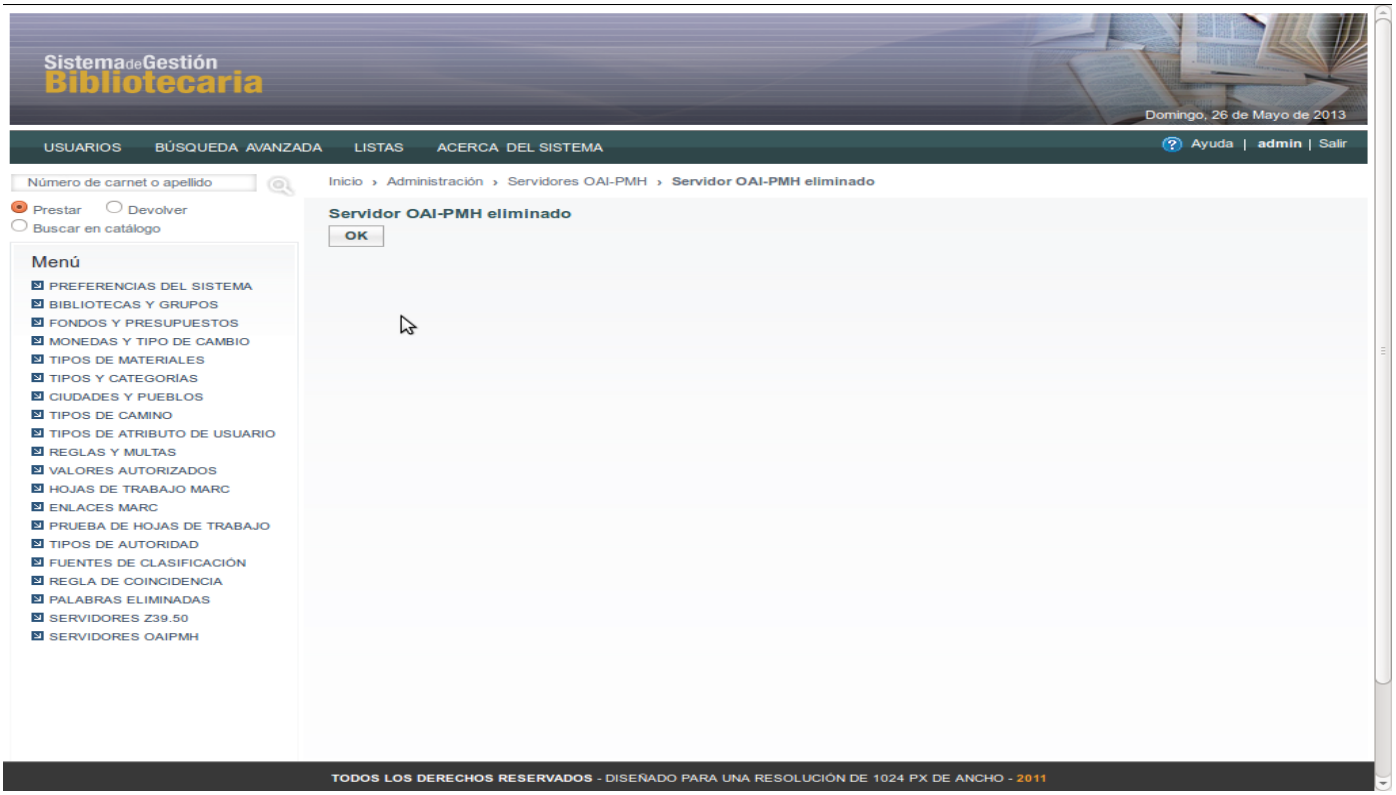
Anexo 8



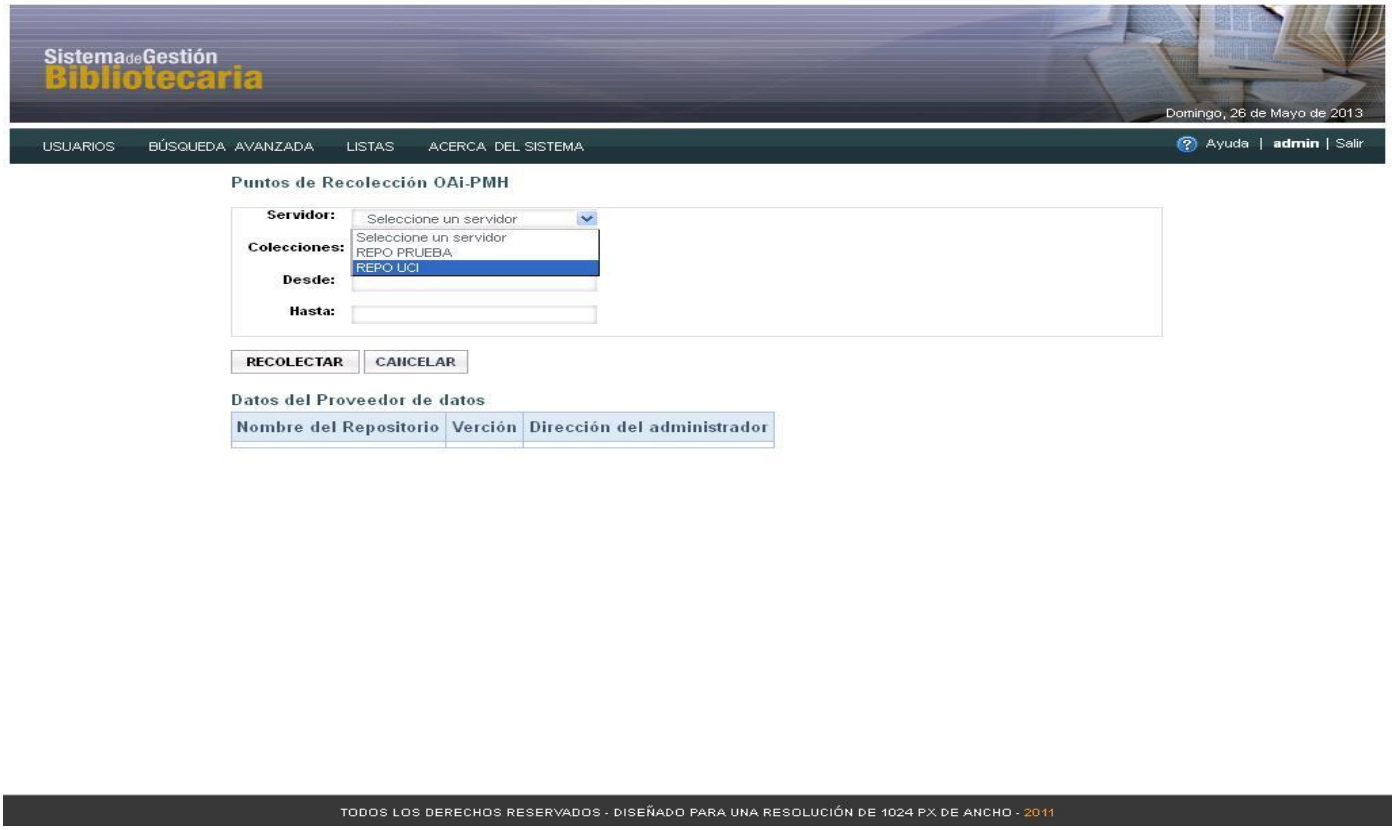
Anexo 9



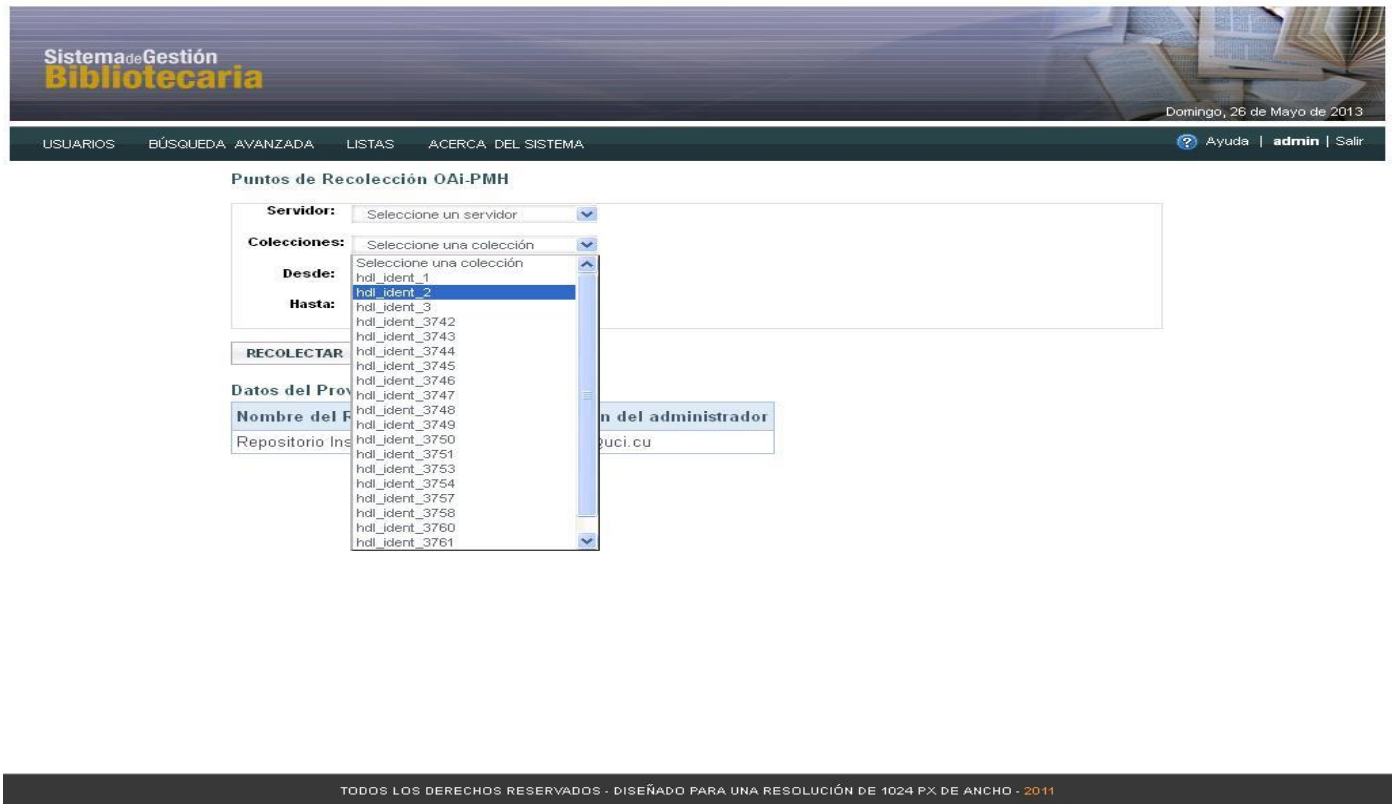
Anexo 10



Anexo 11



Anexo 12


SistemadeGestión
Bibliotecaria
Domingo, 26 de Mayo de 2013
USUARIOS
BÚSQUEDA AVANZADA
LISTAS
ACERCA DEL SISTEMA
Ayuda | admin | Salir
Puntos de Recolección OAi-PMH
Servidor:
Seleccione un servidor
Colecciones:
Seleccione una colección
Desde:
Hasta:
Seleccione una colección
hdl_ident_1
hdl_ident_2
hdl_ident_3
hdl_ident_3742
hdl_ident_3743
hdl_ident_3744
hdl_ident_3745
hdl_ident_3746
hdl_ident_3747
hdl_ident_3748
hdl_ident_3749
hdl_ident_3750
hdl_ident_3751
hdl_ident_3753
hdl_ident_3754
hdl_ident_3757
hdl_ident_3758
hdl_ident_3760
hdl_ident_3761
RECOLECTAR
Datos del Prov
Nombre del P
n del administrador
Repositorio Ins
uci.cu
TODOS LOS DERECHOS RESERVADOS - DISEÑADO PARA UNA RESOLUCIÓN DE 1024 PX DE ANCHO - 2011

Anexo 13