**Universidad de las Ciencias Informáticas**

**Facultad 1**

**Trabajo de Diploma para optar por el título de**

**Ingeniero en Ciencias Informáticas**

# Título:

Módulo para la interconexión y búsqueda en proveedores de datos según el estándar OAI-PMH para el Sistema de Gestión de Documentos Históricos.

**Autora:** Daniurkys Otero Reyes

**Tutores:** Ing. Leodán De los Ángeles Budúen

Ing. Reynier Pernía Rodríguez

**Ciudad de la Habana**
**Junio 2012**

# Declaración de autoría

Declaro que soy la única autora de este trabajo y autorizo al Centro de Informatización Universitaria de la Universidad de las Ciencias Informáticas para que haga el uso que estime pertinente con este trabajo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

**Firma de la Autora**

Daniurkys Otero Reyes

_____

**Firma del Tutor**

Ing. Leodán De los Ángeles Buduén

_____

**Firma del Tutor**

Ing. Reynier Pernía Rodríguez

# Agradecimientos

En primer lugar quiero agradecerle a Josefina y a Bernardo, por ser abuelos y padres a la vez, por ser un ejemplo para mí, por quererme tanto y luchar junto conmigo para que lograra mis sueños.

A mi esposo por el todo cariño, el apoyo, la comprensión y el amor que me brindó durante estos cinco años de carrera.

A mi mamá por preocuparse por mí y apoyarme en todo momento.

A mis hermanos por quererme tanto, en especial a mi hermano Eduardo por cuidar de mis abuelos cuando yo no estoy.

A Mayra y a Carlos por todo el amor que me han dado, por ser como unos padres para mí.

A Alberto Ferrer por ser un gran amigo y ayudarme en todo lo que puede.

A Ana por quererme tanto y por enseñarme a luchar por mis sueños.

A Mirita, Judiht y Margarita por todo el cariño que me han dado y por haberme dado la posibilidad de tenerlas como amigas.

A Yanet, Soyma, Juan Carlos, Sergio e Isabel por todos los momentos felices que me hicieron pasar en estos dos últimos años.

A Alberto, Mayra, Jorge, Aylena, Amanda, Eddy, Dairelys y Hector por toda la ayuda que me han brindado en el itinerario de la carrera.

A Yoani por estar siempre disponible cuando necesité de su ayuda.

A Dunia por toda la ayuda que me brindo durante todo el trabajo de diploma.

A mis tutores por apoyarme y ayudarme con la realización de este trabajo.

A Villar, Lisbet, Ilda, Magdiel, Yaniris y Lisette por ayudarme en todo lo que estuvo a su alcance.

A todas aquellas personas que creyeron en mí.

Gracias.

# Dedicatoria

Esta tesis está dedicada en primer lugar a mis abuelos, Josefina y Bernardo por ser abuelos y padres a la vez, por ser mi razón de ser, por haber dedicado toda su vida a mi formación como persona, por brindarme apoyo en todo momento y por haber tenido siempre plena confianza en mí.

A mi esposo por ser una de las mejores cosas que me ha pasado en la vida y por todo el amor, entrega y dedicación que me ha dado.

# Resumen

Actualmente el Sistema de Gestión de Documentos Históricos ArchiVenHIS no brinda la posibilidad de interactuar con otros sistemas de archivos, por esta razón el presente trabajo de diploma está dirigido a desarrollar un módulo para ArchiVenHIS que le permita recuperar información de otros sistemas. Para el desarrollo del mismo se realizó un estudio sobre los protocolos de intercambio y recuperación de la información donde se seleccionó el estándar OAI-PMH para darle respuesta al problema que dio origen a esta investigación.

Para lograr una mayor comprensión del sistema, se realiza el modelo de dominio donde se representan las clases más importantes dentro de este contexto. Se realizó la especificación de los requerimientos funcionales y no funcionales del sistema. Se modelaron los artefactos relacionados con el flujo de trabajo de Análisis y Diseño y se definió el modelo de datos para el desarrollo de la base de datos. Finalizando con la elaboración y ejecución de los casos de prueba realizados al sistema.

