Universidad de las Ciencias Informáticas

PROVEEDOR DE DATOS

Trabajo de Diploma para optar por el título de
Ingeniería en Informática

Autor:

Kelvys Gálvez Cabrera

Tutor:

Ing. Luis Guzmán Hernández

Consultante:

MSc. María Caridad Valdés Rodríguez

Ciudad de la Habana, abril de 2006

"Año de la Revolución Energética en Cuba"

DECLARACIÓN DE AUTORÍA

Por este medio declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas (UCI) para que haga el uso que estimen pertinente con este trabajo.
Para que así conste firmo el presente a los __ días del mes de abril del 2006.

_____ _____

Firma de la Autor Firma de la Tutor

OPINIÓN DEL USUARIO DEL TRABAJO DE DIPLOMA

El Trabajo de Diploma, titulado: “Proveedor de Datos”, fue realizado en la Universidad de las Ciencias Informáticas. Esta Entidad considera que, en correspondencia con los objetivos trazados, el trabajo realizado le satisface:

- ☐ Totalmente
- ☐ Parcialmente en un ____ %

Los resultados de este Trabajo de Diploma le reportan a esta Universidad los beneficios siguientes:

_____
_____
_____
_____
_____
_____
_____
_____

Como resultado de la implantación de este trabajo se reportará un efecto económico que asciende a _____.

Y para que así conste, se firma la presente a los ____ días del mes de abril del año 2006.

_____ _____

Representante de la entidad Cargo

_____ _____

Firma Cuño

Gracias a la UCI y al Comandante
por formarnos como profesionales Informáticos y Revolucionarios
A la UCLV por iniciarnos en este mundo
A todos mis compañeros de aula
A los profes que siempre confiaron en nosotros, a Fabio a Matilde
A Madelin por su paciencia y dedicación
A Alain, Osmany, Marbys, Yanosky, Casa, Maykel al Ruso
A mi tutor Luís Guzmán sin el no fuera posible el éxito de la tesis
A Marilin y a Gensy que no pudieron compartir este momento
A todos muchas gracias.

A mis padres que nunca dudaron,
por su amor, su ejemplo, y dedicación
por ellos y para ellos
le dedico no la graduación sino la vida.

A Keysita
tu también puedes.

A mi familia.

RESUMEN

El crecimiento sin precedentes del mundo digital ha provocado que Internet se haya convertido en un inmenso almacén, situación que algunos expertos han comparado con una biblioteca sin catálogos. Esa excesiva disponibilidad produce un efecto perturbador en el usuario final, lo que ha motivado la necesidad de mejorar la manera de acceder a lo relevante en este vasto entorno. El protocolo de transmisión de metadatos OAI-PMH (Open Archives Initiative – Protocol for Metadata Harvesting), surge con el objetivo de mejorar el acceso a la información. El mismo define un entorno de trabajo, el cual facilita la interoperabilidad entre los proveedores de datos y los proveedores de servicios, mediante el intercambio automático de registros de metadatos. El presente trabajo está centrado en la elaboración de un sistema automatizado que cumpla con las especificaciones del proveedor de datos. Entre los objetivos fundamentales se hallan: establecer a Dublin Core como formato de los registros de los metadatos, emplear un sistema de conexión a la base de datos, que garantice el uso de los principales gestores de base de datos y crear una interfaz de administración que permita la inserción y eliminación de los registros. Como resultado se alcanza una aplicación Web, la cual atiende las solicitudes de los diferentes proveedores de servicios y como resultado le envía, en un formato XML, los metadatos solicitados.

ABSTRACT

The growth without precedents of the digital world has caused that Internet has become an immense warehouse, situation that some experts have compared with a library without catalogs. That excessive readiness produces a perturbed effect in end users, and that has motivated the necessity to improve the way to access relevant contents in this vast environment. The metadata transmission protocol OAI-PMH (Open Files Initiative-Protocol for Metadata Harvesting) surges with the objective of improving the access to information. It defines a work environment, which facilitates the interoperability between the Data Providers and the Service Providers, by means of automatic exchange of metadata registries. The present work is centered in the elaboration of an automated system that fulfills the Data Providers specifications. The main goals are: to establish a Dublin Core as metadata registries format, to use a connection method with the database that guarantees the use of the main database agents and to create an administration interface that allows the insertion and elimination of registries. As a result, a web application will be developed, which will manage the requests of different services suppliers and will answer sending the requested metadata in an XML format.

ÍNDICE

INTRODUCCIÓN 1

1 FUNDAMENTACIÓN DEL TEMA 6

1.1 INTRODUCCIÓN. 6
1.2 OPEN ARCHIVES INITIATIVE – PROTOCOL FOR METADATA HARVESTING (OAI-PMH). 6
1.2.1 Historia, de la Convención de Santa Fe a la OAI. 7
1.2.2 Flujo actual de los procesos involucrados en el campo de acción. 11
1.2.3 Proveedores de datos y de servicios. 15
1.2.4 Relación existente con otro protocolo como el Z39.50. 19
1.3 METADATOS. 20
1.3.1 Definiciones y conceptualizaciones de metadatos 20
1.3.2 Funciones de los metadatos 22
1.3.3 Diferentes tipos de metadatos 22
1.3.4 DUBLIN CORE 23
1.4 OBJETO DE ESTUDIO. 29
1.5 ANÁLISIS DE OTRA SOLUCIÓN EXISTENTE. 29
1.6 CONCLUSIONES. 30

2 TENDENCIAS Y TECNOLOGÍAS ACTUALES A CONSIDERAR. 31

2.1 INTRODUCCIÓN. 31
2.2 SOFTWARE LIBRE. 31
2.3 APLICACIONES WEB. 31
2.4 LA SEGURIDAD DE LAS TRASMISIONES EN LA WEB. 32
2.4.1 Seguridad en la transmisión Web. 33
2.4.2 Protocolo SSL. 33
2.5 SERVIDOR WEB. 34
2.6 LENGUAJES DE PROGRAMACIÓN PARA LA WEB. 34
2.6.1 Selección del lenguaje a utilizar. 37
2.7 LENGUAJE DE MARCADO EXTENSIBLE (EXTENSIBLE MARKUP LANGUAGE, XML). 37
2.8 SISTEMAS GESTORES DE BASES DE DATOS (SGBD). 38
2.9 HIBERNATE. 40
2.10 METODOLOGÍAS DE DESARROLLO DE SOFTWARE. 41
2.10.1 Selección de la metodología a utilizar. 44
2.11 PROPUESTA. 45
2.12 CONCLUSIONES. 45

3 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA 46

3.1 INTRODUCCIÓN. 46
3.2 REGLAS DEL NEGOCIO A CONSIDERAR. 46
3.3 DEFINICIÓN DE LAS ENTIDADES Y LOS CONCEPTOS PRINCIPALES 47
3.4 MODELO DE DOMINIO. 48

3.5 REQUISITOS FUNCIONALES. 49
3.6 REQUISITOS NO FUNCIONALES. 50
3.7 DESCRIPCIÓN DEL SISTEMA PROPUESTO. 51
3.8 MODELO DE CASO DE USO DEL SISTEMA. 52
3.8.1 Expansión de los Casos de Uso. *55*
3.9 CONCLUSIONES. 62
4 CONSTRUCCIÓN DE LA SOLUCIÓN PROPUESTA 63
4.1 INTRODUCCIÓN. 63
4.2 DIAGRAMAS DE CLASES. 63
4.2.1 Subsistema de cosecha de metadatos. *64*
4.2.2 Subsistema de administración. *66*
4.3 DISEÑO DE LA BASE DE DATOS. 68
4.3.1 Diagrama de clases persistentes. *68*
4.3.2 Modelo de datos *69*
4.4 PRINCIPIOS DE DISEÑO DE INTERFAZ. 69
4.4.1 Estándares de la interfaz de la aplicación. *70*
4.4.2 Tratamiento de excepciones. *71*
4.5 ESTÁNDARES DE CODIFICACIÓN. 71
4.6 MODELO DE DESPLIEGUE. 72
4.7 MODELO DE IMPLEMENTACIÓN 73
4.8 CONCLUSIONES. 75
5 ESTUDIO DE FACTIBILIDAD 76
5.1 INTRODUCCIÓN. 76
5.2 ESTIMACIÓN DE COSTO. 76
5.3 BENEFICIOS TANGIBLES E INTANGIBLES. 79
5.4 ANÁLISIS DE COSTO / BENEFICIO. 79
5.5 CONCLUSIONES. 80
CONCLUSIONES 81
RECOMENDACIONES 82
BIBLIOGRAFÍA 83
BIBLIOGRAFÍA 84
GLOSARIO DE TERMINOS 85
6 ANEXOS 87

INTRODUCCIÓN

Hoy día, con el alto crecimiento que ha experimentado Internet ha provocado que exista un gran cúmulo de conocimientos e informaciones disponibles. El gran reto es poder acceder a ese conocimiento de una manera rápida y eficiente para obtener la información relevante o específica que se pueda aplicar en la solución de problemas.

Actualmente, la mayoría de los centros de documentación cuentan con una cantidad considerable de material almacenado en formato digital, por lo que necesitan sistemas de cómputo que le permitan realizar búsquedas precisas en sus repositorios y a la vez que le faciliten la difusión de una manera correcta de sus contenidos. Incrementando de esta manera la disponibilidad, brindando la oportunidad para que cualquier organización, institución o persona con acceso a Internet pueda hacer uso de la información brindada.

Estas y otras muchas razones tuvieron los especialistas en bibliotecas digitales, en octubre de 1999, en una reunión celebrada en Santa Fe (Nuevo México, USA), donde los resultados fueron un conjunto de acuerdos técnicos y organizativos conocidos como la Convención de Santa Fe. Los aspectos técnicos incluían tres puntos fundamentales: un formato para los metadatos, un protocolo y un sistema de identificación; y no fue hasta enero de 2001, que se consumó lo planteado, con la publicación del Open Archives Initiative – Protocol for Metadata Harvesting (OAI-PMH) versión 1.0.

La Iniciativa de Archivos Abiertos (OAI), desarrolla y promueve estándares de interoperabilidad que facilita la diseminación, el intercambio y el acceso a colecciones heterogéneas de documentos científicos y académicos. Para lograrlo, se diseñó un mecanismo: el Protocol for Metadata Harvesting o Protocolo para la Recolección de Metadatos (PMH), que permite a los proveedores de información a través de una interfaz diseñada especialmente para tal efecto, poner a disposición sus metadatos.

El protocolo, define un entorno de trabajo, el cual facilita la interoperabilidad entre los proveedores de datos y los proveedores de servicios, a través del intercambio automático de registros de metadatos.

El proveedor de datos posee información (colecciones digitales) que desea compartir mediante un servicio de búsqueda. Estos proveedores, disponen su información en un depósito de metadatos, el cual contiene los registros de los metadatos que describen a los documentos almacenados en el servidor: tesis, manuscritos, fotografías y archivos de video.

El proveedor de servicios, usa una aplicación (software agent) para recolectar los registros de los metadatos de los depósitos y los combina dentro de un servicio unificado o genérico de búsqueda, disponible en una página web.

La importancia de este protocolo se puede resumir en una frase: “OAI-PMH está llamado a ser a las bibliotecas digitales lo que HTTP es hoy a la Web” [4].

La Universidad de las Ciencias Informáticas (UCI), como centro de producción de software y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación, no ha quedado exenta de los adelantos científicos-técnicos que ocurren en el mundo hoy día, por lo que el grupo de investigación y desarrollo perteneciente a la Universidad se dio a la tarea de investigar los posibles beneficios y utilidades brindadas por el protocolo de recolección de metadatos OAI-PMH.

Con el estudio detallado del estándar de interoperatividad y tras la aprobación por parte del centro, se pasó a la fase de implementación, producto del interés mostrado por diferentes organismos del país, entre ellos se encuentran la prensa y la música, los cuales planteaban que sería de mucha utilidad poseer un software que les permitiera, bajo las especificaciones del protocolo, intercambiar sus archivos vía Internet.

Para lograr un desarrollo exitoso en la implementación del software, se determinó por parte de la dirección del proyecto realizarlo en dos fases, una primera que estaría formada por la elaboración del proveedor de datos, la cual se estará abordando más adelante en este trabajo y la segunda fase donde se pasaría a la culminación de las especificaciones del protocolo, con la confección del proveedor de servicios.

El presente trabajo está centrado en la primera fase de la confección del protocolo, en la elaboración de un proveedor de dato, el cual no se realizaría en su totalidad debido a la no existencia en estos momentos de una entidad a dónde enfocar el sistema. Por lo que se

determinó confeccionar el núcleo del proveedor de dato, el cual está integrado por todas las funcionalidades exigidas por el protocolo, pero elaborado de la manera más flexible posible, quedando implementado un trabajo que se pueda reutilizar o adaptar fácilmente a las múltiples exigencias de los diferentes clientes.

El contar con una aplicación que presente todos esos servicios provocaría un cambio en la filosofía de estructurar y diseminar nuestra información, esto está dado por la forma en que se ha venido desarrollando este sector en las diferentes instituciones.

El OAI-PMH, es similar a la manera en la que normalmente funcionan los buscadores de Internet, pero difiere en varios aspectos, uno de ellos es que las búsquedas en estos sistemas se realizan en todo el documento, analizando las similitudes de palabras, y con el uso del protocolo, los usuarios solamente buscan en los registros de los metadatos que describen a los documentos. Esta diferencia significa que el recurso obtenido de los servicios de búsqueda de recolección de metadatos, es significativamente menos granular que en los servicios de texto completo.

Otro de los aspectos, es que el usuario conoce los depósitos y las características de su contenido, de manera que puede, si la interfaz de búsqueda se lo permite, seleccionar únicamente aquellos que sean de su interés.

En fin, OAI-PMH, difiere de la búsqueda convencional, en el uso de un entorno de trabajo formalizado, enfocado a la interoperabilidad y que depende exclusivamente de los metadatos.

Por tanto, el hecho de disponer de un software con estas características provocaría un gran número de beneficios tanto para la institución que desea diseminar su información como para el cliente que quiere hacer uso de ella. Esto está dado por la posibilidad de realizar búsquedas más personalizadas, en un tiempo menor, provocando un mayor intercambio, recuperación y uso de la información.

La **Hipótesis** del trabajo: si se desarrolla una aplicación capaz de realizar intercambios de metadatos con un proveedor de servicio, bajo las especificaciones del protocolo OAI-PMH, permitiría a centros de documentación una mayor diseminación de su información y a los usuarios conocer dónde se encuentra la información que necesitan.

El **objetivo general** es implementar un sistema automatizado que cumpla con las especificaciones del proveedor de datos del protocolo OAI-PMH.

Objetivos específicos:

- Establecer a Dublin Core como formato de los registros de los metadatos.
- Emplear un sistema de conexión a la base de datos, que garantice el uso de los principales gestores de base de datos.
- Crear una interfaz de administración que permita la inserción y eliminación de los registros.

Para dar cumplimiento a los objetivos planteados se realizaron las siguientes tareas:

- Estudio y descripción del protocolo Open Archives Initiative – Protocol for Metadata Harvesting.
- Estudio profundo del proveedor de datos.
- Análisis de los diferentes formatos de registros de metadatos, específicamente Dublin Core.
- Analizar la arquitectura de la aplicación.
- Estudio del framework de persistencia, Hibernate.
- Implementación del proveedor de datos.

El presente documento consta de cinco capítulos:

El **Capítulo 1** ***Fundamentación del tema****:* se explican los conceptos generales teóricos sobre los que se sustento el desarrollo de la aplicación y se formaliza el objeto de estudio.

El **Capítulo 2** ***Tendencias y tecnologías actuales a considerar***: se describe las principales tecnologías y tendencias utilizadas durante la implementación del trabajo.

El **Capítulo 3** ***Descripción de la solución propuesta****:* se plantea el modelo de dominio y se describen los casos de uso del sistema, se exponen los requisitos funciónales y los no funcionales.

El **Capítulo 4** ***Construcción de la solución propuesta***: se representa los diagramas de clases y los de secuencias, se muestra el diseño de la base de datos y los de despliegue.
El **Capítulo 5** ***Estudio de factibilidad***: se calcula el esfuerzo en hombres - mes, y se analizan los costos y beneficios que implicaría el desarrollo del trabajo.

CAPÍTULO

FUNDAMENTACIÓN DEL TEMA

1.1 Introducción.

Para lograr un desarrollo exitoso en el proceso de implementación de un software, se necesita de un amplio conocimiento de la materia que se está tratando, en la cual se apoya el analista del proyecto para diseñar la infraestructura del sistema.

El presente capítulo pretende explicar los conceptos generales teóricos, que fueron la base sobre la que se sustentó la implementación del proveedor de dato. Entre estas concepciones se encuentran la descripción del protocolo OAI-PMH, enfatizando en aspectos, como los diferentes formatos de metadatos, en especial Dublin Core.

Se analizan otros sistemas que se encuentran en explotación, que cumplen con las especificaciones del protocolo y que pudieron utilizarse como una posible solución, justificándose su no conveniencia.

1.2 Open Archives Initiative – Protocol for Metadata Harvesting (OAI-PMH).

Antes de comenzar a explicar los aspectos técnicos, se debería comprender el significado de su nombre. El término archivo refleja los orígenes de la iniciativa en el seno de las comunidades de eprints donde es sinónimo de depósito de documentos científicos a texto completo. No tiene nada que ver aquí con el concepto tradicional de archivo con connotaciones de preservación y conservación. Se utiliza por lo tanto con un sentido mucho más amplio, como un depósito para almacenar cualquier tipo de información. El término abierto se refiere al punto de vista de la arquitectura del sistema. Se tratan de definir interfaces que faciliten la disponibilidad de

contenidos procedentes de una variedad de proveedores. Apertura tampoco significa gratuidad o acceso ilimitado a dicha información.

La Open Archives Initiative (OAI) se creó con la misión de desarrollar y promover estándares de interoperabilidad para facilitar la difusión eficiente de contenidos en Internet. Surgió como un esfuerzo para mejorar el acceso a archivos de publicaciones electrónicas (eprints) y para incrementar la disponibilidad de las publicaciones científicas.

Los trabajos iniciales se centraron en el desarrollo de marcos de interoperabilidad para la federación de archivos de eprints, pronto apareció evidente que dichos marcos (permitir el intercambio de múltiples formatos bibliográficos entre distintas máquinas utilizando un protocolo común) tenían aplicaciones más allá de esta comunidad. Por ello se adoptó un objetivo mucho más amplio: abrir el acceso a un rango de materiales digitales.

Por tanto, la OAI no es solamente un proyecto centrado en publicaciones científicas, sino en la comunicación de metadatos sobre cualquier material almacenado en soporte electrónico. No hay nada en el protocolo que impida a los implementadores transmitir el contenido propiamente dicho de esos materiales. No obstante esto no es el objeto principal de OAI-PMH.

Los metadatos a transmitir vía OAI-PMH deberán codificarse al menos en Dublin Core sin calificar, con objeto de minimizar los problemas derivados de las conversiones entre múltiples formatos. Aunque se está investigando la creación de servicios tales como, una interfaz de búsqueda a través de formatos heterogéneos de metadatos, una solución menos complicada y por lo tanto más fácil de implementar es requerir a los implementadores convertir sus datos a un formato común. Los quince elementos del Dublin Core han evolucionado a lo largo de los pasados años como el estándar para los metadatos simples y multidisciplinares.[]

La OAI no define ningún esquema para la gestión de derechos. Los temas relacionados con restricciones en el acceso y gestión de la propiedad intelectual son la responsabilidad de los proveedores de datos.

1.2.1 Historia, de la Convención de Santa Fe a la OAI.

Los orígenes de OAI radican en un creciente interés en la búsqueda de alternativas a los modelos tradicionales de comunicación científica. En algunas disciplinas, principalmente en

ciencias, comenzaron a surgir los llamados archivos o repositorios de documentos electrónicos como alternativa para la rápida comunicación de resultados de investigación. Esos documentos se han llamado eprints de forma genérica. Este nuevo concepto agrupa tanto aquellos documentos que no han pasado por un proceso de certificación, como aquellos que si han sido certificados.

En Octubre de 1999 se organizó una reunión en Santa Fe (Nuevo México, USA), con la idea de que la interoperabilidad de estos archivos de eprints era clave para aumentar su impacto entre la comunidad académica. Con ella se podrían asociar varios archivos, intercambiar registros o realizar búsquedas en disciplinas relacionadas al mismo tiempo. Los participantes en esta reunión fueron especialistas en bibliotecas digitales, así como representantes de los principales archivos existentes:

- arXiv.org Considerado como el primer ejemplo de archivo de eprints. Fue fundado en 1991. Aunque comenzó como archivo de prepublicaciones ha evolucionado para incluir también artículos publicados en revistas tradicionales. Igualmente comenzó centrado en Física de Altas Energías pero ha incorporado otras disciplinas relacionadas como las Matemáticas, Informática.
- CogPrints Proyecto de la University of Southampton en el Reino Unido. Es una exportación del modelo de arXiv.org al campo de la Psicología y disciplinas relacionadas.
- NCSTRL Es la Networked Computer Science Technical Reference Library, una colección de informes y documentos en Informática. Está basado en una arquitectura distribuida en la que los documentos son almacenados en archivos distribuidos y son hechos disponibles a través de servicios que se comunican utilizando el protocolo Dienst.
- NDLTD Es la Networked Digital Library of Theses and Dissertations. Su objetivo es construir una biblioteca digital de tesis en formato electrónico, cuyos autores sean estudiantes de las instituciones miembros.
- RePEc [12] Son las siglas de Research Papers in Economics. También se basa en un modelo distribuido. Proporciona a los autores la opción de remitir sus documentos de

trabajo a un archivo local de su propia institución o, si no existe uno, al EconWPA[5], un archivo mantenido por la Washington University at Saint Louis siguiendo el modelo de arXiv.org[2]. Todos los archivos siguen el denominado Protocolo de Guildford que garantiza la interoperabilidad entre los archivos y los servicios a los usuarios finales.

La interoperabilidad de los archivos tiene varias facetas como son, por ejemplo, sistemas de identificación comunes, formatos de metadatos, modelos de documentos o protocolos. Los participantes establecieron que una solución minimalista era imprescindible si se quería alcanzar una amplia adopción entre la comunidad de proveedores de eprints. La solución adoptada fue la recolección de metadatos (metadata harvesting). Esta solución permite a los proveedores de eprints exponer sus metadatos a través de una interfaz, con el objetivo de que la misma pueda ser utilizada como la base para el desarrollo de servicios de valor añadido.

El resultado de la reunión fue un conjunto de acuerdos técnicos y organizativos conocidos como la Convención de Santa Fe. Los aspectos técnicos incluían tres puntos fundamentales: un formato para los metadatos, un protocolo basado en el antiguo Dients y un sistema de identificación.

Tras hacer públicos los resultados de la reunión, en Febrero del 2000, quedó claro que había un interés en esta iniciativa más allá de las comunidades de eprints. En principio, bibliotecarios y museólogos se mostraron interesados en descubrir formas de hacer visibles a los motores de búsqueda en Internet partes de las colecciones de bibliotecas y museos. Estas necesidades se expresaron en una serie de reuniones celebradas en el contexto de las principales jornadas sobre bibliotecas digitales, las cuales se llevaron a cabo en USA y en Europa. Respondiendo a este amplio interés se procedió a la reconsideración de las decisiones tomadas en Santa Fe. Así se decidió ampliar el objeto de trabajo más allá de los eprints, para incluir disciplinas que no tuvieran este tipo de documentación. Los aspectos técnicos aplicables exclusivamente a eprints fueron reconsiderados. Además la credibilidad del esfuerzo era incierta debido a la falta de una estructura organizacional. Los profesionales son lógicamente reacios a adoptar estándares cuando los responsables de promoción y mantenimiento de los mismos son cuestionables.