**Palabras clave:** estándar, OAI-PMH, protocolos, Sistema de Gestión de Documentos Históricos

# Índice

# Introducción

La historia de la evolución humana ha podido construirse gracias a las salvaguardas de la información en disímiles soportes a través del tiempo. Los espacios físicos destinados al almacenamiento de los documentos a lo largo de la historia han tomado diferentes nombres en dependencia de las culturas y de las funciones para los cuales fueron creados. En la actualidad los lugares destinados a guardar estos documentos se conocen con el nombre de archivos.

Los archivos son considerados una institución o una parte estructural de ella, que realiza la recepción, organización y conservación de los documentos para su utilización; conjunto orgánico de documentos producidos y/o acumulados por una persona natural o jurídica (1) para conservar y esparcir el conocimiento que se acumula generación tras generación. Sus funciones han sido tradicionalmente: seleccionar, describir, conservar y servir los documentos (2).

Los archivos juegan un papel esencial para la gestión documental, pues garantizan el tratamiento de los documentos a lo largo de toda su vida. La gestión documental es un área de la administración general que se encarga de garantizar la economía y la eficiencia en la creación, mantenimiento, uso y disposición de los documentos administrativos durante todo su ciclo de vida (2).

Una adecuada gestión documental puede reportar a las organizaciones ventajas en términos de ahorro de costos y agilización de la tramitación de contratos y documentos. La gestión documental según la archivística integrada se divide en cinco etapas: la primera se conoce como Génesis, la segunda Archivo de Gestión, la tercera Archivo Central, la cuarta Archivo Intermedio y la quinta Archivo Histórico. Esta última tiene como misión atesorar, organizar, custodiar y conservar la documentación de valor permanente que constituye el patrimonio documental de una nación o territorio.

Producto del incremento del volumen de documentación y el desarrollo de las tecnologías de la información y las comunicaciones (TIC), surgieron los Sistemas de Gestión de Documentos Históricos, una solución que contribuye a la conservación y difusión de la memoria histórica resguardada por instituciones de archivo.

Cuba no se encuentra aislada de estos avances tecnológicos, es por esto que en el año 2002 se creó la Universidad de las Ciencias Informáticas (UCI) que entre sus objetivos incluye formar a un profesional altamente calificado, informatizar el país y desarrollar la industria del *software* para contribuir al desarrollo económico del mismo. Para cumplir con estos objetivos, la misma cuenta con diferentes centros de producción dentro de los cuales se encuentra el Centro de Informatización Universitaria (CENIA), y en este un departamento de gestión documental con varias líneas de desarrollo encargadas del desarrollo de software para facilitar el tratamiento que se le da a los archivos

a lo largo de su vida.

Durante el año 2007 se desarrolla el Sistema de Gestión de Documentos Históricos ArchiVenHIS, en el marco del proyecto “Uso y Aplicación de las TIC’s para el mejoramiento de la Gobernabilidad y Aumento de la Soberanía Tecnológica (Fase 1)”, en el que se proporciona al Archivo General de la Nación (AGN) de la República Bolivariana de Venezuela una solución, desarrollada bajo tecnologías libres y estándares abiertos, que contribuye a la preservación y difusión del fondo documental bajo su custodia.

ArchiVenHIS cuenta con variadas funcionalidades (búsqueda sencilla, búsqueda avanzada, exploración) que facilitan al usuario la localización de los documentos con base en los metadatos descritos a través del sistema siguiendo la Norma Internacional General de Descripción Archivística ISAD (G). Sin embargo, el sistema no es capaz de brindar información sobre el material custodiado por otras instituciones de archivo aun cuando estas utilicen un sistema informático con mecanismos que garanticen la consulta de esta información.

La capacidad de proveer semejante servicio constituye un importante valor agregado para los Sistemas de Gestión de Documentos Históricos. En la actualidad muchos de ellos brindan soporte para estas funcionalidades, contribuyendo a una mejor difusión de la memoria histórica custodiada por las instituciones donde se utilizan.

Por lo anteriormente expuesto se plantea como **problema a resolver:** ¿cómo recuperar metadatos archivísticos desde otros Sistemas de Gestión de Documentos Históricos a través de ArchiVenHIS?