El último punto, credibilidad, fue el primero en tratarse y así en Agosto del 2000, la Digital Library Federation y la Coalition of Networked Information de los USA anunciaron que ofrecerían el soporte de su organización a la iniciativa. A partir de este momento comenzaron a funcionar dos comités, uno de gestión y otro técnico, que se encargarán de la coordinación de la iniciativa.

Las especificaciones revisadas fueron hechas públicas en Enero del 2001 con la publicación del Open Archives Initiative – Protocol for Metadata Harvesting (OAI-PMH) versión 1.0. La intención era que este protocolo, con mínimas modificaciones, permaneciera estable al menos durante un año, mientras las distintas comunidades lo probaban y experimentaban con él.

Desde ese momento la implementación del protocolo comenzó y aparecieron las primeras instituciones que lo utilizaron para poner en Internet sus metadatos. En su implementación, el OAI-PMH es una tecnología que sigue lo que Sapiro y Varian (Sapiro, 1999) denominan efectos de red, la adopción inicial es lenta y progresiva pero la respuesta positiva a la misma aumenta de forma dramática la tasa de adopción. Esto se ha cumplido en los dos años que lleva funcionando el protocolo. Ya son más de 100 las instituciones que han creado archivos abiertos, el número de servicios basados en la utilización de la información almacenada en los anteriores no ha parado de crecer tanto en número como en calidad de los valores añadidos que ofrecen. En estos momentos hay registrados en el servidor de OAI unos 12 servicios. Igualmente han aparecido toda una serie de herramientas de software destinadas a facilitar la creación y mantenimiento de archivos. Han sido muchos los proyectos de investigación que se han concedido durante los años anteriores para estudiar la aplicación del protocolo y temas relacionados. Así en USA está por ejemplo, la Metadata Harvesting Initiative de la Fundación Mellon en el seno de la que se han financiado cuatro proyectos por valor de 1.5 millones de dólares con objeto de crear servicios basados en OAI-PMH. La National Science Digital Library, un proyecto de la National Science Foudation tiene como objeto la creación de lo que será la mayor biblioteca digital hasta el momento. Ha adoptado el protocolo como base para la comunicación de metadatos entre los participantes. En Europa se han financiado proyectos por parte de la UE como por ejemplo, el Open Archives Forum, cuyo objeto es la creación de una comunidad de interés en OAI; por medio

de la organización de jornadas y actividades de soporte a la implementación de archivos y servicios.

Inmediatamente después de la difusión de la versión 1 comenzó el trabajo del comité técnico para tratar los problemas de definición o funcionalidad que se fueran descubriendo. Ese trabajo desembocó en la elaboración de la versión 2 del protocolo anunciada en Junio del 2002. Los principales cambios que se introdujeron fueron relacionados con la clarificación de ambigüedades o mejores medios para expresar las funcionalidades existentes. Es decir, no se introdujeron cambios sustanciales.

Entre los planes para el futuro están la creación de una versión SOAP (Simple Object Access Protocol)[11] del protocolo. Se espera que éste se convierta en una parte integral del trabajo en bibliotecas digitales. Otro área de interés son los formatos de metadatos, básicamente determinar si cumplen su función. El sistema utilizado actualmente (Dublin Core no calificado).

1.2.2 Flujo actual de los procesos involucrados en el campo de acción.

Los participantes en Santa Fe tomaron una decisión clave en cuanto a la arquitectura del protocolo. Adoptaron un modelo que rechazaba la búsqueda distribuida a favor de simplemente tener servidores proporcionado metadatos, sujetos sólo a criterios de alcance bastante simples, tales como, por ejemplo, proporcionar todos los registros añadidos o cambiados desde una fecha específica.

OAI-PMH utiliza transacciones HTTP para emitir preguntas y obtener respuestas entre un servidor o archivo (*data provider*) y un cliente o servicio recolector de metadatos (*service provider*).

El segundo realiza la recopilación de datos, a través de peticiones que le hace al primero, enviándole metadatos según determinados criterios, como por ejemplo la fecha de creación de los datos.

El primero responsable de la publicación y almacenamiento de los recursos en un repositorio y del mantenimiento de los metadatos, para que puedan “recolectarse” desde los proveedores de servicios, emite su respuesta, que no es más que un conjunto de registros en formato XML, incluyendo identificadores (URLs por ejemplo) de los objetos descritos en cada registro.

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

- GetRecord. Utilizado para recuperar un registro concreto. Necesita dos argumentos: identificador del registro pedido y especificación del formato bibliográfico en que se debe devolver.
- Identify. Utilizado para recuperar información sobre el servidor: nombre, versión del protocolo que utiliza, dirección del administrador, entre otras.
- 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. Recupera los registros completos. Permitiendo argumentos como el tipo de elementos y las fechas entre los que queremos recuperar los datos.
- 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.

En el ANEXO II (Fig. 6-1) se puede observar el flujo de trabajo, la forma en la que se realiza la comunicación entre los diferentes elementos del sistema. La cual se basa en una serie de verbos, antes mencionados, que forman parte del protocolo y que permiten la comunicación entre aplicaciones de forma transparente.

El protocolo soporta múltiples formatos para expresar los metadatos, no obstante requiere que todos los servidores ofrezcan los registros utilizando Dublin Core no calificado, codificado en

XML. Además de éste formato cada servidor es libre de ofrecer los registros en otro/s formatos adicionales (MARC por ejemplo). Un cliente puede pedir que los registros se le sirvan en cualquiera de los formatos soportados por el servidor. La idea subyacente aquí es que en el futuro las diferentes comunidades que utilicen el protocolo definan sus propios formatos, que sean más ricos y más precisos que el Dublin Core. Por ejemplo, la comunidad de archivos de eprints está trabajando en un formato denominado AMF (Acacemic Metadata Format) que sea capaz de describir todos los elementos que intervienen en el proceso de comunicación científica: documentos, autores, instituciones y canales de distribución de documentos.

Las respuestas del servidor estarán formateadas según el protocolo HTTP con los adecuados encabezamientos. Serán documentos XML correctos que se podrán validar contra el esquema definido en el protocolo [14]. Un ejemplo de petición y respuesta sería:

Petición:

```
http://an.oa.org/OAI-script?
 verb=GetRecord&identifier=oai:arXiv:hep-th/9901001&metadataPrefix=oai_dc
```

Respuesta:

```
<?xml version="1.0" encoding="UTF-8" ?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
    http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
 <responseDate>2002-05-01T19:20:30Z</responseDate>
 <request verb="GetRecord" identifier="oai:arXiv:hep-th/9901001"
         metadataPrefix="oai_dc">http://an.oa.org/OAI-script</request>
 <GetRecord>
   <record>
     <header>
           <identifier>oai:arXiv:cs/0112017</identifier>
   <datestamp>2001-12-14</datestamp>
```

```
   <setSpec>cs</setSpec>
   <setSpec>math</setSpec>
     </header>
<metadata>
      <oai_dc:dc
        xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/
        http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
       <dc:title>Using Structural Metadata to Localize Experience of Digital Content       </dc:title>
       <dc:creator>Dushay, Naomi</dc:creator>
       <dc:subject>Digital Libraries</dc:subject>
       <dc:description>With the increasing technical sophistication of
          both information consumers and providers, there is
          increasing demand for more meaningful experiences of digital
          information. We present a framework that separates digital
          object experience, or rendering, from digital object storage
          and manipulation, so the rendering can be tailored to
          particular communities of users.
       </dc:description>
       <dc:description>Comment: 23 pages including 2 appendices,
          8 figures</dc:description>
       <dc:date>2001-12-14</dc:date>
      </oai_dc:dc>
    </metadata>
</record>
 </GetRecord>
```

</OAI-PMH>

Aspectos que no trata el protocolo son, por ejemplo, cuestiones de gestión o autorización para el acceso de los clientes. El servidor deberá recurrir a métodos externos si desea limitar a los clientes; en relación con este punto está la utilización que estos hagan de los datos. Finalmente, tampoco trata el tema de cómo los clientes pueden localizar aquellos servidores que contengan los datos que necesitan.

1.2.3 Proveedores de datos y de servicios.

De la sección anterior se desprende que 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, con el objetivo de incorporarles algún valor añadido y presentarlo a los usuarios finales. Por tanto, si se centra en la arquitectura propia de cada uno de los elementos que componen al protocolo, se encuentra una subarquitectura para cada uno de ellos, que será necesario respetar para la optimización del sistema.

En el caso del proveedor de dato, cuando recibe una petición de metadatos procedente del proveedor de servicio, la procesa a través de una capa de aplicación programada, y realiza una consulta a su sistema de gestión de base de datos, que contiene los metadatos referentes a los recursos que se encuentran en el repositorio. Una vez hecho esto, la aplicación que ha recopilado los metadatos y que contiene una implementación del protocolo OAI-PMH, compone la respuesta en una sintaxis XML y la envía nuevamente a través de HTTP.

La capa de aplicación del Servide Provider recoge la respuesta y se encarga de introducir en su propia base de datos los metadatos recibidos para componer posteriormente los servicios de valor añadido que presentará al usuario final.

En el ANEXO II (Fig. 6-2) se describe la estructura y el flujo de trabajo del sistema aquí descrito de forma detallada, especificando el tipo de peticiones HTTP.

Los componentes que debe presentar un proveedor de metadatos son:

- Un analizador que valide los argumentos recibidos a partir de las peticiones OAI.

- Un generador de errores que cree respuestas XML para codificar los mensajes de error.
- Sistema de consulta a la base de datos o al sistema de ficheros para la recuperación de los metadatos.
- Generador de respuestas XML para la codificación de los metadatos que se enviarán como resultado de la petición de Service Provider o del Harvester.

En el ANEXO II (Fig. 6-3) se muestra el diagrama de interacción de los componentes del protocolo.

Desde Enero del 2001 se ha mantenido un registro de todos los archivos que han implementado el protocolo OAI-PMH. El registro no es obligatorio por lo que se supone que son muchos los archivos que existen y no se han registrado. Desde esa fecha el incremento del número de archivos ha sido constante llegando hasta 45 en la actualidad (sólo los que han adoptado la versión 2). Entre ellos tenemos como más destacables: arXiv.org junto con el resto de iniciativas que mencionamos en el epígrafe 1.2.1, CERN que recoge informes y prepublicaciones en el área de Física o Citebase, que proporciona datos sobre citas recibidas por los eprints almacenados en varios archivos.

Habría que destacar también los archivos que se están abriendo en el área de Biblioteconomía y Documentación. En estos momentos hay tres disponibles, aunque solamente uno de ellos está registrado:

- **@rchiveSIC** es un proyecto de colaboración entre varias instituciones francesas (universidades y centros de investigación como el CNRS). En estos momentos almacena unos 80 documentos, la mayor parte de ellos en francés. Incluye documentos de áreas relacionadas como Museología.
- **DLIST** (Digital Library of Information Science and Technology) Es un archivo creado por la School of Information Resources and Library Science y Arizona Health Sciences Library (University of Arizona). Almacena más de 100 documentos. Su objetivo es recoger todo tipo de documentos científicos en Documentación pero con dos áreas temáticas de

mayor énfasis: materiales educativos y bibliometría. Solamente aceptan documentos en inglés.

- **E-LIS** (Eprints in Library and Information Science) Es el proyecto más reciente dado que aun no se ha hecho público. Es un esfuerzo internacional para crear un archivo multinacional y multilingüe de documentos científicos en las áreas de Biblioteconomía y Documentación. Ha sido financiado parcialmente por el Ministerio de Educación.

Al igual que proveedor de datos el de servicio también presenta sus requisitos los cuales se presentan a continuación:

- Soporte de almacenamiento de metadatos: Es recomendable que los metadatos sean almacenados en una base de datos relacional a la que se pueda acceder mediante consultas SQL. También se puede utilizar un sistema de ficheros tipo LDAP o bases de datos XML.
- Soporte Web: Servidor web tipo Apache, IIS o otros.
- Mantenimiento de un API para la posterior implementación del protocolo El API debe contemplar el acceso a base de datos o al sistema de ficheros, si bien no es necesario implementar seguimiento de sesiones.

Y los componentes del proveedor de servicios son:

- Gestión del archivo: Sistema de administración web que permita dar de alta a repositorios (*Data Providers*) y gestionar la base de datos.
- Creador de peticiones: Se encarga de generar las peticiones para el proveedor de datos en función de los verbos OAI y de enviarlas a través de HTTP.
- Temporizador (*Scheduler*): Encargado de comprobar si se han hecho modificaciones en los Data Provider que forman parte de la iniciativa.
- Control de flujo (*Flow control*): Implementación para la reanudación de la señal en el caso de tener que recibir el listado de recursos de un proveedor de datos en varias veces debido a que este sea demasiado grande. El error de HTTP 503 (que significa

servicio no disponible) permite el análisis de la respuesta al mantener un periodo de reintento (*retry-after*).

- Mecanismo de actualización (*Update mechanism*): Mecanismo que se encargue de actualizar la información que se tiene en la base de datos sobre un proveedor de datos si es que el temporizador comprueba que se ha modificado.
- Analizador XML (*Parser XML*): Se encarga de analizar las respuestas extraídas de los proveedores de datos, con validación incluida.
- Herramienta de normalización (*Normaliser*): Se encarga de trasformar los distintos formatos de metadatos que se reciben de los proveedores de servicios a una estructura homogénea, armonizando su posterior presentación.
- Base de datos (*Database*): Base de datos para almacenar la información sobre los proveedores de datos.
- Comprobador de redundancias (duplication checker): Se encarga de comprobar si dos proveedores de datos distintos repiten algún recurso y lo soluciona asignando un identificador distinto al recurso en función del proveedor de servicios al que pertenezcan.
- Modulo de servicios (*Service module*): Proporciona los servicios al usuario final, que únicamente interactúa con la base de datos que se ha creado después de recopilar los metadatos de los distintos Data Providers.

La interacción de estos componentes se materializa en la figura mostrada en el ANEXO II (Fig. 6-4).

Igualmente se mantiene un registro de los servicios creados utilizando los datos proporcionados por los anteriores. Tampoco aquí es obligatorio el registro por lo que es imposible saber cuántos servicios existen realmente en la actualidad. Sin duda son muchos más que los 12 que aparecen en el registro oficial. Como más destacables:

- **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.

- **OAIster** Es un proyecto financiado por la fundación Mellon con el objetivo de crear una amplia colección de recursos digitales gratuitos, útiles y que previamente eran de muy difícil acceso y ponerla al alcance de cualquier usuario de la forma más sencilla posible. Es decir, trata de sacar a la luz colecciones que antes eran invisibles. Todos los recursos tienen el texto completo disponible en la red de forma que siempre se pueda llegar a los contenidos. Recoge datos de todos los archivos conocidos. En total 122 archivos con más de un millón de registros.
- **Perseus** Es una biblioteca digital especializada en Humanidades. Su servicio OAI también recupera datos de todos los servicios conocidos. Está financiada por la National Science Foundation en los USA.
- **Cyclades** Es un proyecto financiado por la Unión Europea. Su objetivo no está directamente relacionado con OAI ya que es crear un marco de colaboración entre los investigadores de los centros que participan en el proyecto. Intenta fomentar la colaboración entre los mismos, emitir recomendaciones y crear servicios personalizados.

En resumen, los proveedores de servicios están proliferando cada vez más y están proporcionando servicios más sofisticados. Se puede decir que existe un mercado donde los servicios pueden competir, por ejemplo existen hasta 10 interfaces diferentes a los datos proporcionados por arXiv.org, cada uno de ellas con características diferentes.

1.2.4 Relación existente con otro protocolo como el Z39.50.

El marco diseñado por OAI es intencionalmente simple con el propósito de proporcionar una mínima complicación para las instituciones que deseen implementarlo. Los protocolos como el Z39.50 tienen una funcionalidad más completa, por ejemplo, tratan cuestiones como el manejo de sesiones, gestión de conjuntos de resultados y permiten la especificación de predicados para filtrar los resultados obtenidos. Sin embargo, esta funcionalidad acarrea un incremento en la

complejidad de la implementación y, en consecuencia, de los costes. Por lo tanto no se trata de reemplazar otras iniciativas, sino desarrollar una alternativa que sea fácil de implementar y de desarrollar para propósitos diferentes de los que ya tratan los sistemas de interoperabilidad existentes. El futuro juzgará si esta barrera mínima de interoperabilidad es realista y funcional.

1.3 Metadatos.

Según Floridi, cuando se habla del espacio intelectual de la organización de la información e el conocimiento distinguimos tres diferentes dimensiones. La primera está representada por los campos ordinarios de los conjuntos de datos primarios. Es la principal información que podemos adquirir cuando tenemos acceso a los documentos. Los datos primarios son multimediales (textos, fotografías, diagramas estadísticos, base de datos de texto completo, entre otros.)

La segunda dimensión está representada por los denominados datos secundarios o metadatos (metadata). Son indicaciones que conducen a los datos de la primera dimensión, por ejemplo información de copyright, registros catalográficos entre otros.

La tercera dimensión esta representada por los datos derivativos, que pueden ser extraídos tanto de los conjuntos de datos primarios como de los secundarios, cuando los últimos son utilizados como fuentes para análisis comparativos y cuantitativos.

1.3.1 Definiciones y conceptualizaciones de metadatos

En el sentido general los metadatos, son datos referentes a datos. Por ejemplo los registros catalográficos, al igual que los registros de bases de datos podrían ser considerados como metadatos, dada su función de proporcionar información básica (autor, fecha de creación, conexiones a otras obras, entre otras) del recurso de información.

En consecuencia, desde la perspectiva bibliotecaria, la catalogación tradicional produciría como resultado un tipo de metadatos. Caplan, coincide con la idea básica establecida, y hace confluir en esta categoría, tanto un registro catalográfico tradicional con uno de naturaleza digital (por ej., Dublin Core, TEI Header) o cualquier otra forma de descripción. Para ella, metadatos significa *"un buen término neutro que cubre todas las bases"*. Acorde a esto, en la actividad bibliotecaria, se trataría de seguir catalogando recursos digitales como si nada hubiera sucedido. Burnett, desde

el contexto bibliotecológico, sostiene que metadatos es concebido como *"cualquier información que registra la caracterización y relaciones de los datos fuente, o el conjunto de elementos de datos que pueden ser utilizados para describir y representar objetos de información".*

Sin embargo, la misma autora sostiene que, desde el contexto informático, metadatos, posee una significación diferente siendo *"cualquier información que soporta la efectiva utilización de datos, incluyendo información que pueda facilitar la gestión, el acceso y el análisis de los mismos"*. Heery, argumenta que este término está siendo cada vez mas utilizado estrictamente para el entorno digital. En consecuencia, desde esta posición, los metadatos difieren de los datos catalográficos tradicionales en que la información de localización esta contenida dentro del registro permitiendo la recuperación directa del documento a partir de la aplicación de un software apropiado. Avalando y reforzando esta postura, T. Berners-Lee agrega: *"Metadatos es una información que entiende el ordenador sobre recursos web y otras cosas".* Esta definición se contextualiza en información manipulada por productores de software, diferentes de los usuarios tradicionales de bibliotecas, y en que su objetivo lo constituye la eficacia y celeridad de su funcionamiento, en comparación con la catalogación tradicional.

Mientras los esquemas tradicionales utilizados en bibliotecas (AACR2, MARC, ISBD) han sido utilizados, por décadas, para el control y acceso bibliográfico, cabe la pregunta: ¿cómo utilizarlos para catalogar los documentos disponibles en Internet? Esto ha dado surgimiento al pensamiento que los documentos digitales necesitan ser auto-descriptos, como opuesto a la catalogación y agregación de valor realizado por organismos creados a tal fin. Sin embargo, es obvio que para que los documentos sean auto-descriptos, un conjunto central básico de elementos de metadatos necesitan ser identificados y cada creador de documentos electrónicos debería estar capacitado para implementarlo en el registro creado. Stuart Weibel, miembro de OCLC y promotor de Dublin Core, desde sus inicios, enfatiza esta postura como una de las bases fundacionales de este, junto con la de facilitar el descubrimiento de recursos en la red. En consecuencia, los creadores de la meta-información no serían necesaria ni inevitablemente catalogadores de bibliotecas, ni la prioridad estaría puesta en la descripción del recurso.

Coincidimos con la conclusión de Gradmann: *"el contexto general de producción y uso de esta información es substancialmente diferente y esta impulsada para ir mas allá del paradigma tradicional de la catalogación. Considerar que el proceso de creación de metadatos es un tipo de catalogación simplificada, sería probablemente un error importante"* Por tanto, metadatos podría definirse como *"cualquier información que soporta la efectiva utilización de datos, incluyendo información que pueda facilitar la gestión, el acceso y el análisis de datos".* [14]

1.3.2 Funciones de los metadatos

Las funciones de los metadatos pueden ser analizadas desde el nivel del sistema y desde el nivel del usuario final. En el primero, facilitan la interoperabilidad y la capacidad de compartir datos entre las herramientas de descubrimiento de recursos, lo que acelera la concreción de proyectos, mejora la utilidad de investigaciones y de la toma de decisiones, y reduce costos al minimizar la duplicación de esfuerzos. Desde la perspectiva del usuario pueden facilitar la capacidad de determinar: que datos están disponibles (¿existen los objetos de información?, ¿dónde están?); si satisfacen necesidades especificas (¿es auténtico?, ¿es bueno?, ¿cómo puedo determinar si es útil o no?); cómo adquirirlo; y, cómo transferirlo a un sistema local.

Los Metadatos tienen dos funciones principales:

- Proveer un medio para descubrir que datos existen y como podrían ser obtenidos o accedidos
- Proveer un mecanismo de búsqueda para coleccionar metadatos. Documentar contenido, calidad y rasgos o características de un conjunto de datos y así dar una indicación de la propiedad, idoneidad o correspondencia de uso.

1.3.3 Diferentes tipos de metadatos

En un documento electrónico editado por la University of Bristrol, focalizado en datos secundarios que describen imágenes, se postula la necesidad de acompañar estos datos intrínsecos con otros tipos de metadatos que realce sus datos extrínsecos. Entre ellos se encuadrarían los siguientes tipos:

- Administrativos: contienen información relevante a la gestión de datos en un repositorio específico; por ej., fecha de creación del recurso, el nombre del técnico que produjo el "scanning" de la imagen y por ende la creó, y las especificaciones de la imagen, como tamaño, resolución, etc.
- De Procedencia/Origen: contienen información respecto al origen de los objetos descriptos (data object); por ej., para el "scanning" de imágenes debería especificarse que fueron creadas a partir de artefactos físicos de alguna colección especial.
- De Vinculación (linkage): contienen información de cómo están relacionados un conjunto de recursos; por ej., el hecho de que la página Web con la imagen visualizada forma parte de un sitio que contiene mas páginas con relación mutua, a través de la especificación de su vínculo (link).
- De Términos y Condiciones: especifica acuerdos de licencias y otros ordenamientos de acceso; por ej., determinados "scanning" de imágenes de documentos solo pueden ser accedidos por clientes registrados por la institución Estructurales: describen algún aspecto de la estructura interna de un objeto; por ej., tabla de contenidos, lista de figuras y tablas.

1.3.4 DUBLIN CORE

OCLC (Online Computer Library Center) y NCSA (National Center for Supercomputing Applications) promocionaron el primer Metadata Workshop, en Dublin, Ohio, en Marzo de 1995. Por un lado, bibliotecarios, archivistas, científicos de humanidades y geógrafos, por el otro: profesionales de la ciencia informática, especializados en Internet, protocolo Z39.50 y SGML (Standard Generalized Markup Language) se reunieron para identificar el alcance de la cuestión, así como obtener consenso a partir de una lista de elementos que puedan describir datos de forma simple en un amplio espectro de áreas temáticas y establecer las bases para futuros progresos en el área de la descripción de información digital. Las características y objetivos centrales del Dublin Core pueden sintetizarse en:

- **Simplicidad:** se tiende a la auto-descripción de parte de los productores de sitios Web. El modelo de descripción posee una semántica de comprensión corriente,

equivalente al denominador común menor de descripción de recursos. No intenta reemplazar a modelos descriptivos como AACR2/MARC, sino proveer un núcleo (core set) de elementos descriptivos utilizables tanto por catalogadores como por no catalogadores para una descripción simple del recurso

- **Interoperabilidad semántica:** diferentes comunidades (bibliotecas, museos, GIS (Geographic Information System), que conviven en Internet, utilizan diferentes normas para la descripción de recursos, que responden a diferentes necesidades de las mismas y que evolucionan de manera independiente. Sin embargo, la mayoría de los recursos comparten elementos comunes, aún con nombres diferentes, dependiendo de cada comunidad. Se intenta promover un conjunto de descriptores comprensibles a todas las disciplinas, favoreciendo de este modo la búsqueda interdisciplinaria.
- **Flexibilidad:** permite incorporar estructuras simples "la auto-descripción ya aludida" hasta aquellas más elaboradas semánticamente 'Document-like Objects' (DLO's): en el DC-Metadata Workshop, de Marzo de 1995, esta expresión 'objetos como documentos' nunca fue precisamente definida. Se partió de la convicción "extraída a partir de estadísticas de uso" que la mayoría de los objetos solicitados en Internet eran fundamentalmente textuales. Según el consenso logrado, se sostuvo que la probabilidad de progresar respecto a este problema estaría incrementada si inicialmente se concentrara la atención en la descripción de algo familiar, para luego extenderse a recursos no-textuales como imágenes, sonidos, videoclips, o multimediales. A partir de esta visión restringida, se manifiesta en el formato un predominio de elementos intrínsecos sobre los extrínsecos, impactando negativamente tanto en recursos electrónicos textuales (la ausencia de elementos como costo, modo de acceso, etc.) como no-textuales (por ej. su incapacidad para describir completamente imágenes geoespaciales). Carácter intrínseco y extensible: prioriza la descripción de elementos intrínsecos del recurso, pero reconoce la importancia de los extrínsecos permitiendo mecanismos de extensión, que posibilitan al usuario agregar material descriptivo extra para sitios específicos o campos especializados.

- **Independencia sintáctica:** hasta el momento se evitan las reglas sintácticas formales ya que está orientado a ser aplicado por un amplio espectro de disciplinas y programas informáticos.
- **Carácter opcional:** ningún elemento tiene carácter obligatorio ya que es el productor de la información quien tiene a su cargo la descripción, radicando allí su simpleza. Se agrega el hecho, que no en todas las circunstancias son necesarios todos los elementos prescriptos para describir un recurso.
- **Repetibilidad:** todos los elementos son repetibles.
- **Uso de calificadores (qualifiers):** cada elemento puede ser enriquecido por medio del uso de calificadores. Estos proporcionan información adicional y valor agregado a los datos del elemento original.
- **Encastre (Embedded):** el modo más fácil de desplegar metadatos en la Web es encastrándolos en documentos HTML, utilizando META tag, que permiten la asignación de más y mejores calificadores. Los metadatos están integrados al recurso y pueden ser buscados y recuperados por los buscadores Web.

La descripción de los elementos de Dublin Core se sintetizan en:

Título

Etiqueta: DC.Title

El nombre dado a un recurso, usualmente por el autor.

Autor o Creador

Etiqueta: DC.Creator

La persona u organización responsable de la creación del contenido intelectual del recurso. Por ejemplo, los autores en el caso de documentos escritos, artistas, fotógrafos e ilustradores en el caso de recursos visuales.

Claves

Etiqueta: DC.Subject

Los tópicos del recurso. Típicamente, Subject expresará las claves o frases que describen el título o el contenido del recurso. Se fomentará el uso de vocabularios controlados y de sistemas de clasificación formales.

Descripción

Etiqueta: DC.Description

Una descripción textual del recurso, tal como un resumen en el caso de un documento o una descripción del contenido en el caso de un documento visual.

Editor

Etiqueta: DC.Publisher

La entidad responsable de hacer que el recurso se encuentre disponible en la red en su formato actual, por ejemplo la empresa editora, un departamento universitario u otro tipo de organización.

Otros Colaboradores

Etiqueta: DC.Contributor

Una persona u organización que haya tenido una contribución intelectual significativa en la creación del recurso pero cuyas contribuciones son secundarias en comparación a las de las personas u organizaciones especificadas en el elemento Creator (por ejemplo, editor, ilustrador y traductor).

Fecha

Etiqueta: DC.Date

Una fecha en la que el recurso se puso a disposición del usuario en su forma actual. Esta fecha no ha de confundirse con la que pertenece al elemento Coverage, que sería asociada con el recurso sólo en la medida en que el contenido intelectual está de algún modo relacionado con esa fecha.

Recomendamos la utilización de uno de los formatos definidos en el documento "Date and Time Formats", basado en la norma ISO 8601 que incluye, entre otras, fechas en el formato AAAA y AAAA-MM-DD. De esta forma la fecha 1994-11-05 correspondería al 5 de Noviembre de 1994.

Tipo del Recurso

Etiqueta: DC.Type

La categoría del recurso, por ejemplo página personal, romance, poema, minuta, diccionario. Para asegurar la interoperabilidad, Type debería ser seleccionado de entre una lista de valores que actualmente se encuentra bajo desarrollo en un grupo de trabajo.

Formato

Etiqueta: DC.Format

El formato de datos de un recurso, usado para identificar el software y posiblemente, el hardware que se necesitaría para mostrar el recurso. Para asegurar la interoperabilidad, los valores de Format deberían ser seleccionados de entre una lista de valores que actualmente se encuentra bajo desarrollo en un grupo de trabajo.

Identificador del Recurso

Etiqueta: DC.Identifier

Secuencia de caracteres usados para identificar unívocamente un recurso. Ejemplos para recursos en línea pueden ser URLs y URNs (cuando estén implementados). Para otros recursos pueden ser usados otros formatos de identificadores, como por ejemplo ISBN ("International Standard Book Number" - Número Internacional Normalizado para Libros)

Fuente

Etiqueta: DC.Source

Secuencia de caracteres utilizado para identificar unívocamente un trabajo a partir del cual proviene el recurso actual. Por ejemplo, es posible usar Source con la fecha de 1603 como descripción de una película basada en una obra de Shakespeare, pero es preferible, en ese caso, usar Relation "IsBasedOn" con una referencia a un recurso distinto cuya descripción contenga el elemento Date con valor 1603.

Lengua

Etiqueta: DC.Language

Lenguaje del contenido intelectual del recurso. Prácticamente el contenido de este campo debería coincidir con los de la RFC 1766; por ejemplo: en, es, de, fi, ja y zh.

Relación

Etiqueta: DC.Relation

Un identificador de un segundo recurso y su relación con el recurso actual. Este elemento permite enlazar los recursos relacionados y las descripciones de los recursos. Por ejemplo:

IsVersionOf	Incluye la edición de un trabajo
IsBasedOn	La traducción de un trabajo
IsPartOf	Un capítulo de un libro
IsFormatOf	Un mecanismo de transformación de un conjunto de datos en una imagen

Para asegurar la interoperabilidad, las relaciones deberían ser seleccionadas de una lista de elementos que actualmente se encuentra bajo desarrollo en un grupo de trabajo.

Cobertura

Etiqueta: DC.Coverage

La característica de cobertura espacial y/o temporal del contenido intelectual del recurso.

La cobertura espacial se refiere a una región física (por ejemplo, sector celestial); uso de coordenadas (por ejemplo, longitud y latitud) o nombres de lugares extraídos de una lista controlada.

La cobertura temporal se refiere al contenido del recurso en vez de a cuándo fue creado o puesto accesible ya que este último pertenece al elemento Date.

Derechos

Etiqueta: DC.Rights

Una referencia (URL, por ejemplo) para una nota sobre derechos de autor, para un servicio de gestión de derechos o para un servicio que dará información sobre términos y condiciones de acceso a un recurso. Una especificación formal del elemento Rights se encuentra actualmente en discusión y por lo tanto su uso se considera experimental.

En el ANEXO II (Fig. 6-5) se representa una tabla con los quince elementos componentes de Dublin Core.

1.4 Objeto de estudio.

El aspecto más relevante de la Sociedad de la información ha sido la concreción de una red mundial, la cual ha tenido un fuerte impacto en todos los ámbitos sociales y profesionales. Sin embargo, a pesar de todas sus bondades, Internet se ha convertido en un inmenso almacén, donde se albergan conocimientos de muy diversos contenidos, relevancia, accesibilidad y utilidad; situación que algunos expertos han comparado a una biblioteca sin catálogos.

Se han acuñado términos tales como sobre-información, para referirse a su excesiva disponibilidad, lo que produce un efecto desorientador y perturbador en el usuario final, que se ve desbordado ante la enorme cantidad de información potencial, dispersa y a veces poco relevante.

El crecimiento sin precedentes del mundo digital ha motivado la necesidad de mejorar la manera de acceder a lo relevante en este vasto entorno.

Durante los últimos años han surgido propuestas imaginativas que intentan reconducir esta situación, mejorar la disponibilidad y accesibilidad de aquella información que es realmente pertinente.

El protocolo de transmisión de datos OAI-PMH (Open Archives Initiative – Protocol for Metadata Harvesting), surge con el objetivo de mejorar lo antes planteado. El cual fue el objeto de estudio de este trabajo, profundizando en el Data Providers, siendo esta la parte implementada del protocolo y la que será abordada en este documento.

1.5 Análisis de otra solución existente.

Para la realización del proyecto se estudiaron una serie de software que realizarán las funciones especificadas por el protocolo. Un ejemplo de ello es DSpace, que es un repositorio digital.

DSpace es un proyecto con similares características a los objetivos trazados y se puede decir que de los estudiados fue el que más se acercó a las exigencias planteadas. Fue realizado en el lenguaje de probación Java, utiliza JDBC para la conexión a la base de datos y es multiplataforma, pero como todos los demás esta enfocado a las exigencias que los desarrolladores se propusieron y la utilización de él implicaría interpretar y remodificar parte de su

código y adaptarlo a objetivos trazados. Por lo que se determinó realizar un nuevo proyecto tomando las experiencias y las ventajas de los ya implementados.

1.6 Conclusiones.

En el presente capítulo se han descrito los conceptos necesarios para lograr un correcto entendimiento, de los términos que serán tratados a profundidad en capítulos posteriores. Una de las concepciones tratadas fueron los tipos de metadatos, donde se detalló el formato Dublin Core, el cual es el formato requerido por el protocolo.

Se ha formalizado el objeto de estudio, describiéndose la situación problemática que dio lugar al trabajo. También se realizó un análisis de otra posible solución, planteándose las razones que dieron lugar a este trabajo.

2

CAPÍTULO

TENDENCIAS Y TECNOLOGÍAS ACTUALES A CONSIDERAR.

2.1 Introducción.

En este capítulo se analizan las tendencias y tecnologías que pudieran ser útiles en el desarrollo de la solución propuesta. Se describen los Sistemas Gestores de Bases de Datos, y se da una descripción general el Framework de persistencia Hibernate. Se realiza un análisis de los lenguajes de programación Web así como se hace una breve descripción del lenguaje de marcado extensible (XML). Determinando lo más apropiado a utilizar en el proyecto.

2.2 Software Libre.

Dadas las restricciones y dependencias que proporciona el software propietario, ya sea desde el punto de vista económico como político, se ha observado recientemente una tendencia de emigración hacia el software libre.

Cada vez se promociona más la migración desde los sistemas con licencia comercial que están en poder de unos pocos monopolios de la rama de la informática (que se reservan el derecho de vender sus productos a quienes consideren apropiado), hacia aquellos que se denominan “libres”, es decir una vez obtenido un software libre, puede ser usado, copiado, estudiado, modificado y redistribuido libremente, incluso suelen estar disponible gratuitamente en Internet.

El presente trabajo determino hacer uso de las herramientas y tecnologías libres.

2.3 Aplicaciones Web.

La Web en sus orígenes fue pensada como un medio para desplegar información, esta reposa de manera estática en los servidores la cual es accedida a través de una consulta hecha por un

navegador valiéndose del protocolo HTTP (Hyper Text Transfer Protocol). Actualmente se maneja el mismo concepto en la comunicación cliente-servidor (navegador - webserver) solo que no necesariamente el resultado de la comunicación debe provenir de la carga de una página estática, esta puede ser el resultado de la ejecución en el servidor de alguna lógica de programación. Esto último no necesariamente se le llama una aplicación Web, pero se acerca al concepto. Se puede considerar como una aplicación Web a "un sitio Web donde la navegación a través de él y la entrada de datos por parte de un usuario, afectan el estado de la lógica del negocio. En esencia, una aplicación Web usa un sitio Web como entrada (front-end) a una aplicación típica. ...Si no existe lógica del negocio en el servidor, el sistema no puede ser llamado aplicación Web". Bajo este concepto las aplicaciones Web no solo se encargan de desplegar información, sino que también, deben contener una lógica asociada que permita apoyar algún proceso propio del negocio para el cual fue diseñada.

Se puede apreciar que los componentes de la arquitectura Web son: el servidor Web, la red física que permite la comunicación y un navegador o cliente.

Existen casos en que esta arquitectura es un poco más compleja, o sea, incluye un nuevo elemento: una aplicación que se ejecuta en el servidor. Este tipo de arquitectura permite manejar la lógica del negocio a través de una "aplicación Web", que es como se denomina a este tipo de sistemas Web. La aplicación que se ejecuta en el servidor se encarga de controlar el estado del negocio y de gestionar los datos almacenados con ayuda de algún Sistema Gestor de Bases de Datos (SGBD).

En la actualidad se ha generalizado el uso de aplicaciones Web dada las grandes posibilidades que brindan, y dado que los clientes sólo necesitan un navegador, capaz de interpretar código con formato HTML, para hacer uso de ellas: no tienen que instalar ningún componente de software adicional.

2.4 La seguridad de las trasmisiones en la Web.

Dado el gran auge que hoy en día tiene Internet, su uso se ha masificado enormemente. Desde páginas meramente informativas hasta sitios interactivos usando tecnologías nuevas.

Empresas de diversa índole ya usan la Internet para comunicarse y el problema principal que surgió es la confiabilidad en que lo que se esta comunicando no sea visto por personas que puedan hacer mal uso de dicha información.

A raíz de todo esto surgieron tecnologías que persiguen mejorar la seguridad de todas estas comunicaciones.

2.4.1 Seguridad en la transmisión Web.

La seguridad de este tipo se basa en el hecho de poder encriptar los mensajes que se envían por la red entre un servidor y un cliente y que sólo ellos puedan descifrar los contenidos a partir de una clave común conocida solo por los dos.

Para llevar a cabo esta seguridad se crearon diversos protocolos basados en esta idea:

- SSH: Usado exclusivamente en reemplazo de telnet
- SSL: Usado principalmente en comunicaciones de hipertexto pero con posibilidad de uso en otros protocolos
- TSL: Es del mismo estilo del anterior.
- HTTPS: Usado exclusivamente para comunicaciones de hipertexto

2.4.2 Protocolo SSL.

En la actualidad la mayoría de los navegadores utilizan el protocolo **SSL** (Secure Sockets Layer) para transmitir información de manera segura.

Es un sistema diseñado y propuesto por Netscape Communications Corporation. En la pila OSI, se encuentra entre los niveles de TCP/IP y el de los protocolos HTTP, FTP, SMTP. Proporciona sus servicios de seguridad cifrando los datos intercambiados entre el servidor y el cliente con un algoritmo de cifrado simétrico, típicamente el RC4 o IDEA, y cifrando la clave de sesión de RC4 o IDEA mediante un algoritmo de cifrado de clave pública, típicamente el RSA. La clave de sesión es la que se utiliza para cifrar los datos que vienen de él y van al servidor seguro. Se genera una clave de sesión distinta para cada transacción, lo cual permite que aunque sea reventada por un atacante en una transacción dada, no sirva para descifrar futuras transacciones. [1]

Proporciona cifrado de datos, autenticación de servidores, integridad de mensajes y, opcionalmente, autenticación de cliente para conexiones TCP/IP.

2.5 Servidor Web.

Entre los distintos tipos de servidores Web que existen en la actualidad, se seleccionó Tomcat la versión 5.0 para utilizarlo en la solución propuesta. Este servidor de aplicaciones también llamado Jakarta Tomcat o Apache Tomcat funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation.

Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems.

2.6 Lenguajes de Programación para la Web.

Entre los distintos lenguajes de programación para la Web que existen en la actualidad, se destacan dos grupos, que se diferencian entre sí por el lugar que ocupan en la arquitectura Cliente / Servidor característica de los sistemas Web. El primer grupo está formado por los lenguajes que se ejecutan en el servidor (en inglés, server side languages). Como ejemplos más sobresalientes tenemos algunos como PERL, ASP, PHP, Java, JSP, los módulos CGI, etc. Estos se caracterizan por desarrollar la lógica de negocio dentro del Servidor, además de ser los encargados del acceso a bases de datos, tratamiento de la información, etc. Dentro del segundo grupo se encuentran aquellos lenguajes que se ejecutan en el cliente (client side languages). En este caso están el JavaScript y el Visual Basic Script, que son los encargados de aportar dinamismo a la aplicación en los navegadores.

Es importante seleccionar el lenguaje a utilizar, tanto del lado del servidor como del lado del cliente. A continuación se hace un breve análisis de los lenguajes de uso más común en la actualidad, con vistas a hacer una apropiada selección.

Perl (Practical Extracting and Reporting Language):

Es un lenguaje de programación muy utilizado para construir aplicaciones CGI para el Web, y es considerado el lenguaje perfecto para este fin dadas sus facilidades en cuanto a la manipulación de texto. Es un lenguaje de libre uso. Antes estaba muy asociado a la plataforma Unix, pero en la

actualidad está disponible en otros sistemas operativos como Windows. Perl es un lenguaje de programación interpretado, al igual que muchos otros lenguajes de Internet como JavaScript o ASP.

ASP (Active Server Pages):

Es una tecnología propietaria de Microsoft. Se utiliza exclusivamente en los servidores Web de Microsoft (Internet Information Server y Personal Web Server), lo cual constituye su principal desventaja. Los scripts ASP se ejecutan, por lo tanto, en el servidor y puede utilizarse conjuntamente con HTML y JavaScript para realizar tareas interactivas y en tiempo real con el cliente. Con ASP se pueden realizar fácilmente páginas de consulta de bases de datos, funciones sencillas como obtener la fecha y la hora actual del sistema servidor, cálculos matemáticos simples, etc. Actualmente se ha presentado ya la segunda versión de ASP: el ASP.NET, que comprende algunas mejoras en cuanto a posibilidades del lenguaje y rapidez con la que funciona, y que presenta algunas diferencias en cuanto a sintaxis, de modo que se ha de tratar de distinta manera uno de otro. Para implementar ASP.NET es necesario instalar en el Servidor la Plataforma .NET.

PHP (Personal Home Page):

Es un lenguaje de programación del lado del servidor gratuito, de código abierto, e independiente de plataforma, muy rápido, con una gran librería de funciones y mucha documentación. Es también un lenguaje interpretado y embebido en el HTML. Su sintaxis es muy parecida a la del lenguaje C.

PHP cuenta con un motor de plantillas denominado SMARTY que permite separar la lógica de la programación de la presentación, es decir, el código PHP del código HTML. Por tanto se puede modificar uno sin afectar el otro. SMARTY es sumamente rápido, entre otros aspectos porque cada plantilla se compila sólo una vez y sólo recompila aquellas que han sido modificadas. Permite que una misma plantilla sea utilizada por varias páginas PHP que muestren el contenido en el mismo formato, independientemente de que los procedimientos usados para obtener la información a mostrar sean diferentes.

Otra de las ventajas de utilizar PHP, es poder usar la librería ADOdb, una capa de abstracción de base de datos, de alta velocidad, y que tiene características avanzadas, como la gestión de sesiones, generación automática del código SQL, simulación de SELECT LIMIT para todas las bases de datos y monitorización del rendimiento. Su aprendizaje resulta muy fácil, sobre todo si se está familiarizado con la programación de Windows, ya que utiliza muchas convenciones de ADO. Esta capa lleva siendo utilizada desde el año 2000, actualmente, con una comunidad amplia de usuarios. Su licencia es más que flexible (BSD). Esto significa que se puede incorporar (incluso compilar) en las aplicaciones libre de tasas, sin necesidad de solicitar el permiso del autor.

Java:

Es un lenguaje orientado a objetos, eso implica que su concepción es muy próxima a la forma de pensar humana. También posee otras características muy importantes:

Es un lenguaje que es compilado, generando ficheros de clases compilados, pero estas clases compiladas, son en realidad interpretadas por la máquina virtual. Siendo la máquina virtual de java la que mantiene el control sobre las clases que se estén ejecutando.

Es un lenguaje multiplataforma: El mismo código java que funciona en un sistema operativo, funcionará en cualquier otro sistema operativo que tenga instalada la máquina virtual java.

Es un lenguaje seguro: La máquina virtual, al ejecutar el código java, realiza comprobaciones de seguridad, además el propio lenguaje carece de características inseguras, como por ejemplo los punteros.

Gracias al API de java podemos ampliar el lenguaje para que sea capaz de, por ejemplo, comunicarse con equipos mediante red, acceder a bases de datos, crear páginas HTML dinámicas, crear aplicaciones visuales al estilo Windows.

Para poder trabajar con este lenguaje es necesario emplear un software que permita desarrollar en java. Existen varias alternativas comerciales en el mercado: JBuilder, Visual Age, Visual Café y un conjunto de herramientas libres, que permiten trabajar con java.

2.6.1 Selección del lenguaje a utilizar.

Dadas las características antes planteadas de cada uno de los lenguajes, la decisión se encontraba entre PHP y Java.

- PHP, con todas las ventajas que presenta por su propia esencia, es más rápido y es realmente Open Source.
- Java, por su parte es completamente orientado a objeto, puede ejecutarse en un cluster y es posible realizar monitoreo de uso de recursos, puede conectarse a un pool de solicitudes para entornos distribuidos, tiene soporte para documentación incluido (JavaDoc), Java es un lenguaje más estricto en la cuestión de la seguridad que PHP.

En general se plantea que para aplicaciones medianas o pequeñas se recomienda usar PHP, pero para aplicaciones grandes es mejor usar Java.

Después de un análisis profundo se propone como lenguaje del lado del servidor usar Java por ser más robusto.

Como lenguaje del lado del cliente se propone el uso de JavaScript, que es un lenguaje orientado a eventos e interpretado, es decir, no requiere compilación. El navegador del usuario se encarga de interpretar las sentencias JavaScript contenidas en una página HTML y ejecutarlas adecuadamente. En sus inicios fue desarrollado por Netscape y la primera versión JavaScript 1.0 fue por primera vez introducida en su navegador Netscape Navigator 2.0. Luego surgió la versión JavaScript 1.1 que se introdujo en el Netscape Navigator 3.0. Sin embargo hasta este momento no era soportado por el navegador de Microsoft, Internet Explorer. Luego de algunos esfuerzos se logró total compatibilidad entre JavaScript 1.3, que está incluido en Netscape Navigator 4.06 y posteriores versiones, y ECMA-262, un estándar para JavaScript introducido por Microsoft en el Internet Explorer.

2.7 Lenguaje de marcado extensible (Extensible Markup Language, XML).

“XML, es el estándar de Extensible Markup Language, no es más que un conjunto de reglas para definir etiquetas semánticas que nos organizan un documento en diferentes partes. XML es un

metalenguaje que define la sintaxis utilizada para definir otros lenguajes de etiquetas estructurados.”[6]

En primer lugar para entenderlo bien hay que olvidarse un poco, sólo un poco de HTML. En teoría HTML es un subconjunto de XML especializado en presentación de documentos para la Web, mientras que XML es un subconjunto de SGML especializado en la gestión de información para la Web. En la práctica XML contiene a HTML aunque no en su totalidad. La definición de HTML esta contenido totalmente dentro de XML y por lo tanto que cumple la especificación SGML es XHTML (Extensible, Hypertext Markup Language).

“XML está revolucionando la comunicación entre aplicaciones o, de forma más general, la comunicación entre equipos, pues ofrece un formato de datos universal que permite adaptar o transformar fácilmente la información.”

XML intenta ser un formato absolutamente genérico, con el que describir cualquier tipo de fichero.

2.8 Sistemas Gestores de Bases de Datos (SGBD).

Un sistema gestor de bases de datos es un software que permite crear y explotar bases de datos. Se pueden distinguir dos tipologías bien definidas:

- Sistemas de gestión de bases de datos (SGBD) cuyo propósito es general. Acostumbran a basarse en el modelo relacional. Su finalidad principal es la gestión de datos comerciales, administrativos y, en general, cualquier tipo de datos. Son propiamente sistemas de recuperación de datos.
- Sistemas de gestión documental (SGD) que acostumbran a basarse en el modelo textual. Están diseñados para gestionar datos textuales, no están necesariamente bien estructurados y acostumbran a incorporar controles terminológicos. Estos sistemas son temas de recuperación de información.

En la actualidad existe una gran variedad de SGBD, tanto de tipo comercial como libre. Entre los más usados dentro del grupo de los comerciales se encuentra **Oracle**, el cual es considerado el SGBD más completo que existe. Sus características más destacadas son el soporte de transacciones, su gran estabilidad y seguridad, su escalabilidad, así como que es un sistema

multiplataforma, entre otras ventajas. En sus inicios fue muy revolucionario dado que usaba la filosofía de bases de datos relacionales, algo que por los años 70, fecha en que surge Oracle, era todavía desconocido. Hasta hace poco su dominio en el mercado de los servidores de bases de datos empresariales era casi total, pero recientemente está sufriendo la competencia del MS SQL Server de Microsoft y de la oferta de otros SGBD libres.

SQL Server es un potente SGBD que está totalmente habilitado para Web. "Ostenta marcas de referencia en cuanto a escalabilidad y confiabilidad, que son críticas para el éxito de bases de datos de gran tamaño. El SQL Server permite lograr una gran velocidad en el procesamiento de transacciones, y agilidad en todas sus operaciones."[13] A pesar de todas las ventajas que presenta este SGBD, tiene el inconveniente de que, al igual que Oracle, no es un sistema libre.

MySQL "implementa funcionalidades Web que permiten un acceso a los datos, seguro y fácil, desde Internet."[6] "Es uno de los SGBD más populares, desarrollado bajo la filosofía de código abierto."[16]

La licencia GPL de MySQL obliga a distribuir cualquier producto derivado (aplicación) bajo esa misma licencia. Por tanto MySQL tiene sus restricciones: sólo es gratis si se está dispuesto a distribuir la aplicación que se quiere desarrollar bajo esa misma licencia GPL. Si se desea distribuir la aplicación comercialmente, entonces se debe pagar la licencia comercial de MySQL que permite hacer exactamente eso.

MySQL tiene como una de sus principales ventajas la velocidad en la lectura de datos, pero a costa de eliminar un conjunto de facilidades que presentan otros SGBD: integridad referencial, bloqueo de registros, procedimientos almacenados, entre otros. En recientes versiones de MySQL (la versión 4 y la 5) se incluyen algunas de estas características, pero indudablemente esto va en detrimento de la velocidad.

PostgreSQL está considerado como el SGBD de código abierto más avanzado del mundo. PostgreSQL proporciona un gran número de características que normalmente sólo se encontraban en las bases de datos comerciales de alto calibre tales como DB2 u Oracle. [7]

Es un SGBD objeto-relacional, aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Su avanzada funcionalidad se pone de manifiesto con las

consultas SQL declarativas, el control de concurrencia multiversión, soporte multiusuario, transacciones, optimización de consultas, herencia y valores no atómicos (atributos basados en vectores y conjuntos).

Es altamente extensible: soporta operadores y tipos de datos definidos por el usuario. Soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92. Cuenta con una API (del inglés Application Program Interface) flexible lo cual ha permitido dar soporte para el desarrollo con PostgreSQL en diversos lenguajes de programación entre los que se incluyen: Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, y Pike. Tiene soporte para lenguajes procedurales internos, incluido un lenguaje nativo denominado PL/pgSQL, el cual es comparable con el lenguaje procedural de Oracle PL/SQL.

A diferencia de MySQL que, como se explicó anteriormente, tiene sus restricciones en cuanto al tema de las licencias, PostgreSQL es totalmente libre. Las polémicas entre los partidarios de MySQL y los defensores de PostgreSQL pueden clasificarse como del tipo “Guerra Santa”, junto a otras como Linux vs. Windows, Mac. Vs. PC. Muchos desarrolladores en sus discusiones a través de la Web en torno al tema de qué es mejor: MySQL o PostgreSQL, recomiendan la utilización de PostgreSQL para la elaboración de un sistema robusto y para lograr mayor escalabilidad. La mayoría coincide en que cada SGBD tiene sus ventajas y desventajas, y que la elección de uno de los dos depende de lo que se quiera construir. Se destaca sobre todo que MySQL ha avanzado vertiginosamente comparado con PostgreSQL que ya lleva alrededor de 15 años de desarrollo.

2.9 Hibernate.

Hibernate es una capa de persistencia objeto/relacional y un generador de sentencias sql. Te permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. De una manera muy rápida y optimizada se puede generar BBDD en cualquiera de los entornos soportados: Oracle, DB2, MySQL, PostgreSQL, Sybsae, SAP DB, HypersonicSQL, Microsoft SQL Server, Progress, Mckoi SQL,

Pointbase, Interbase e Ingres. Y otras de los aspectos más importante, es que esta herramienta es open source, lo que supone, entre otras cosas, que no tenemos que pagar nada por adquirirlo.
Uno de los posibles procesos de desarrollo consiste en, una vez tengamos el diseño de datos realizado, mapear este a ficheros XML siguiendo la DTD de mapeo de Hibernate.
Desde estos podremos generar el código de nuestros objetos persistentes en clases Java y también crear BBDD independientemente del entorno escogido.
Hibernate se integra en cualquier tipo de aplicación justo por encima del contenedor de datos. Una posible configuración básica de Hibernate se muestra en el ANEXO II (Fig. 6-6) donde se puede observar como Hibernate utiliza la BBDD y la configuración de los datos para proporcionar servicios y objetos persistentes a la aplicación que se encuentre justo por arriba de él.
Dentro de las ventajas que proporciona, la que marcó una pauta decisiva en el uso de Hibernate en la implementación del protocolo, fue el hecho de soporta la mayoría de los sistemas de bases de datos SQL. Permitiendo que sea el cliente quien decida, cual es el gestor de base de datos.

2.10 Metodologías de Desarrollo de Software.

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación, para el desarrollo de productos software.
Es como un libro de recetas de cocina, en el que se van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Además detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla.
Analizaremos a continuación tres de las más conocidas.

El Proceso Unificado de Rational (RUP):

La metodología RUP, llamada así por sus siglas en inglés Rational Unified Process. Se basa en casos de uso para describir lo que se espera del software y está muy orientado a la arquitectura del sistema, documentándose lo mejor posible, basándose en UML (Unified Modeling Language) como herramienta principal.

RUP “se divide en 4 fases el desarrollo del software:

- **Inicio**, El Objetivo en esta etapa es determinar la visión del proyecto.
- **Elaboración**, En esta etapa el objetivo es determinar la arquitectura óptima.
- **Construcción**, En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.
- **Transmisión**, El objetivo es llegar a obtener el reléase del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los Objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

Vale mencionar que el ciclo de vida que se desarrolla por cada iteración, es llevada bajo dos disciplinas:

- Disciplina de Desarrollo
- Disciplina de Soporte

Es recomendable que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, y que cada una se convierte luego en un entregable al cliente. Esto trae como beneficio la retroalimentación que se tendría en cada entregable o en cada iteración.

Los elementos del RUP son:

- **Actividades**, Son los procesos que se llegan a determinar en cada iteración.
- **Trabajadores**, Vienen hacer las personas o entes involucrados en cada proceso.
- **Artefactos**, Un artefacto puede ser un documento, un modelo, o un elemento de modelo.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

Esta metodología también tiene la ventaja de venir acompañada de una potente herramienta que soporta todos los procesos básicos de RUP: Rational Rose Enterprise Edition 2003.

Programación Extrema (XP):

La Programación Extrema, es una de las metodologías de desarrollo de software más exitosas en la actualidad, utilizadas para proyectos de corto plazo.

XP, como toda metodología ágil, intenta reducir la complejidad del software por medio de un trabajo orientado directamente al objetivo, basado en las relaciones interpersonales y la velocidad de reacción.

Supone la disposición en todo momento, por parte del equipo de trabajo, de un representante competente del cliente, que debe estar en condiciones de dar una respuesta rápida y correcta a cualquier pregunta del equipo de desarrollo de forma que no se retrase la toma de decisiones.

La base para el desarrollo del software que usa esta metodología son las llamadas User Stories, historias escritas por el cliente en las que describe escenarios sobre el funcionamiento del sistema y que no sólo están limitados a la interfaz de usuario, sino que también pueden describir modelos, dominio, etc. Estas User Stories junto a la arquitectura que se persigue, sirve de base para crear un plan de “entregas de software” entre el equipo de desarrollo y el cliente, para cada una las cuales se definen objetivos y las iteraciones (generalmente cortas) necesarias para cumplirlos. Las User Stories y los casos de pruebas son la base sobre la que se asienta el trabajo del desarrollador.

Esta metodología apuesta por iteraciones cortas que generan software que el cliente puede ver.

La codificación del software se realiza siempre en parejas (dos programadores, un ordenador), las cuales no son fijas sino que rotan a lo largo del proyecto, y el código que escriben no les pertenece sólo a ellos sino al equipo completo. El objetivo ideal sería que cada integrante del equipo trabaje al menos una vez con cada uno de los demás integrantes y con cada componente software, de forma que el conocimiento de la aplicación completa lo posea el equipo entero y no unos pocos miembros. Se programa sólo la funcionalidad requerida para la entrega en curso, se trabaja en función de las necesidades del momento, por lo que no se le da importancia al análisis como fase independiente.

Según Kent Beck, "la programación extrema es una forma ligera, eficiente, flexible, predecible, científica y divertida de generar software" (Kent Beck, Extreme Programming Explained).

Microsoft Solution Framework (MSF):

Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

Presenta las siguientes características:

- **Adaptable:** es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.
- **Escalable:** puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas a más.
- **Flexible:** es utilizada en el ambiente de desarrollo de cualquier cliente.
- **Tecnología Agnóstica:** porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación.

2.10.1 Selección de la metodología a utilizar.

Las metodologías analizadas tienen pocas similitudes entre si y cada una presenta sus propias características:

- La Metodología RUP es más adaptable para proyectos de largo plazo.
- La Metodología XP en cambio, se recomienda para proyectos de corto plazo.
- La Metodología MSF se adapta a proyectos de cualquier dimensión y de cualquier tecnología.

De donde se puede inferir, que lo más importante antes de elegir la metodología que usará para la implementación del software, es determinar el alcance que tendrá y luego de ahí ver cuál es la que más se acomoda a la aplicación.

Por otra parte, es importante mencionar que algunos autores consideran que dado el carácter general de RUP, todas las otras metodologías son casos particulares de esta.

En cuanto a la relación del equipo con el cliente, XP se basa en la disposición de un representante competente que reúna una serie de condiciones, pero en la práctica, es poco probable que el cliente pueda prescindir de una persona con esas características por un tiempo, ya que le resulta poco menos que imprescindible.

XP es un proceso muy orientado a la implementación, en el que se genera poca documentación y en que la funcionalidad exacta del sistema final no se define nunca formal y contractualmente. Es por eso que este método es más aplicable para desarrollos internos.

Dadas las características del proyecto, se decidió utilizar a RUP, por su carácter general y porque las demás metodologías estudiadas presentan ciertas particularidades que no puede asumir el proyecto.

2.11 Propuesta.

Después del análisis realizado en el capitulo y dados los requerimientos por parte del protocolo la propuesta a desarrollar consiste en una aplicación Web, que atienda las solicitudes entrantes por parte del proveedor de servicio, que preste una interfase que permita un servicio de entrada de metadatos, utilizando como lenguaje del lado del cliente a JavaScript del lado del servidor a Java, el cual posibilita la utilización del Framework de persistencia Hibernate, permitiendo este la utilización de cualquier gestor de base de datos y como servidor Web se utilizó al Apache Tomcat. Como metodología de desarrollo se utilizará RUP.

2.12 Conclusiones.

En el presente capítulo se analizó las tendencias y tecnologías a utilizar en el desarrollo de la propuesta de solución. Se fundamentaron las decisiones tomadas sobre el lenguaje, el gestor de base de datos, el tipo de servidor Web y la metodología de desarrollo de software a utilizar, así como otras características del sistema que serán utilizadas en el transcurso del trabajo.

CAPÍTULO 3

DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

3.1 Introducción.

En este capítulo, se realiza un análisis de la propuesta de solución, para ello se describen las reglas del negocios asociados al objeto de estudio, se plantea el modelo de dominio, debido a la no formación con claridad de un modelo de negocio, permitiendo comprender y describir las clases, entidades y conceptos más importantes dentro del contexto del sistema.

También se plantean los requisitos funcionales y no funcionales, y finalmente se describe el sistema propuesto mediante los modelos de casos de uso del sistema.

3.2 Reglas del Negocio a considerar.

Para describir las reglas del negocio que se relacionan con el trabajo, es necesario centrar la atención en el protocolo OAI-PMH, principalmente en sus especificaciones.

El mismo, como se ha explicado en capítulos anteriores, define un entorno de trabajo, el cual facilita la interoperabilidad entre los proveedores de datos y los de servicio, a través del intercambio automático de registros de metadatos, posibilitando una gran diseminación de la información.

El proveedor de datos, posee información que desea poner a disposición de los diferentes recolectores de datos, mediante un servicio de búsqueda. La misma se encuentra almacenada en un depósito de metadatos y consiste en registros que describen documentos electrónicos: tesis, manuscritos, fotografías, archivos de video y otros. Nótese que en los depósitos no se encuentran los objetos o documentos, solamente las fichas con los metadatos que describen sus características y contenido.

El proveedor de servicio, usa una aplicación (software agent) para recolectar registros de los depósitos y los combina dentro de un servicio unificado o genérico de búsqueda, disponible en una página web. Cuando los usuarios localizan algo a través de uno de estos servicios, son dirigidos directamente al contenido original del objeto.

El intercambio de información se realiza de la siguiente manera:

Partiendo del proveedor de servicios, encontramos una primera capa de aplicación que contiene una implementación del protocolo OAI-PMH y que es la responsable de lanzar una petición, a través de HTTP a repositorios reconocidos como proveedores de datos o *data provider*.

Una vez recibida la petición, se procesa a través de otra capa de aplicación, para realizar una consulta a al base de datos que contiene los metadatos referentes a los recursos que se encuentran en el repositorio. Una vez hecho esto, la aplicación que ha recopilado la información solicitada, compone la respuesta en formato OAI-PMH con sintaxis XML y se la envía nuevamente a través de HTTP.

La capa de aplicación del proveedor de servicio recoge la respuesta y se encarga de introducir en su propia base de datos los archivos recibidos, para componer posteriormente los servicios de valor añadido que presentará al usuario final.

De esta descripción se infiere que las características o reglas que debe cumplir la aplicación son:

- Las peticiones, deben cumplir con las especificaciones del protocolo, debiendo estar formada primeramente por el verbo, que indica el tipo de recolecta y seguido de las características que lo identifican.
- Las respuestas deben cumplir también con el protocolo, y ser enviadas en un formato XML.
- El tránsito de información se realizará, mediante el protocolo HTTP.

3.3 Definición de las entidades y los conceptos principales

Los principales conceptos, en los que se apoya el modelo de dominio son:

- **Registro**: agrupan las características que describen a los documentos almacenados en el servidor.

- **Tipos**: se refiere únicamente a los registros, es un modo de clasificarlos con la intención de agruparlos.
- **Analizador**: es el encargado de desarrollar el análisis del sistema, almacenar los registros, y se apoya en control de flujo para el intercambio de metadatos.
- **Pedido**: Solicitud realizadas al sistema, el mismo puede ser de seis tipos diferentes, GetRecord, ListRecords, ListIdentifiers, ListSets, ListMetadataFormats, Identify.
- **Control de flujo**: encargado de controlar el trámite de respuestas.
- **Generador XML**: confecciona, a partir de un conjunto de registros, un fichero XML el cual es emitido como respuesta a la solicitud entrante.
- **Generador de errores**: confecciona una respuesta en formato XML, en caso que las peticiones entrantes al sistema sean erróneas.

3.4 Modelo de Dominio.

El modelo del dominio se describe mediante diagramas UML, específicamente con un diagrama de clases conceptuales significativas en el dominio del problema.

DIAGRAMA DEL MODELO DEL DOMINIO

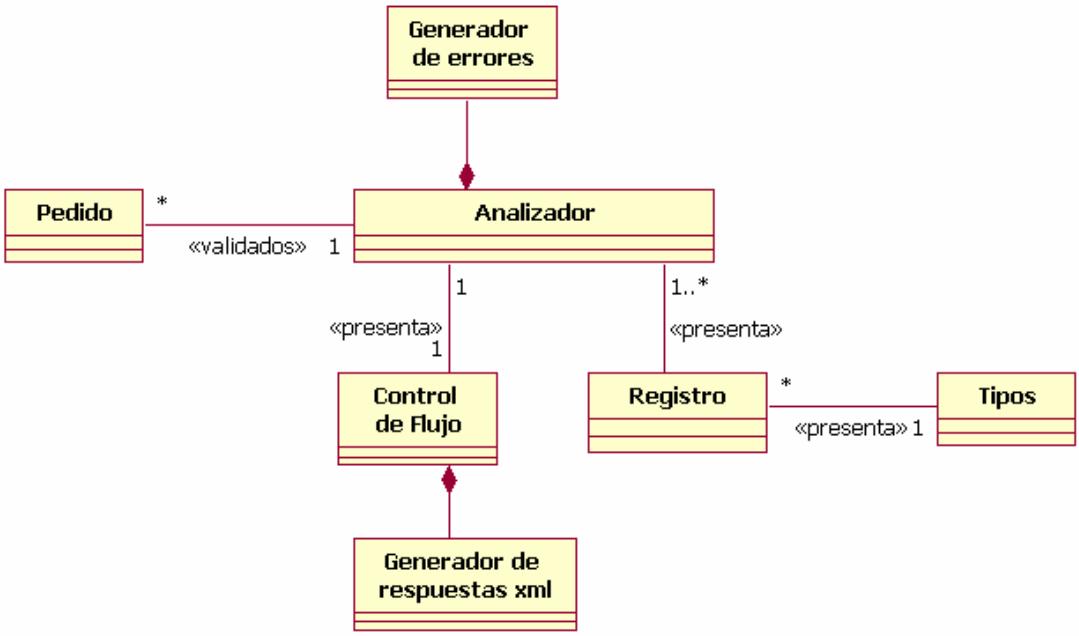

Fig. 3-1. Diagrama del modelo del dominio.

3.5 Requisitos funcionales.

Conocidas las concepciones generales, se analizan las exigencias que describe al sistema propuesto, por lo que a continuación se muestran las condiciones o capacidades que el producto debe cumplir:

R1. Cosechar metadatos.

1.1. Permitir que cualquier Proveedor de servicio, pueda conectarse al sistema y solicitar los metadatos.

R2. Formato de los metadatos.

2.1. Utilizar como formato para la descripción de recursos electrónicos el Dublin Core.

R3. Validación de pedidos.

3.1. Mantener una estricta validación de los pedidos, debido a que sobre ella se basa la cosecha de los metadatos.

R4. Soporte lógico para una jerarquía de conjuntos (sets).

4.1. Permite el mantenimiento de clasificaciones de cara a una ordenación temática o conceptual de los recursos.

R5. Control de flujo.

5.1. Permitir controlar el flujo de transmisión, que se origina de pedidos muy extensos. Este control de flujo se realiza a través de la reanudación de la señal (resumption Token), en la que se basa el diálogo entre el proveedor de metadatos y el de servicio.

R6. Autentificar al Administrador.

6.1. Pedir usuario y contraseña antes de permitirle realizar alguna función.

6.2. Crear un administrador por defecto, con usuario admin y contraseña admin.

6.3. Permitir cambiar el usuario y la contraseña una vez autenticado.

R7. Administrar el sistema.

7.1. Permitir la entrada de nuevos registros.

7.2. Permitir la entrada de nuevos tipos de registro.

7.3. Permitir eliminar un registro de la base de datos, dado su identificador.

7.4. Permitir seleccionar y eliminar un tipo de registro de la base de datos, pero antes debe eliminar todos los registros.

7.5. Permitir verificar que el actor este autentificado.

3.6 Requisitos no funcionales.

Dentro de las propiedades o cualidades que el producto debe tener encontramos:

Usabilidad.

El sistema podrá ser usado por cualquier proveedor de servicio, que cumpla con las especificaciones del protocolo OAI-PMH.

Rendimiento.

Tiempo de respuestas rápidas al igual que la velocidad de procesamiento de la información.

Seguridad.

Garantice una integridad absoluta de los metadatos almacenados.

Identificar al administrador antes de que pueda realizar cualquier acción sobre el sistema.

Garantizar que la información sea dada únicamente por quien la ha solicitado, manteniendo una sección para cada cliente.

Portabilidad.

Independencia de la plataforma.

Confiabilidad.

Tratamiento adecuado de las excepciones y validación de las peticiones entrantes.

Soporte.

Soporte de almacenamiento de metadatos: Un gestor de base de datos.

Soporte Web: Un servidor web.

Hardware.

512 MB de memoria RAM

Apariencia o interfaz externa.

Diseño sencillo, que permita un rápido aprendizaje por parte del usuario.

3.7 Descripción del sistema propuesto.

Para dar cumplimiento a los objetivos trazados al inicio de este trabajo y teniendo en cuenta los requisitos anteriormente planteados se propone el desarrollo de un módulo de cosecha de metadatos, bajo las especificaciones del protocolo OAI-PMH y además se plantea el módulo de administración permitiendo el manejo de los archivos.

Con estas características, el sistema presenta un rol, que sería un proveedor de servicio, encargado de realizar la cosecha en la base de datos y un administrador que controlará todas las entradas y salidas de metadatos.

El módulo de recolecta es el encargado de posibilitar la extracción de los registros almacenados, los mismos deben ser enviados en un formato XML, de forma que pueda ser interpretado por cualquier software que cumpla con las especificaciones del protocolo, sin necesidad de

autentificarse. El objetivo principal que se persigue con el trabajo es compartir la información, por eso no se debe poner restricciones.

El módulo de administración, usado únicamente por el administrador, mantiene actualizada la base de datos, es el encargado de entrar nuevos metadatos al sistema y también tiene la responsabilidad de darle de baja una vez que no tenga sentido tenerlo almacenado.

3.8 Modelo de caso de uso del sistema.

Haciendo uso de las facilidades brindadas por el UML, se plasman los requisitos funcionales del sistema en los diagramas de caso de uso.

Un caso de uso es una descripción de la secuencia de interacciones que se producen entre un actor (agente externo) y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica.

En este caso con el sistema interactúan dos actores que se definen a continuación:

Tabla 3-1. Actores del negocio.

ACTORES	JUSTIFICACIÓN
Proveedor de servicio	Representa a un software que utilizará el sistema para recolectar los metadatos.
Administrador	Representa la persona que controla la entrada y salida de metadatos.

Seguidamente se presentan los casos de uso determinados para satisfacer los requisitos del sistema:

CU-1	Cosechar.
Actor	Proveedor de servicio
Descripción	Mediante la red y el uso del protocolo HTTP el actor hace una solicitud, la cual es atendida por el sistema, auxiliándose del caso de uso de validación comprueba la petición realizada, luego de esta operación extrae los metadatos solicitados y mediante el de control de flujo establece los trámites del envío de la respuesta.
Referencia	R1, R2, R4

CU-2	Validación
Actor	
Descripción	Se inicia a solicitud del caso de uso Cosechar, para verificar que la información entrante esté correcta, en caso de que presente errores se confecciona un XML indicando el tipo de error.
Referencia	R3

CU-3	Control de flujo.
Actor	
Descripción	Se inicia a solicitud del caso de uso Cosechar, para controlar el flujo de información que se origina entre el actor y el sistema, donde se confeccionan la respuestas que se enviaran mediante un fichero XML.
Referencia	R5

CU-4	Agregar Registro.
Actor	Administrador
Descripción	El administrador una vez autentificado, a través de la interfaz, llena las características solicitadas y almacena el registro.
Referencia	R7

CU-5	Eliminar Registro.
Actor	Administrador
Descripción	El administrador, una vez autentificado a través de la interfaz, identifica el registro a eliminar y el sistema le da de baja.
Referencia	R7

CU-6	Autentificar.
Actor	Administrador
Descripción	Para lograr una interacción entre el administrador y el sistema es necesario de su autentificación, entrando el nombre y la contraseña.
Referencia	R6

A continuación se muestra el diagrama donde se representa la relación existente entre los actores y los casos de uso.

DIAGRAMA DE CASOS DE USO DEL SISTEMA

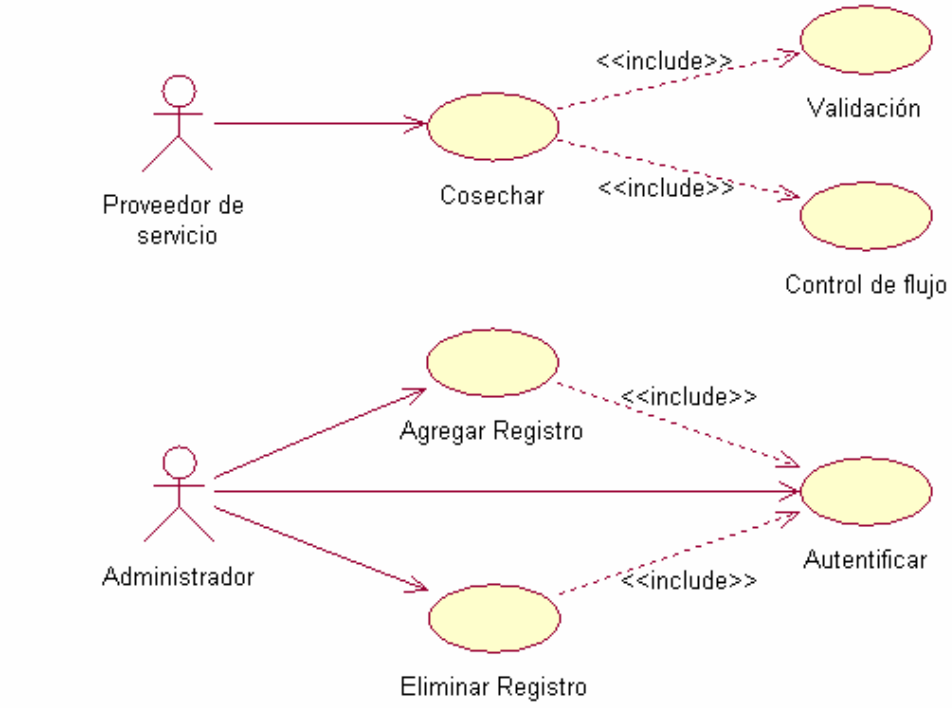

Fig. 3-2. Diagrama de Caso de Uso del Sistema.

3.8.1 Expansión de los Casos de Uso.

Los casos de uso expandidos, posibilitan describir paso a paso la secuencia de eventos que realizan los actores, para completar un proceso a través del sistema. El uso de ello implicaría un fácil entendimiento tanto por los clientes como por los desarrolladores.

Tabla 3-2. Descripción de CU. Cosechar.

<table>
<tr><th colspan="5">Caso de uso</th></tr>
<tr><td colspan="2">CU1</td><td colspan="3">Cosechar.</td></tr>
<tr><td colspan="2">Propósito</td><td colspan="3">Permitir la recolección de metadatos.</td></tr>
<tr><td colspan="5">Resumen: El caso de uso se inicia cuando el proveedor de servicio solicita una cosecha de metadatos determinada, se toman de la base de datos los metadatos pertinentes y se le envían en un formato XML.</td></tr>
<tr><td colspan="2">Referencias</td><td colspan="3">R1, R2, R4</td></tr>
<tr><td colspan="2">Precondición</td><td colspan="3">La solicitud por parte del proveedor de servicio.</td></tr>
<tr><td colspan="2">Acción del actor</td><td colspan="3">Respuesta del sistema</td></tr>
<tr><td>1</td><td>El proveedor de servicio envía una petición a través del protocolo HTTP, solicitando una cosecha determinada de metadatos.</td><td>2

3

4</td><td colspan="2">El sistema valida la petición a través del caso de uso Validación.
Si el resultado es satisfactorio se conecta a la base de datos y extrae los registros solicitados.
Confecciona el resultado y establece el trámite de la información mediante el caso de uso Control de Flujo.</td></tr>
<tr><td colspan="5">Flujo alternativo</td></tr>
<tr><td colspan="2">Acción del actor</td><td colspan="3">Respuesta del sistema</td></tr>
<tr><td>Paso 1</td><td colspan="2">Si el pedido no es válido.</td><td colspan="2">Se envía un fichero XML indicando el error ocurrido.</td></tr>
<tr><td colspan="2">Poscondición</td><td colspan="3">En caso que la respuesta sea muy extensa el sistema mantiene la sesión por si el actor solicita los demás restantes registros.</td></tr>
</table>

Tabla 3-3. Descripción de CU. Validación.

Caso de uso			
CU2	**Validación.**		
Propósito	Permite verificar que las solicitudes sean correctas.		
Resumen: Se inicia a solicitud del caso de uso Cosechar, el cual le envía los datos a verificar, en caso que estos datos tengan algún error, ya sea desde el punto de vista sintáctico como semántico, se retorna la especificación del error en formato XML.			
Referencias	R3, R4		
Precondición	El actor debió haber enviado una solicitud de cosecha de metadatos al sistema.		
Acción del actor		**Respuesta del sistema**	
1	Envía los datos a verificar.	2 3	El sistema toma los datos y los procesa. De estar correcto, envía la certificación.
Flujo alternativo			
Acción del actor		**Respuesta del sistema**	
Paso 1	Si los datos son erróneos.	Se genera un fichero de error en formato XML.	
Poscondición	Se realiza la cosecha o se genera un mensaje de error.		

Tabla 3-4. Descripción de CU. Control de flujo.

Caso de uso			
CU3	**Control de flujo.**		
Propósito	Permitir el intercambio de metadatos entre los sistemas.		
Resumen: Se inicia a solicitud del caso de uso Cosechar, el cual con los registros ya recolectados, le pide establecer el control de la información a transmitir.			
Referencias	R5		
Precondición	Haber realizado la extracción de los elementos de la base de datos.		
Acción del actor		**Respuesta del sistema**	
1	Envía los registros ya cosechados.	2 3	Recibe los registros y los analiza. Confecciona el fichero XML, en correspondencia al tipo de pedido realizado y lo retorna.
Flujo alternativo			
Acción del actor		**Respuesta del sistema**	
Paso 1	Si el pedido no provocó una recolecta de datos.	Se genera un fichero en formato XML indicando que no hay datos al pedido solicitado.	
Poscondición			

Tabla 3-5. Descripción de CU. Agregar Registros.

Caso de uso			
CU4	**Agregar Registros.**		
Propósito	Incorporar nuevos registros en la base de datos.		
Resumen: El caso de uso se inicia cuando el actor intenta agregar un registro, para ello, debe entrar los datos que lo caracterizan.			
Referencias	R7		
Precondición	El usuario este autenticado.		
Acción del actor		**Respuesta del sistema**	
1	Solicitud por el administrador, de agregar un registro.	2	Se verifica que el autor este autentificado y en ese caso se muestran los controles necesarios para que el usuario introduzca las características correspondientes al registro.
3	Introduce las características del registro.	4	Se insertan en la base de datos.
Flujo alternativo			
Acción del actor		**Respuesta del sistema**	
Paso 1	En caso que el usuario no este autenticado.	Se envía al usuario a autenticarse.	
Poscondición	En la base de datos queda registrado un nuevo elemento.		

Tabla 3-6. Descripción de CU. Eliminar Registro.

Caso de uso			
CU5	**Eliminar Registro.**		
Propósito	Permitir eliminar los registros existentes en la Base de Datos.		
Resumen: El caso de uso se inicia cuando el actor intenta eliminar un registro, para ello debe entrar la característica que lo identifica y el sistema lo elimina.			
Referencias	R6		
Precondición	El usuario este autenticado.		
Acción del actor		**Respuesta del sistema**	
1	Solicitud por el administrador, de eliminar un registro.	2	Se verifica que el autor este autenticado y en ese caso se muestra los controles necesarios para que el usuario introduzca la característica que identifica al registro.
3	Introduce la característica que identifica al registro.	4	Se elimina el registro de la base de datos.
Flujo alternativo			
Acción del actor		**Respuesta del sistema**	
Paso 1	En caso que el usuario no este autenticado.	Se envía al usuario a autenticarse.	
Paso 2	Si el registro no existe.	El sistema muestra un mensaje de error.	
Poscondición	Es eliminado un registro de la base de datos.		

Tabla 3-7. Descripción de CU. Autentificar.

Caso de uso			
CU6	**Autentificar**		
Propósito	Permitir el acceso al sistema, únicamente a la persona autorizada.		
Resumen: El caso de uso se inicia cuando el usuario envía al sistema sus datos (nombre y contraseña). En caso que los datos sean correctos, el usuario puede realizar cualquier operación permitida por el sistema.			
Referencias	R6		
Precondición	Que el actor trate de realizar alguna función sobre el sistema sin estar autentificado.		
Acción del actor		**Respuesta del sistema**	
1 3	El actor intenta autentificarse. Introduce usuario y contraseña	2 4	Muestra la página de autentificación. El sistema verifica la entrada de los datos.
Flujo alternativo			
Acción del actor		**Respuesta del sistema**	
Paso 1	En caso que el usuario y la contraseña sean incorrectas.	Se muestra en mensaje de error.	
Poscondición	Se ha autentificado el administrador al sistema.		

3.9 Conclusiones.

En este capítulo se realizó una descripción general de la solución propuesta, donde se planteó el modelo de dominio y los requisitos que debería cumplir el sistema, estos últimos quedaron plasmados, en los Diagramas de Casos de Uso, los cuales fueron expandidos para describir paso a paso todas las acciones de los actores del sistema.

Luego de esta descripción, están las condiciones creadas para pasar a la fase de construcción de la solución propuesta, el cual será abordado en el capitulo siguiente.

CAPÍTULO

CONSTRUCCIÓN DE LA SOLUCIÓN PROPUESTA

4.1 Introducción.

En este capítulo se exponen los modelos y diagramas relacionados, con la construcción de la solución propuesta. Primeramente se presenta el diagrama de clase, facilitando la representación del sistema propuesto a partir de la cual los desarrolladores podrán trabajar. Seguidamente se muestra el diseño de la base de datos mediante el diagrama de clases persistentes y el modelo de datos.

También se describen los principios del diseño de la aplicación, referentes a estándares de interfaz, tratamiento de excepciones y estándares de codificación.

4.2 Diagramas de clases.

Los diagramas de clases para las Aplicaciones Web, difiere un poco de los diagramas estándares debido a que estos se centran más en la modelación de la lógica y estado del negocio que de los detalles de presentación. Tal motivo provocó que los especialistas de Rational crearan una extensión para UML que atendiera este tipo de problema, dentro de ellos se encuentran el modelado de las páginas y los enlaces entre ellas.

De acuerdo a la forma en que se ha organizado el contenido del trabajo y con el fin de lograr una mayor comprensión, se consideró conveniente presentar los diagramas de clase organizados por subsistemas.

Un primer subsistema dedicado a la recolección de metadatos, que agrupa los casos de uso de Cosecha, Control de Flujo y de Generación de Errores, y un segundo que describe la

administración del sistema, agrupando los casos de usos de Agregar, Eliminar y el de Administración.

Para una mejor comprensión de los modelos se explicará el propósito de las principales clases.

Tabla 4-1. Descripción de las principales Clases.

Clase	Propósito
Analizador	Encargado de analizar todos los pedidos que entran al sistema y en dependencia de ello, determina cual acción ejecutar.
Generador de Errores	Cuando en el transcurso de la operación ocurre un error contemplado en el protocolo esta es la clase encargada de elaborar el fichero XML, que informa el problema ocurrido.
Control de Flujo	Encargado de elaborar los fichero XML con los metadatos solicitados, y también incluye información referente a la cantidad de datos pendientes a transmitir.
ObjectServicie	Encargada de establecer todos los servicios de extracción y de asignación de objetos a la base de datos.
Formato	Es la clase es la representación del formato Dublin Core.
TipoSet	Es la que caracteriza a las instancias de la clase anterior.
AdicionarEliminar	Encargada de controlar la entrada y eliminación de objetos.

A continuación se presentan los diagramas que representan los procesos más importantes.

4.2.1 Subsistema de cosecha de metadatos.

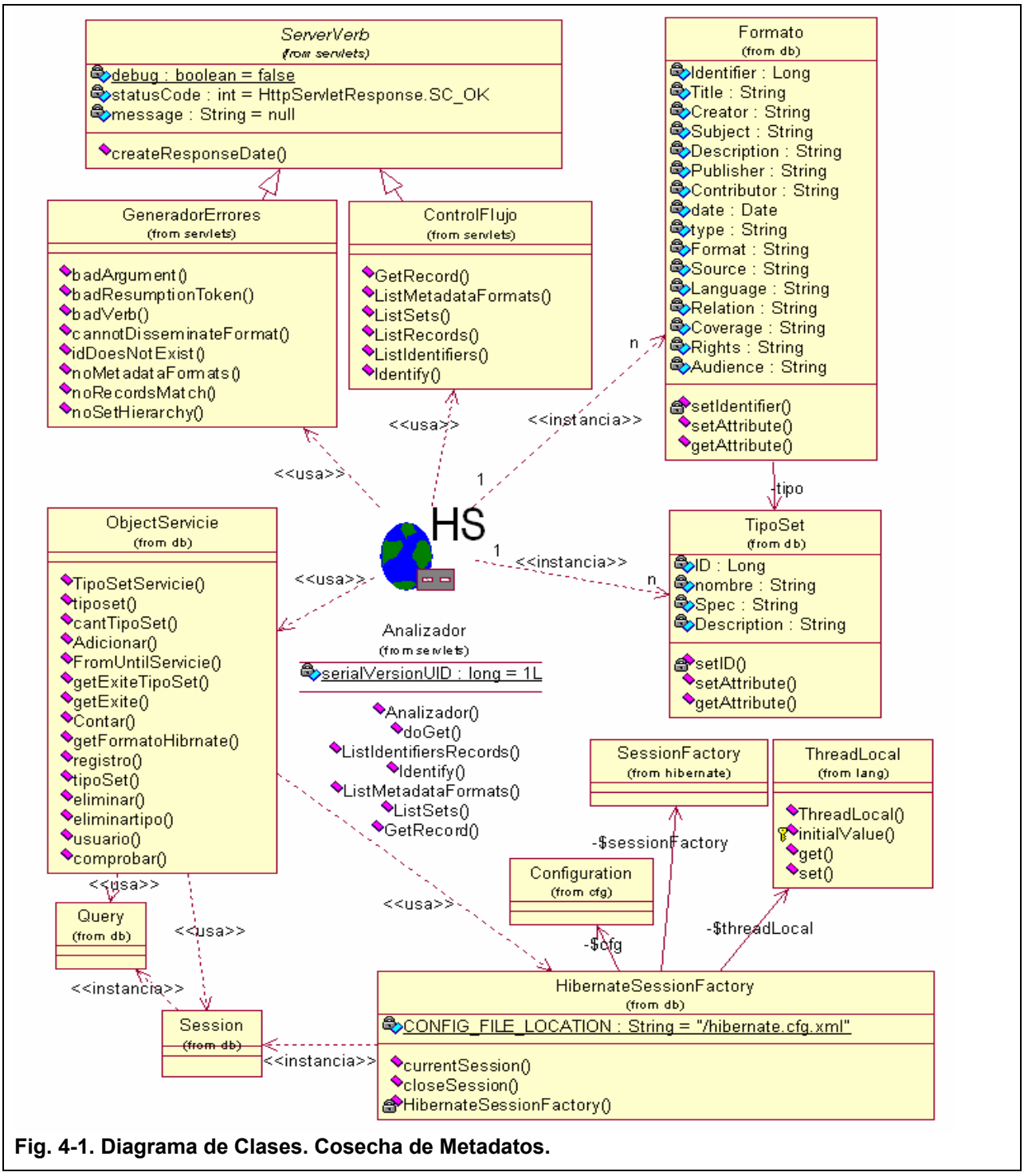

Fig. 4-1. Diagrama de Clases. Cosecha de Metadatos.

4.2.2 Subsistema de administración.

Para lograr una mejor visualización del diagrama se determino dividirlo en dos partes, una para la presentación y otro para las clases referentes al sistema.

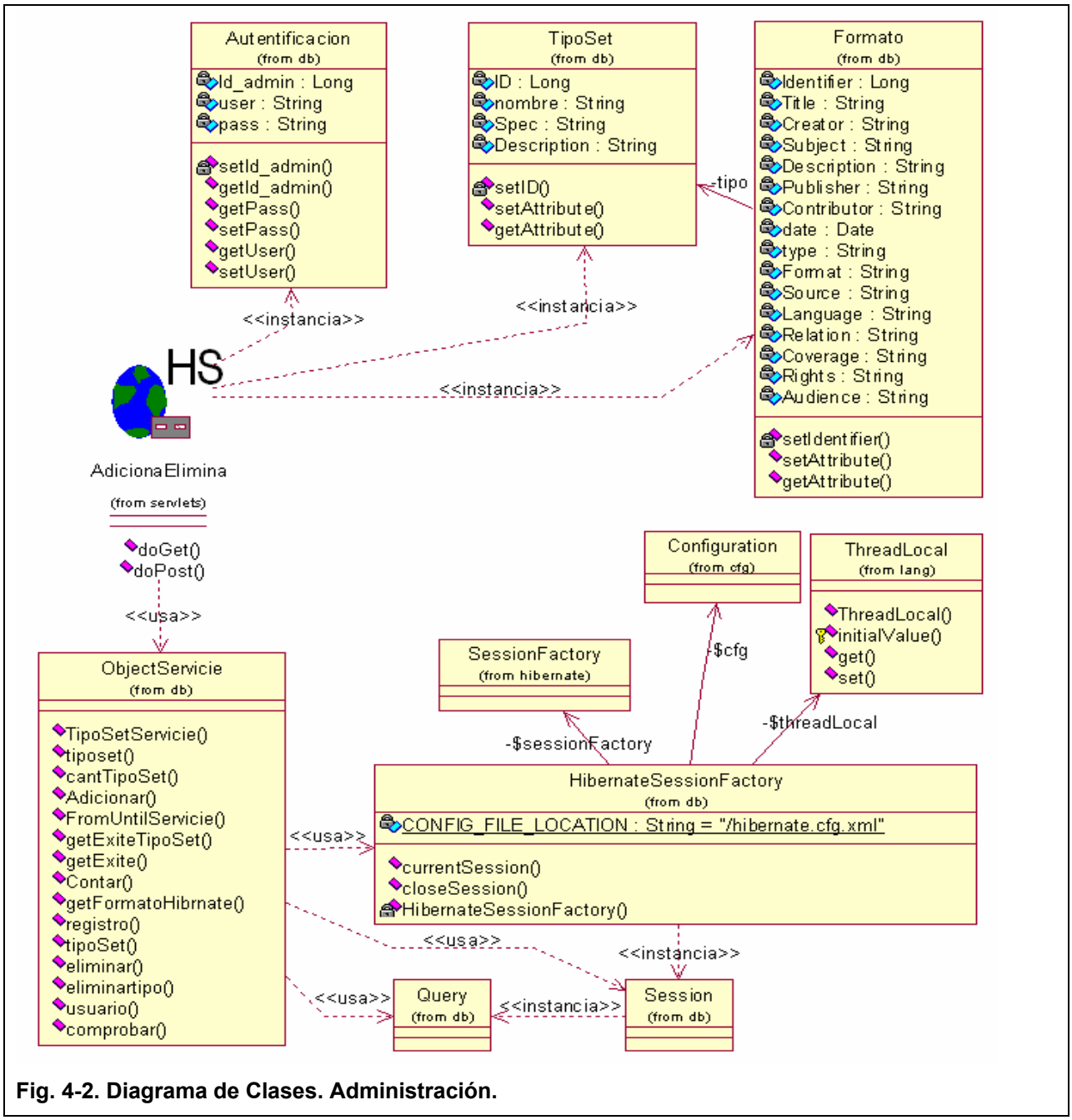

Fig. 4-2. Diagrama de Clases. Administración.

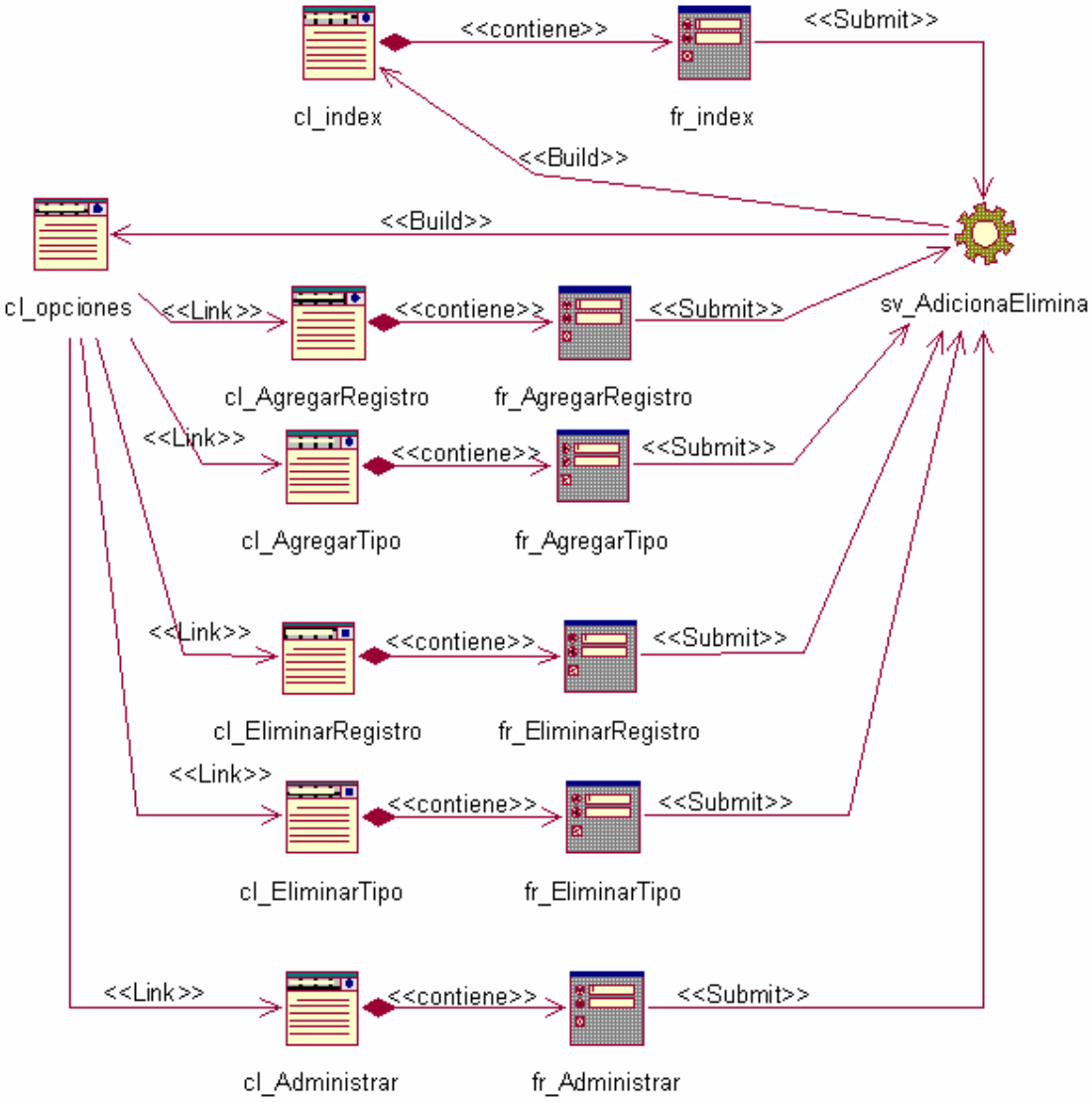

Fig. 4-3. Diagrama de Clases. Presentación.

En el ANEXO I se presentan cada uno de los diagramas de secuencias que describen las operaciones correspondientes a los diagramas de clases planteados.

4.3 Diseño de la base de datos.

Para lograr un diseño correcto de la base de datos, se utilizó el diagrama de clases persistentes y el modelo de datos.

Las clases que se pueden representar a través de estos diagramas, son las que persistan, es decir las que representan los datos que se obtiene y almacena durante los procesos de la aplicación.

4.3.1 Diagrama de clases persistentes.

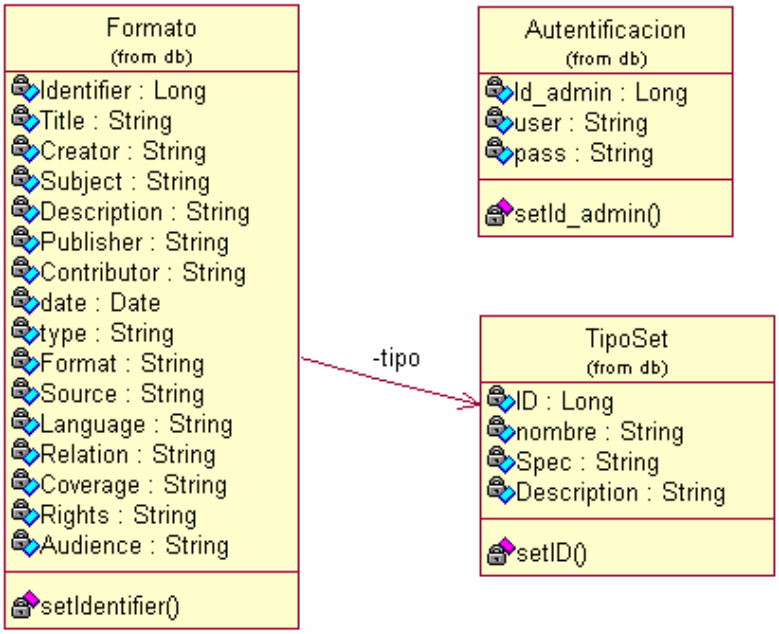

Fig. 4-4. Diagrama de Clases Persistentes.

4.3.2 Modelo de datos

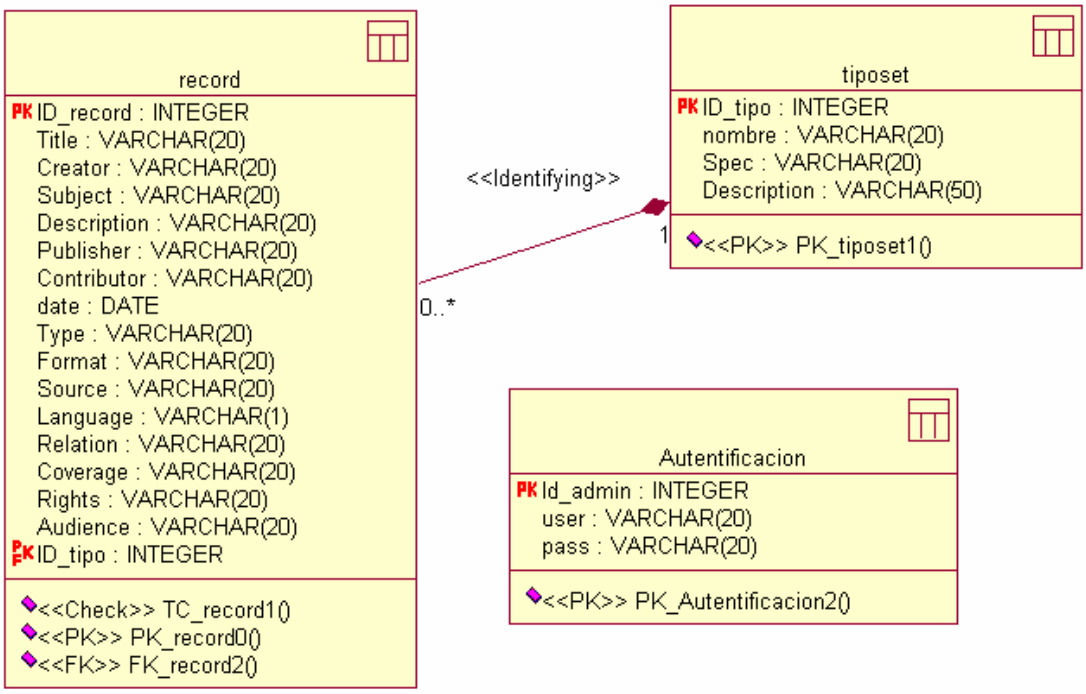

Fig. 4-5. Modelo de Datos.

4.4 Principios de diseño de interfaz.

Internet es un medio que brinda muchas posibilidades para todos los usuarios de la red por su gran volumen de información. Esto provoca que la competitividad, para las aplicaciones Web sea cada vez mayor. Por tal razón, lograr que un sitio cumpla con los principios de diseño es muy importante si se quiere lograr un resultado exitoso.

Dentro de estos principios encontramos:

1. Uso equiparable: El diseño es útil y vendible a personas con diversas capacidades.
2. Uso flexible: El diseño se acomoda a un amplio rango de preferencias y habilidades individuales.
3. Simple e intuitivo: El uso del diseño es fácil de entender, atendiendo a la experiencia, conocimientos, habilidades lingüísticas o grado de concentración actual del usuario.

4. Información perceptible: El diseño comunica de manera eficaz la información necesaria para el usuario, atendiendo a las condiciones ambientales o a las capacidades sensoriales del usuario.
5. Con tolerancia al error: El diseño minimiza los riesgos y las consecuencias adversas de acciones involuntarias o accidentales.
6. Tamaño y espacio para el acceso y uso: Que proporcione un tamaño y espacio apropiados para el acceso, alcance, manipulación y uso, atendiendo al tamaño del cuerpo, la postura o la movilidad del usuario.

4.4.1 Estándares de la interfaz de la aplicación.

Antes de entrar en detalles, se debe plantear que la aplicación por ser la implementación del proveedor de datos, no requería de una interfaz.

Pero el equipo de trabajo determino crear una interfaz sencilla de administración, la cual permitiría la entrada y eliminación de la información.

La misma constaría con un esquema Cabecera-Navegador-Contenido. En el navegador se encuentran las acciones a realizar, permitiendo desde cualquier pagina ejecutar una operación en especifico y en el área del contenido se muestran los formularios de entrada, y de selección de los elementos a eliminar.

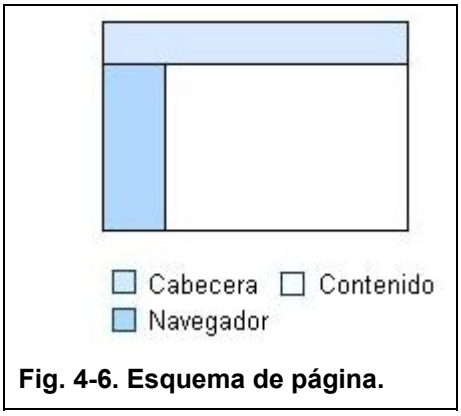

Fig. 4-6. Esquema de página.

Es importante aclarar que debido a la sencillez de la interfaz y dada la peculiaridad que el usuario que interactúa con ella, será una persona especializada o al menos familiarizada con las

características del protocolo, se determino no incorporar la ayuda, por ser acciones tan elementales e intuitivas.

4.4.2 Tratamiento de excepciones.

Los errores que puedan ocurrir durante la puesta en marcha de la aplicación son considerados por las especificaciones del protocolo, en ese caso se confeccionaría un fichero XML, el cual llevaría implícito la causa de su origen, y es enviado al cliente.

Por otra parte los errores ocurridos mediante la entrada y eliminación de datos son tratados de dos formas diferentes. Una mediante JavaScript, validando todas las posibles entradas mostrando con mensajes los errores al cliente, y otra cuando el error es originado del lado del servidor, los cuales son controlados por la aplicación para que la ocurrencia de ellos no implique la salida de funcionamiento del proveedor de datos. En estos casos se muestra un mensaje al usuario indicando el problema ocurrido y permitiéndole seguir interactuando con el sistema.

4.5 Estándares de codificación.

Con el propósito de hacer el código uniforme y mejorar los rendimientos de la aplicación, se determinó seguir los estándares de codificación; estos no son más que reglas específicas a una lengua, que reducen perceptiblemente el riesgo de que los desarrolladores introduzcan errores. Es notable destacar que los estándares de codificación no destapan problemas existentes, más bien evitan que los errores ocurran.

Para ello se utilizó la programación orientada a objeto, creando clases para diferentes funcionalidades y se tomo como acuerdo comenzar sus nombres con mayúscula. Se utilizo una clase controladora en cada uno de los subsistemas planteados, así como una que controlara el manejo de los metadatos en la base de datos permitiendo un uso controlado de la información almacenada.

Se determino comentar cada una de las clases, así como sus métodos permitiendo una mejor comprensión de su funcionalidad, lo cual ayudaría en tiempos futuros a realizar cambios en su estructura.

4.6 Modelo de despliegue.

El modelo de despliegue describe la distribución física del sistema, muestra como están distribuidos los componentes de software entre los nodos de cómputo.

El diagrama de despliegue de la aplicación se muestra a continuación.

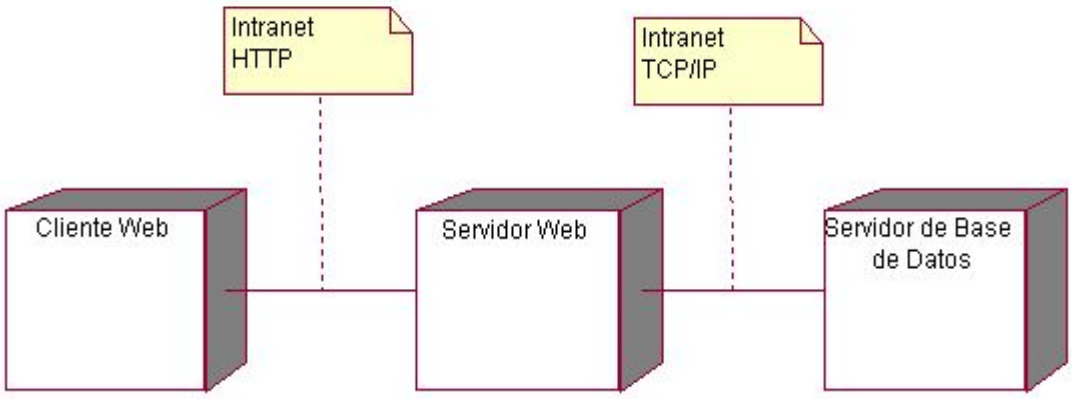

Fig. 4-7. Diagrama de despliegue.

La solución propuesta se basa en una arquitectura de tres capas, a través de las cuales se distribuye toda la aplicación.

Capa de presentación, contiene los componentes de Interfaz de Usuario que garantizan la administración de la aplicación.

Capa empresarial, dedicada a garantiza la lógica del negocio de la aplicación.

Capa de acceso a datos, presenta los componentes que acceden a los datos para realizar consultas y operaciones de inserción, eliminación y modificación.

Teniendo en cuenta lo planteado, el diagrama de despliegue del sistema representa tres nodos. Uno de estos es el Cliente Web, que representa el ordenador del administrador, el cual mediante el protocolo HTTP podrá acceder a la aplicación que se encuentra publicada en el Servidor Web, donde están ubicados todos los componentes de las Capas de Presentación, Empresarial y de

Acceso a Datos, es importante aclarar que el servidor atiende también las peticiones del proveedor de servicio solo que este no necesita de una interfaz.

El Servidor Web se comunica con el Servidor de base de datos (con MySQL como sistema gestor de base de datos), a través del protocolo TCP/IP para realizar consultas y actualizaciones de la información que manipula el sistema.

4.7 Modelo de implementación.

El modelo de implementación, describe cómo los elementos del modelo del diseño se implementan en términos de componentes. Seguidamente se muestra el diagrama de componentes, así como una vista detallada de cada uno de los paquetes, los cuales se encuentran separados con el objetivo de lograr una mayor claridad y comprensión.

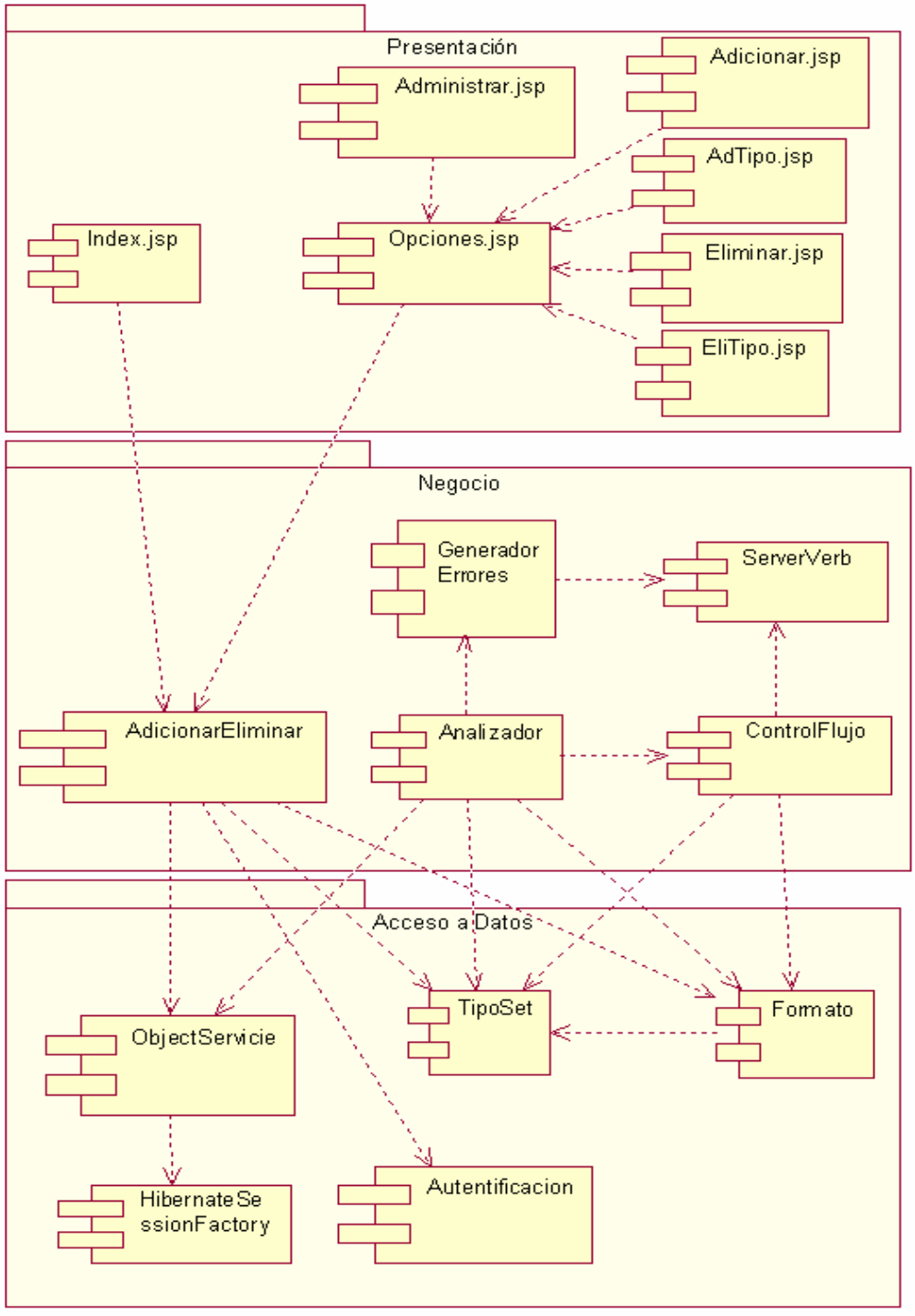

Fig. 4-8. Diagrama de componentes.

4.8 Conclusiones.

En el presente capítulo, se realizó una descripción detallada de la solución propuesta la cual se llevo a cabo mediante los diagramas de clases y los de secuencias correspondientes a las operaciones realizadas por el sistema. Se elaboro el diseño de la base de datos a través del diagrama de clases persistente y del modelo de datos. También se analizó brevemente los principios de diseño enfatizando en la interfaz de la aplicación y en los tratamientos de las excepciones.

Finalmente se planteo el modelo de despliegue y el de implementación a partir del diagrama de componente.

CAPÍTULO 5

ESTUDIO DE FACTIBILIDAD

5.1 Introducción.

En el presente capítulo se realiza un análisis de los costos y los beneficios que se obtendrán con el desarrollo de la aplicación. Esto se hace con el objetivo de demostrar la factibilidad del proyecto.

5.2 Estimación de costo.

Existen diferentes modelos de estimación de los costos de los proyectos de software, entre ellos encontramos los Empíricos, el Cocomo, las Herramientas Automáticas de Estimación entre otros. Uno de los más utilizado en la actualidad es el modelo Cocomo (del inglés Constructive Cost Model). Esta metodología expresa el esfuerzo de desarrollo en términos de Personas Mes y realiza una estimación del costo monetario del proyecto teniendo en cuenta el tiempo estimado de desarrollo, la cantidad de personas involucradas y el salario de estas personas.

Dadas las características que presenta el trabajo, al ser un sistema desarrollado en el entorno de la Universidad de las Ciencias Informáticas, no requiere del cálculo del costo monetario, pues la elaboración del sistema no produjo gastos adicionales por concepto de salario para la institución, debido a que cuenta con el capital humano necesario (estudiantes y profesores). Otro de los beneficios de haber elaborado el software en la Universidad, fue el hecho de contar con la infraestructura tecnológica necesaria para llevar a cabo el proyecto.

Se realizó un estudio del esfuerzo necesario para el desarrollo del sistema, el cual se muestra a continuación.

Tabla 5-1. Entradas externas.

Nombre de la entrada externa	Cantidad Ficheros	Cantidad Elementos Datos	Clasificación
Agregar Registro	1	15	Simple
Agregar Tipos de Registros	1	3	Simple
Eliminar Registros	1	15	Simple
Eliminar Tipos de Registros	1	3	Simple

Tabla 5-2. Peticiones.

Nombre de la petición	Cantidad Ficheros	Cantidad Elementos Datos	Clasificación
Buscar Registro	1	15	Simple
Tipos de registros	1	3	Simple
Lista de Identificadores	2	5	Simple
Lista de Registros	2	15	Media

Tabla 5-3. Ficheros lógicos internos.

Nombre del fichero interno	Cantidad Registros	Cantidad Elementos Datos	Clasificación
Formato	1	15	Simple
TipoSet	1	3	Simple
Autentificar	1	2	Simple

Tabla 5-4. Puntos de función desajustados.

Elementos	Simple		Media		Compleja		Subtotal puntos de función
	No	x Peso	No	x Peso	No	x Peso	
Ficheros lógicos internos	3	7	0	10	0	15	21
Entradas externas	4	3	0	4	0	6	12
Peticiones	3	3	1	4	0	6	13
Total (UFP)							46

Tabla 5-5. Líneas de instrucciones fuentes.

Características	Valor
Puntos de función desajustados	46
Lenguaje(s)	Java (80%) SQL (20%)
Instrucciones fuentes por puntos de función	(63)

	(39)
Instrucciones fuentes por lenguaje (miles de instrucciones)	(2318,4) (460.2)
Instrucciones fuentes (miles de instrucciones)	2,7786

Tabla 5-6. Factores de escala.

Factor	Clasificación	Valor
PREC	Nominal	3,72
FLEX	Nominal	3,04
TEAM	Muy Alto	1,10
RESL	Nominal	4,24
PMAT	Bajo	6,24
Suma		**18,34**

Tabla 5-7. Multiplicadores de esfuerzo.

Multiplicador	Clasificación	Valor
RELY	Nominal	1,00
DATA	Nominal	1,00
CPLX	Nominal	1,00
RUSE	Nominal	1,00
DOCU	Nominal	1,00
PERS	Nominal	1,00
SCED	Nominal	1,00
Producto		**1,00**

La ecuación que plantea COCOMO para calcular el esfuerzo de desarrollo es la siguiente:

$$PM = A \times Size^{E} \times \prod_{i=1}^{n} EM_i$$

donde $E = B + 0.01 \times \sum_{j=1}^{5} SF_j$

E = 0,91 + 0,01 * 18,34 = 1.0934

PM = 2,94 * $(2,7786)^{1,0934}$ * 1.00 = 8,98 **≈ 9 Personas – Mes**

5.3 Beneficios tangibles e intangibles.

El sistema fue diseñado con el objetivo de adaptarse fácilmente a las exigencias del cliente, siempre que estén contempladas en las especificaciones del protocolo OAI-PMH.

Esta flexibilidad se pone de manifiesto en varios aspectos, uno de ellos es el hecho de poder disponer la base de datos en la mayoría de los gestores de base de datos, siendo este uno de los criterios más variables en los clientes.

Otra de las características favorables es poder incorporar nuevas concepciones al sistema sin necesidad de crear otra aplicación. Dentro de ellas encontramos la peculiaridad de adicionar nuevos verbos a partir de los cuales se puedan realizar búsquedas más óptimas. Esta adición no provocaría deshacer lo hecho hasta el momento sino incorporar los nuevos cambios que implicaría su uso.

El trabajo fue realizado de esa manera debido a que se rige por las especificaciones del protocolo y si este introduce cambios, pues el sistema debe adaptarse a ellos.

Desde el punto de vista de funcionamiento la aplicación contribuye a efectúa buscadas específicas más óptimas que los buscadores tradicionales, ya que dispone de la información clasificada.

Como beneficio intangible se destacan los conocimientos adquiridos durante la realización de la aplicación.

5.4 Análisis de costo / beneficio.

El proveedor de datos no requiere de inversión en software ya que todas las herramientas empleadas en su desarrollo, así como las librerías y framework utilizadas, no requieren el pago de una licencia para su uso. Por lo que se plantea que el sistema puede ser comercializable.

Por todo lo antes expresado se considera factible el desarrollo de la aplicación y que el esfuerzo de 9 Personas-Mes, está plenamente justificado.

5.5 Conclusiones.

Luego del estudio realizado, a partir del análisis de los costos y beneficios, se llego a la conclusión que es factible el hecho de desarrollar un proveedor de datos, por todos los aportes esperados.

CONCLUSIONES

Para lograr la correcta implementación del proveedor de datos fue necesario:

- Gestionar el acceso a los datos, mediante el Framework de persistencia Hibernate, permitiendo almacenar la información en los principales gestores de base de datos.
- Incorporar una interfaz de administración, que permitiera la adición de nuevos registros y eliminación de los ya existentes
- Describir los diversos archivos electrónicos, utilizando como formato de metadatos a Dublin Core, por su homogeneidad antes las diferentes características.
- Seguir las instrucciones especificadas por el protocolo.

Por tanto se puede concluir que los objetivos trazados han sido cumplidos satisfactoriamente

RECOMENDACIONES

Se recomienda expandir el sistema a soportar otros formatos de metadatos logrando así una mayor funcionalidad del mismo.

También se recomienda implementar el proveedor de servio y de esta manera dar un integro cumplimiento al protocolo OAI-PMAH.

BIBLIOGRAFÍA

[1] Álvarez Marañón, Gonzalo. *Web Seguro.* http://www.iec.csic.es/criptonomicon/ssl.html (15/6/05).

[2] Archive Ouverte en Sciences de l'Information et de la Communication http://archivesic.ccsd.cnrs.fr/ (30/03/06).

[3] Bankhacker.com. *Web Services*. http://web-services.bankhacker.com/ (2/7/05).

[4] Barrueco, José Manuel *OAI-PMH: Protocolo para la transmisión de contenidos en Internet.* http://www.uv.es/=barrueco/cardedeu.doc (30/03/06).

[5] EconWPA http://econwpa.wustl.edu/ (30/03/06).

[6] González, Carlos D. *Curso: Integral de diseño. Programación de sitios dinámicos con MySQL y PHP*. http://www.usabilidadweb.com.ar/x_int.php (15/04/05).

[7] González, Carlos D. *Curso: Sitios Web dinámicos con Base de Datos PostgreSQL y PHP*. http://www.usabilidadweb.com.ar/postgre.php (15/04/05).

[8] *Metainformación - Dublin Core.* http://www.rediris.es/metadata/ (30/03/06).

[9] *Metadatos en Internet*. http://www.biblioarroyo.com.ar/jornadas/metadatos.htm (30/03/06).

[10] *OAI for Beginners - the Open Archives Forum online tutorial* http://www.oaforum.org/tutorial/index.htm (30/03/06).

[11] SOAP versions http://www.w3.org/TR/soap/ (30/03/06).

[12] RePEc (Research Papers in Economics) http://repec.org (30/03/06).

[13] Syntax and Vocabulary of the Academic Metadata Format http://amf.openlib.org/doc/ebisu.html (30/03/06).

[14] *The Open Archives Initiative Protocol for Metadata Harvesting*. http://www.openarchives.org/OAI/openarchivesprotocol.html (30/03/06).

[15] Villalobos, Jorge *Biblioteca*. http://148.201.94.6/biblioteca/oaithemes/ (30/03/06).

[16] Wikipedia, la enciclopedia libre. *MySQL.* http://es.wikipedia.org/wiki/MySQL (15/04/05).

BIBLIOGRAFÍA

1) Booch., Rumbaugh., Jacobson. “El Proceso Unificado de Software”. Edición en español. Madrid, 2000.
2) GAVIN KING, BAUER CHISTIAN “Hibernate In Action“. USA
3) Larman Craig. “UML Y PATRONES Introducción al análisis y diseño orientado a objetos”. Versión en español, Mexico 1999.
4) Schmuller Joseph. “Aprendiendo UML en 24 Horas”. Edición en español. Mexico 2000.
5) Suárez González Hector “Manual Hibernate”. Edición en español 2003

GLOSARIO DE TERMINOS

Se muestra el significado de algunos términos usados en el documento con el objetivo de lograr una mayor comprensión del trabajo:

1. **Colecciones digitales de información:** están formados típicamente por bases de datos distribuidas y heterogéneas; es decir, aquellas que físicamente no están situadas en un mismo lugar y difieren en sus modelos de datos, así como en los métodos de búsqueda y recuperación de información.
2. **Depósito de metadatos:** Es una base de datos en donde se almacenan los registros que contienen los metadatos recolectados de los diferentes proveedores.
3. **Dublin Core:** Es un estándar que define la manera en la que se van a organizar y presentar los datos que describen los recursos informáticos. Cada elemento de Dublin Core es definido como un conjunto de atributos provenientes de ISO/IEC 11179, y éstos están clasificados en tres grupos que indican la clase o el ámbito de la información que se guarda en ellos:
    a. Elementos relacionados principalmente con el contenido del recurso [Título, tema, descripción, fuente, lenguaje, relación, cobertura].
    b. Elementos relacionados principalmente con el recurso cuando es visto como una propiedad intelectual [autor, editor, colaboradores, derechos].
    c. Elementos relacionados principalmente con la instanciación del recurso
4. **Granularidad:** Nivel de complejidad en la descripción del contenido de un objeto o documento.
5. **Interfaz:** Una interfaz es un componente o dispositivo tecnológico que funciona como conexión física o lógica entre dos objetos, espacios, organismos o sistemas.
6. **Interoperabilidad:** La posibilidad que tienen dos sistemas o componentes para intercambiar información. En este caso en particular, se refiere al conjunto de posibilidades definidas en un protocolo para el intercambio de metadatos.

7. **Metadatos:** El término "meta" viene del griego que significa algo tan alto o más fundamental que lo natural. Metadatos son entonces: datos acerca de otros datos.
8. **Proveedor de metadatos:** El proveedor de metadatos está encargado de administrar el sistema (bases de datos), en donde se almacena la información que deberá estar disponible de acuerdo al protocolo de la OAI. Se puede decir también que son datos estructurados que identifican y describen de manera estandarizada las características de un objeto o recurso informático (libros, tesis, fotografías, archivos en audio,) en formato digital o físico.
9. **Proveedor de servicios:** Tiene como tarea enviar las peticiones que genera el protocolo de archivos abiertos a los sistemas encargados de proporcionar los datos. Una vez que es regresado el metadato, este se utiliza para construir servicios de valor agregado.
10. **Protocolo:** Es un método mediante el cual dos sistemas diferentes pueden intercambiar información.
11. **Recolección de metadatos o harvesting:** Recuperación automática de metadatos desde un sitio remoto en este caso un depósito, hecha por una aplicación (software agent) diseñada para tal efecto.
12. **Robot (software agent) :**Un agente artificial que opera en entornos de software como los sistemas operativos o las bases de datos.
13. **XML:** El XML Extensible Markup Language, (Lenguaje Extendible de Marcación), fue desarrollado por un grupo de trabajo (originalemte conocido como el SGM Editorial Review Board), formado bajo los auspicios del World Wide Web Consortium (W3C) en 1996. El XML, describe una clase de objetos llamados documentos XML y también describe parcialmente el comportamiento de los programas que los procesan. XML es una versión "limitada" del Standard Generalized Markup Language(SGML), que separa la estructura del documento de su contenido.

ANEXOS

ANEXO I.

Diagrama de secuencia de la operación GetRecord, encargada de cosechar un registro.

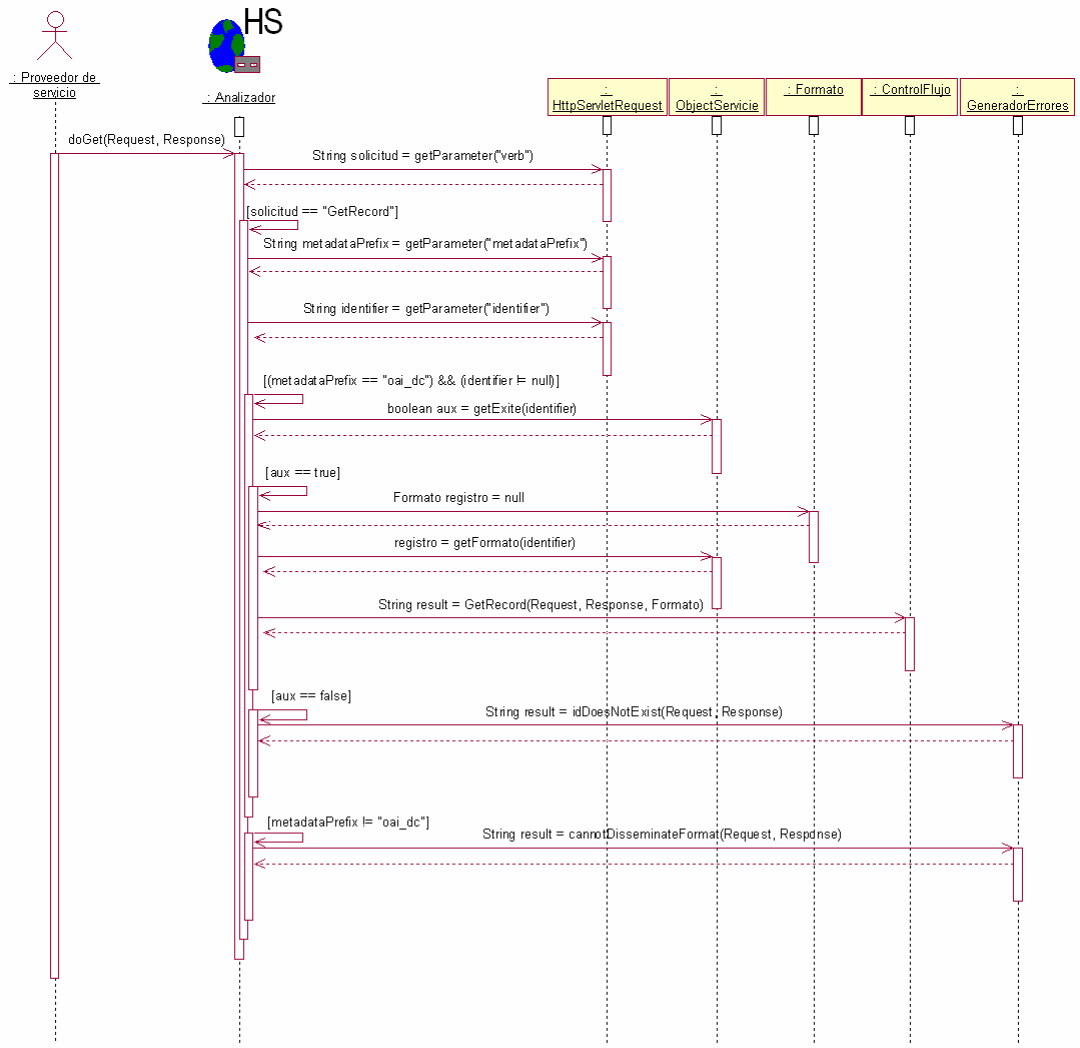

Fig. 6-1. Diagrama de secuencia. GetRecord.

Diagrama de secuencia de la operación Identify, encargada de proporcionar las características del proveedor de datos.

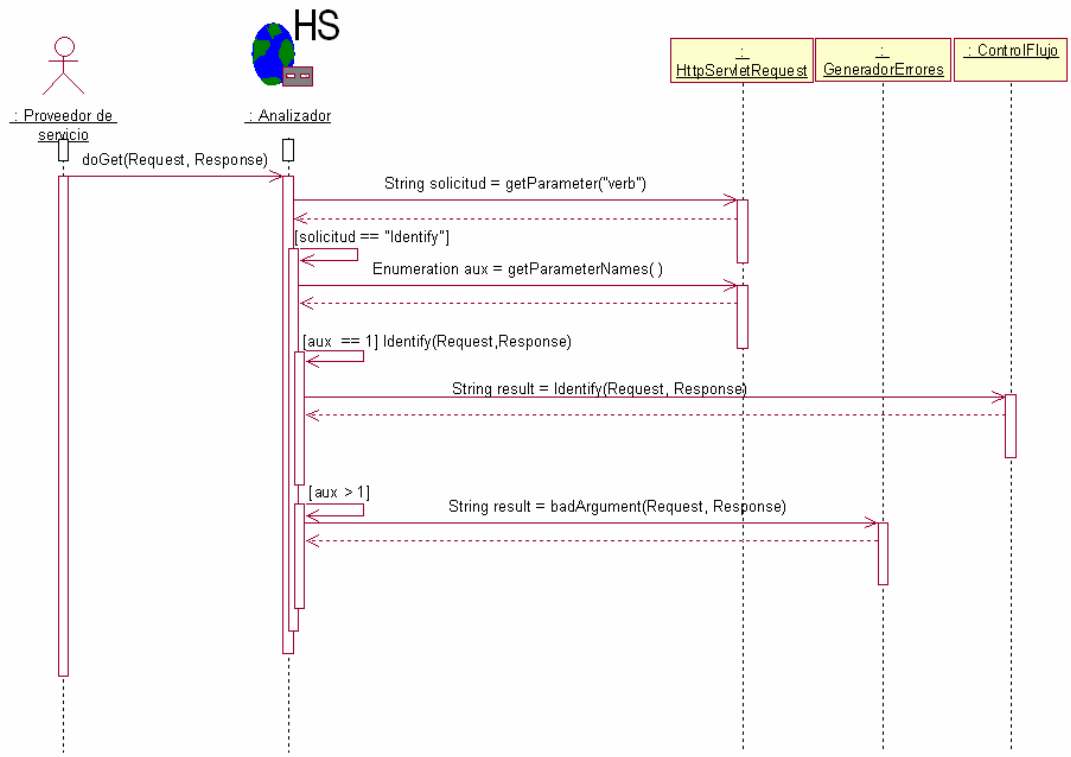

Fig. 6-2. Diagrama de secuencia. Identify.

Diagrama de secuencia de la operación ListMetadataFormats, encargada de proporcionar un listado de los formatos soportados por el proveedor de datos.

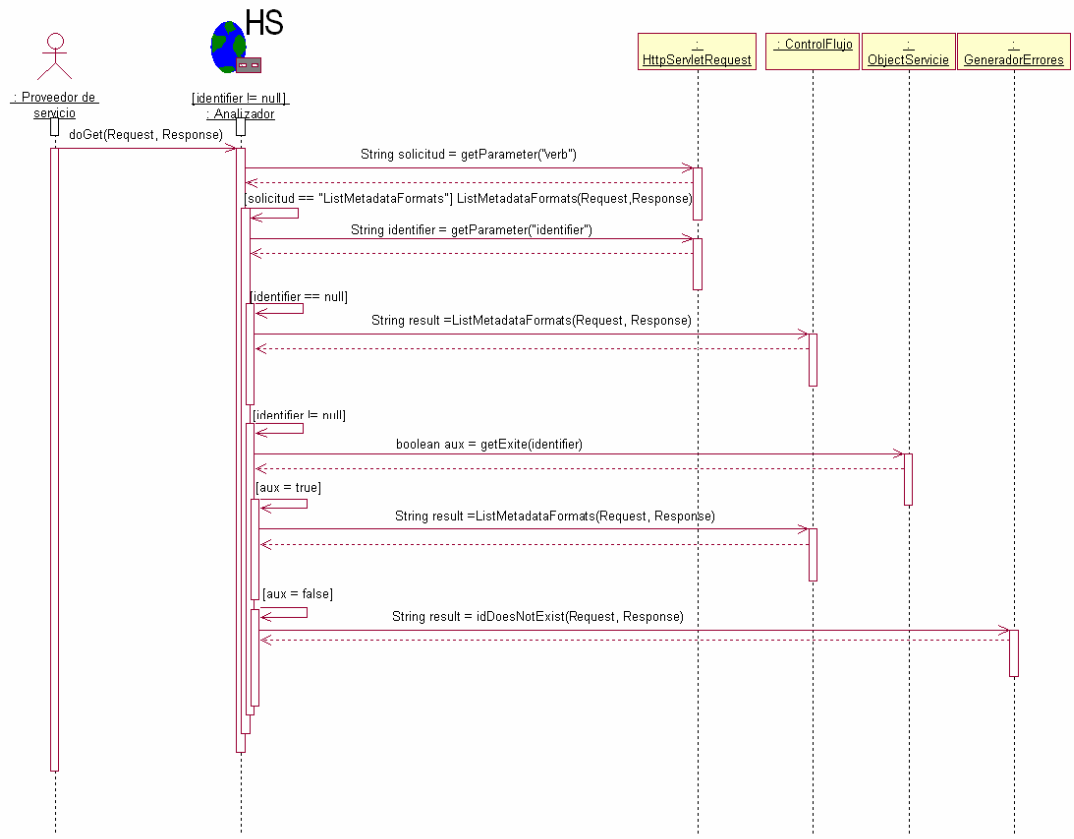

Fig. 6-3. Diagrama de secuencia. ListMetadataFormats.

Diagrama de secuencia de las operaciones ListRecords, encargada de cosechar una lista de registro.

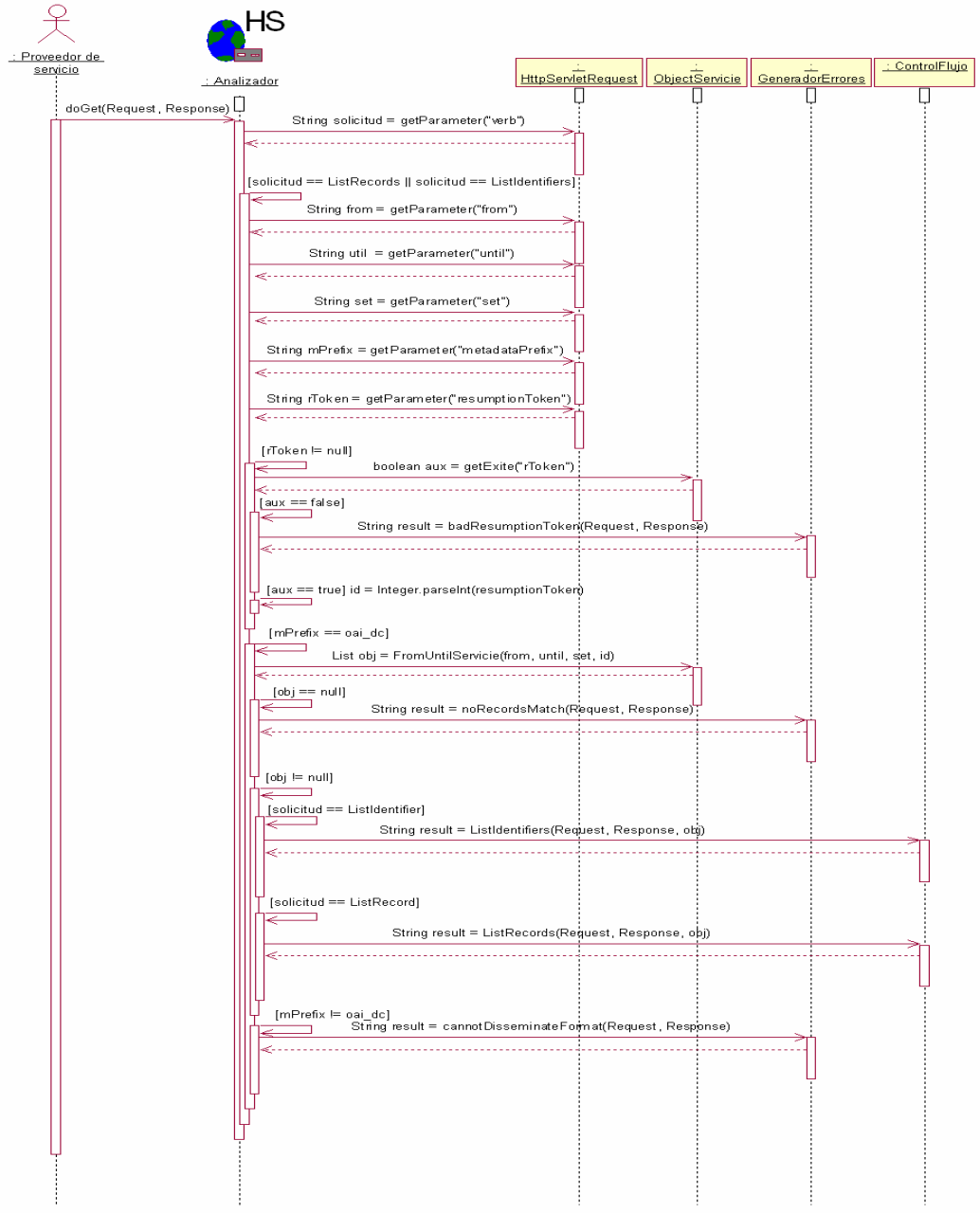

Fig. 6-4. Diagrama de secuencia. ListRecords.

Diagrama de secuencia de la operación ListSets, encargada de cosechar una lista de tipos de registro.

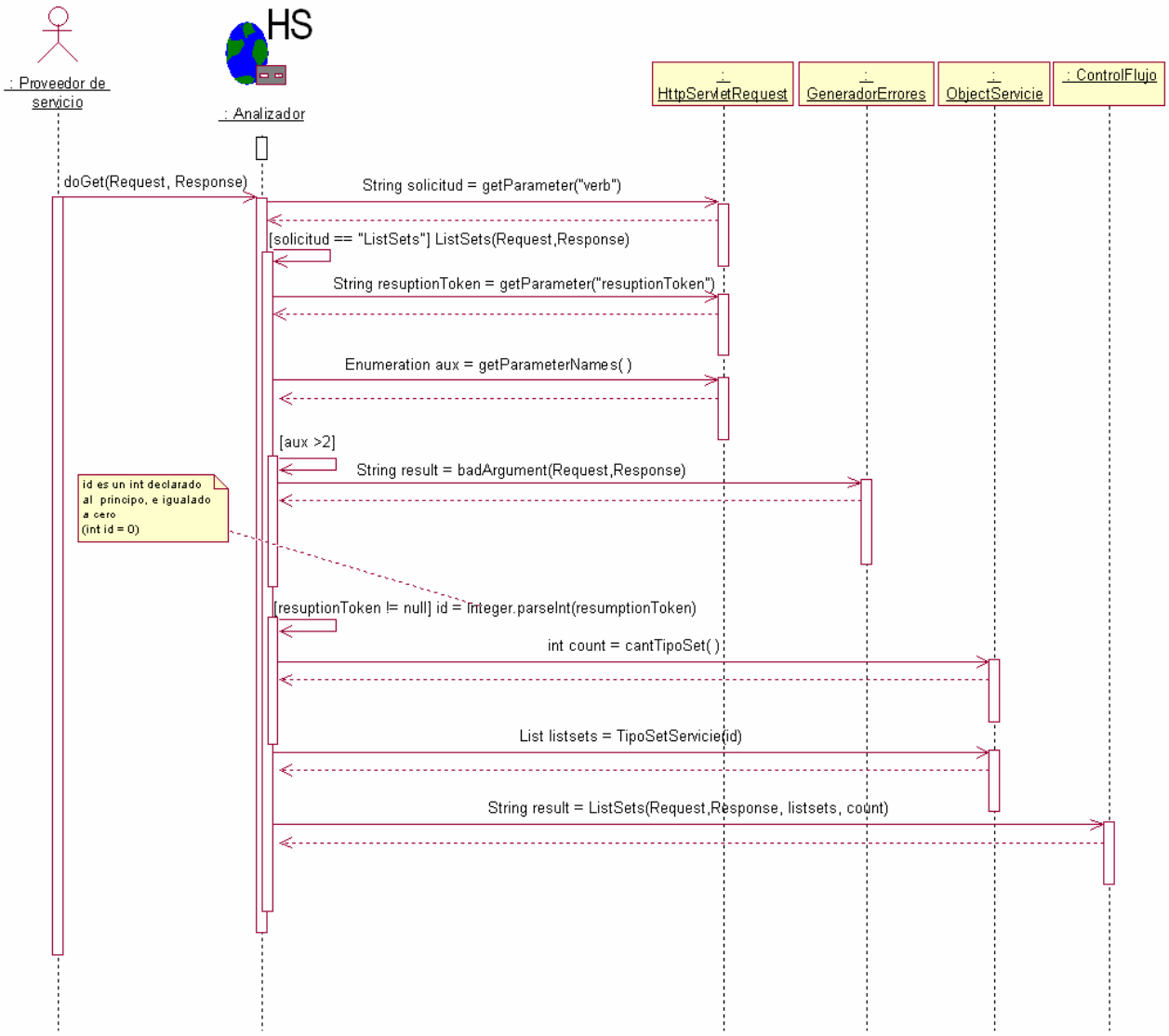

Fig. 6-5. Diagrama de secuencia. ListSets.

Diagrama de secuencia de la operación Adicionar registros en la base de datos.

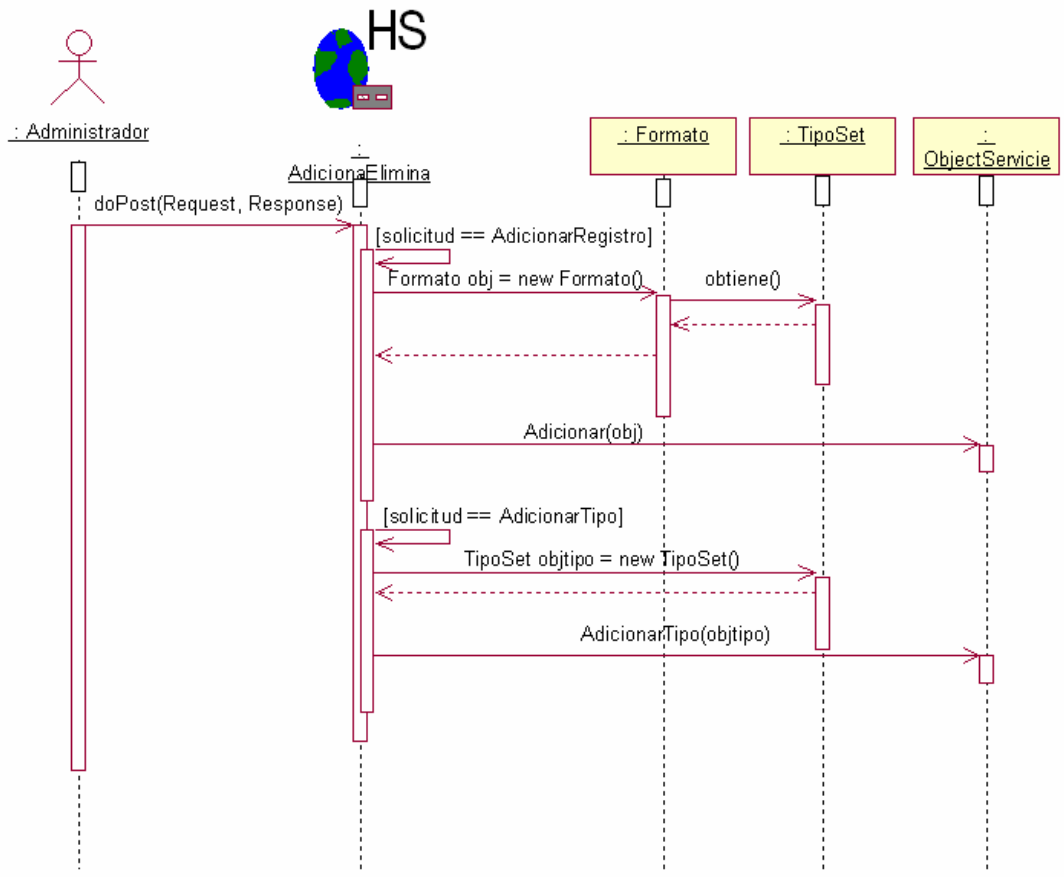

Fig. 6-6. Diagrama de secuencia. Adicionar registros.

Diagrama de secuencia de la operación Eliminar, encargada de eliminar registros de la base de datos.

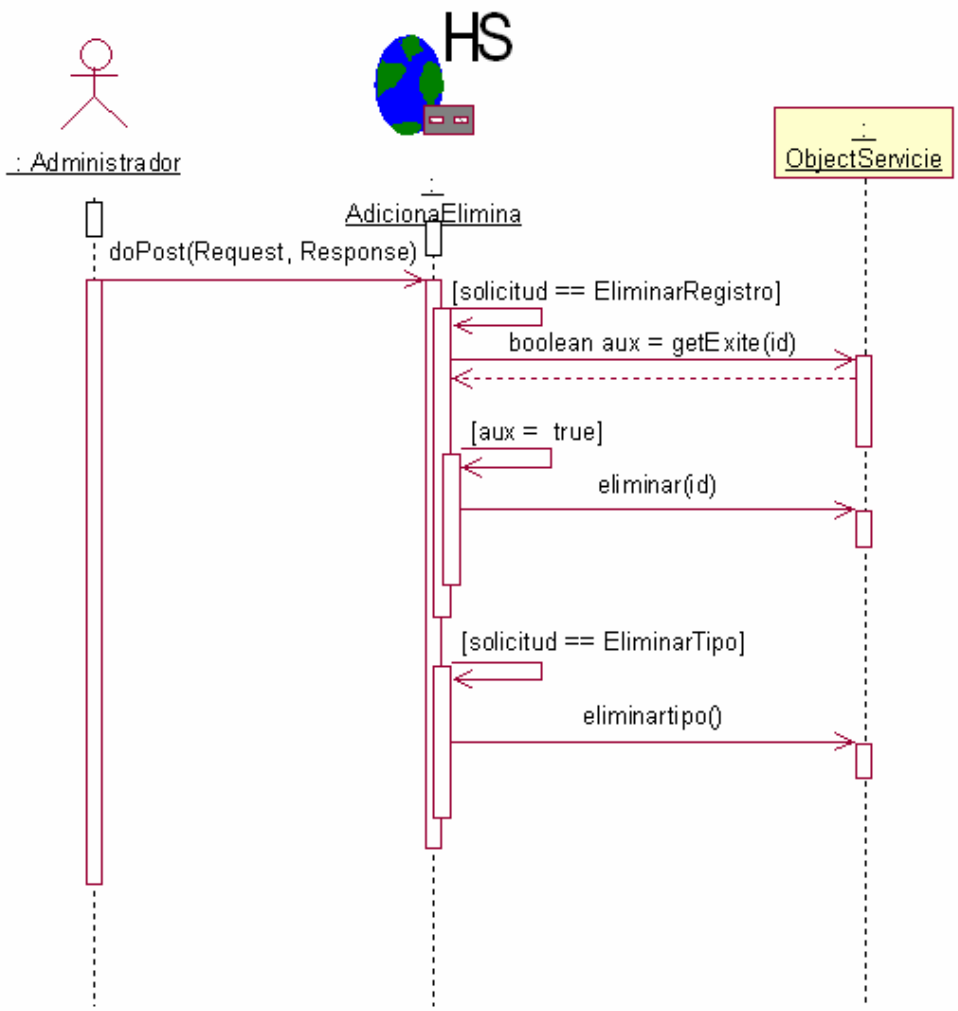

Fig. 6-7. Diagrama de secuencia. Eliminar registros.

Diagrama de secuencia de la operación Autentificarse.

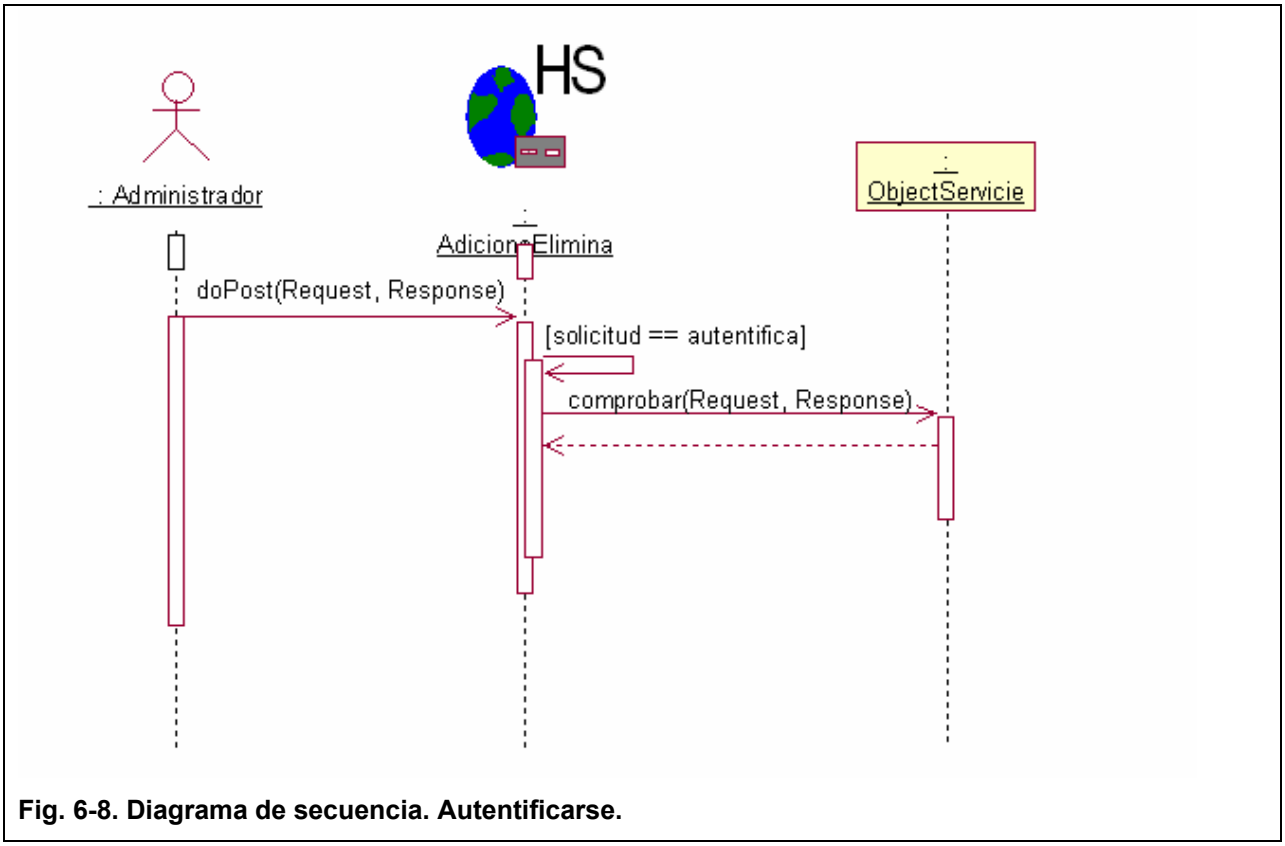

Fig. 6-8. Diagrama de secuencia. Autentificarse.

ANEXO II.

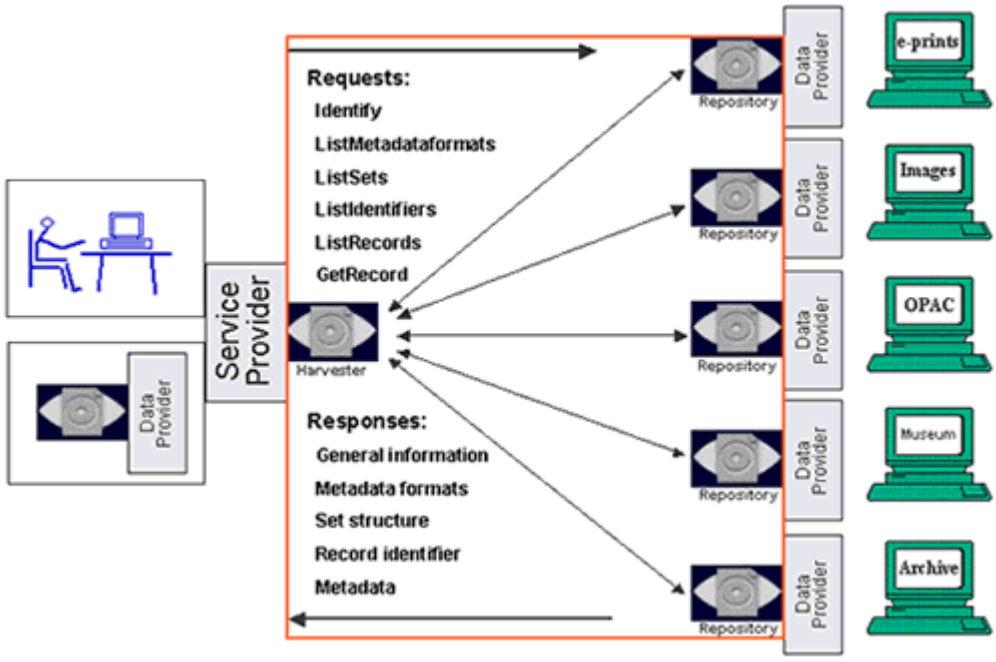

Fig. 6-1.

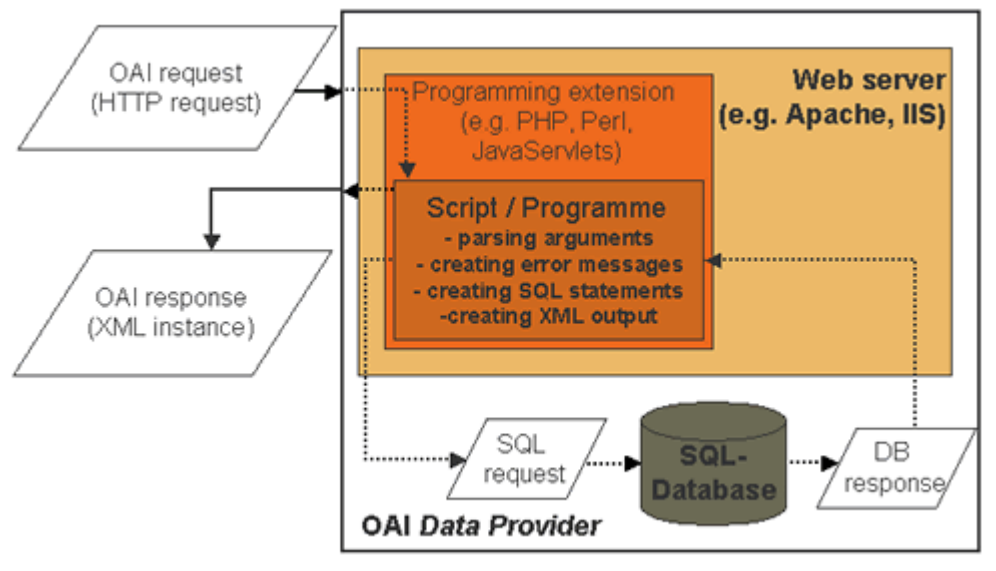

Fig. 6-2.

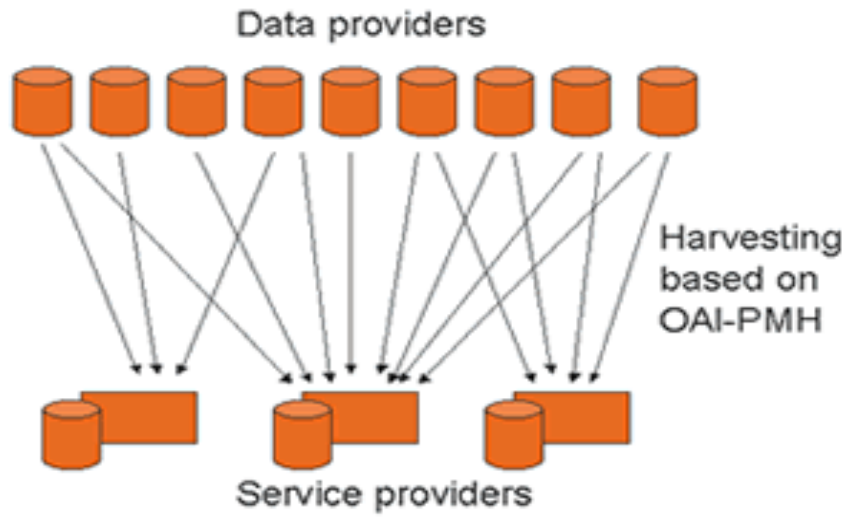
Data providers
Harvesting based on OAI-PMH
Service providers

Fig. 6-3.

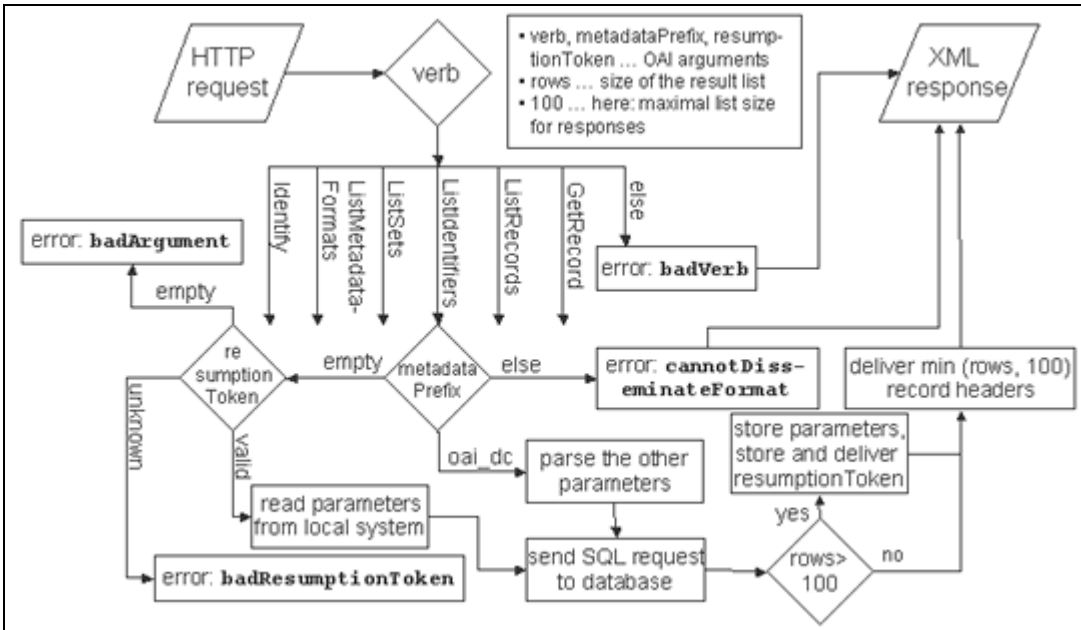
HTTP request
verb
• verb, metadataPrefix, resumptionToken ... OAI arguments
• rows ... size of the result list
• 100 ... here: maximal list size for responses
XML response
Identify
ListMetadataFormats
ListSets
ListIdentifiers
ListRecords
GetRecord
else
error: badArgument
error: badVerb
empty
resumptionToken
empty
metadataPrefix
else
error: cannotDisseminateFormat
deliver min (rows, 100) record headers
unknown
valid
oai_dc
parse the other parameters
store parameters, store and deliver resumptionToken
read parameters from local system
yes
send SQL request to database
rows> 100
no
error: badResumptionToken

Fig. 6-4.

ELEMENTOS	BREVE DESCRIPCION	CALIFICADORES APROBADOS
Title (Título)	Denominación dada al recurso digital por el creador. En caso de no existir este (por ejemplo., imagen satelital, objetos de museo) se utiliza una frase descriptiva para suplirlo.	Título Alternativo (1)
Creator (Creador)	Persona(s) primariamente responsables por el contenido intelectual del recurso.	Afiliación; E-mail; Profesión (1)
Subject (Tema)	Constituye el campo de conocimiento al que pertenece el trabajo, pudiendo ser una descripción amplia de una disciplina o una serie de descriptores específicos y de alcance diferente.	LCSH; MeSH, DDC, LCC,CDU, etc. (2)
Description (Descripción)	Descripción textual del recurso digital.	Tabla de contenidos; Resumen (1)
Publisher (Editor)	Agente o agencia responsable de hacer disponible el recurso digital.	Afiliación; E-mail (1)
Contributor (Colaboradores)	Persona(s) u organización(es), diferentes de los creadores, que han realizado un aporte intelectual importante al recurso digital. Está orientado a describir roles menos comunes, tales como ilustrador, compilador, fotógrafo, etc.	Afiliación; E-mail (1)
Date (Fecha)	Fecha en que el recurso digital fue puesto a disposición en la red	Creado; Valido; Dispo-nible; Modificado; (1) DCMI Period; W3C-DTF (2)
Type (Tipo)	Categoría o género del objeto, por ejemplo. novela, poema, diccionario, tesauro, programa ejecutable, cogido fuente, o cualquier otra categoría juzgada necesaria para el descubrimiento y/o recuperación del recurso	Imagen; Sonido; Texto; Software; etc. (1) DCT1 Type Vocabulary (2)
Format (Formato)	Representación de datos del objeto y provee información sobre software y hardware necesarios para desplegar o manipular el mismo. Esta información debería ser suficiente para facilitar un juicio de utilidad previo a la decisión de acceder al recurso digital; por ejemplo.: archivos postcript; archivos ejecutables; archivo html; etc.	----------

Identifier (Identificador)	Secuencia o número utilizado para identificar individualmente al recurso electrónico	URI (2)
Source (Fuente)	Describe objetos a partir de los cuales se deriva el recurso digital. Identifica objetos de contenido intelectual similar al descriptor, conectándolo con su versión anterior para establecer la historia del mismo.	------
Language (Lengua)	Lengua del contenido intelectual	ISO 639-2;RFC1 766(2)
Relation (Relación)	Identifica objetos de diferente contenido intelectual (digitales o no), con los cuales el recurso está lógicamente conectado. Pueden referenciar al todo (colección de documentos) o a una parte del mismo.	está referenciado por; referencia a; es parte de ; tiene parte de; es reemplazado por; reemplaza a ; etc. (1) URI (2)
Coverage (Cobertura)	Describe las características espaciales y temporales del objeto y es el elemento clave para sostener o soportar una búsqueda en un espectro espacial sobre objetos que describen datos geospaciales.	Lugar; Tiempo (1) ISO 3166; DCMI; W3C-DTF; etc. (2)
Rights (Derechos)	Copyright o enunciados sobre derechos de propiedad intelectual	----------

Fig. 6-5. Dublin Core.

Fig. 6-61. Hibernate.