Después del problema anterior planteado se define como **objeto de estudio** los protocolos que se utilizan para la comunicación entre aplicaciones web y como **campo de acción,** aquellos protocolos que se emplean para la comunicación entre aplicaciones que gestionan documentos históricos.

De acuerdo con la problemática planteada se propone como **objetivo general** desarrollar un módulo para el Sistema de Gestión de Documentos Históricos ArchiVenHIS que le permita recuperar información de otros sistemas.

El presente trabajo queda sustentado por la **idea a defender:** el desarrollo de un módulo para realizar intercambios de metadatos, le permitirá al Sistema de Gestión de Documentos Históricos ArchiVenHIS una mayor difusión de información.

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

- Evaluar las tendencias actuales de los Sistemas de Gestión de Documentos Históricos.
- Estudiar los protocolos que permitan la comunicación entre diferentes sistemas.
- Identificar los requisitos funcionales y no funcionales para el módulo a desarrollar.

- Diseñar las interfaces de usuario.
- Diseñar la base de datos.
- Implementar los elementos de diseño.
- Implementar las pruebas del desarrollador.
- Ejecutar las pruebas del desarrollador.
- Integrar el sistema.

Para dar cumplimiento a las tareas planteadas anteriormente se utilizaron diferentes métodos de la investigación como:

- Métodos Teóricos:
  - **Analítico – sintético:** en el presente trabajo de diploma se utilizó este método en el proceso de análisis de la bibliografía utilizada, realizando la extracción de los elementos más importantes relacionados con los metadatos y con los protocolos de intercambio de información entre sistemas, para así facilitar el resultado de la investigación.
  - **Análisis histórico – lógico**: en el presente trabajo de diploma se utilizó este método para analizar la trayectoria de los protocolos de intercambio de datos entre sistemas de información, su condicionamiento a los diferentes periodos de la historia y las conexiones históricas fundamentales.
- Método Empírico:
  - **Observación:** en el presente trabajo de diploma se utilizó este método para realizar un análisis de los principales Sistemas de Gestión de Documentos Históricos que pueden intercambiar información entre ellos.

El presente trabajo consta de una introducción, **tres capítulos**, conclusiones, recomendaciones, referencias bibliográficas y bibliografía, glosario de términos y anexos, el cual constituye una constancia del modo en el que se da cumplimiento a los objetivos planteados. Los capítulos están estructurados de la siguiente manera:

**Capítulo 1. La gestión archivística. Definición y estándares de comunicación entre sistemas:** Se abordarán los conceptos fundamentales a tener en cuenta para comprender los términos usados en el desarrollo del trabajo. Se realiza un estudio de los principales protocolos para lograr la interoperabilidad entre Sistemas de Gestión de Documentos Históricos. Además, se exponen las distintas tecnologías a utilizar en el desarrollo del módulo propuesto.

**Capítulo 2. Características del módulo para la interconexión y búsqueda en proveedores de datos:** Se realiza la descripción de las actividades metodológicas orientadas por RUP (por su siglas en inglés, *Rational Unified Process*) para la construcción de la solución propuesta. Se desarrolla el modelo

de dominio, capturando los conceptos más importantes en el contexto del sistema. Se realiza un levantamiento de los requerimientos funcionales a partir de los cuales se definen los casos de uso del sistema. Como parte de los procesos de análisis y diseño se confecciona un conjunto de artefactos dentro de los cuales se destacan los diagramas de clases del análisis y los diagramas de clases del diseño. Se define el patrón de arquitectura que será aplicado. Además, se presenta el modelo de datos.

**Capítulo 3. Implementación y prueba del módulo para la interconexión y búsqueda en proveedores de datos:** Se presentan los diagramas de despliegue y componentes como elementos primordiales de la fase de implementación. Además, se describen un conjunto de pruebas realizadas al *software* para medir la funcionalidad del mismo y como es capaz de responder a los requerimientos para los cuales fue concebido.

# Capítulo 1: La gestión archivística. Definición y estándares de comunicación entre sistemas

En este capítulo se presentan los conceptos fundamentales para comprender los términos usados en la presente investigación. Se realiza un estudio de los principales protocolos para lograr la interoperabilidad entre Sistemas de Gestión de Documentos Históricos, seleccionando uno de ellos para darle solución al problema planteado. Además, se realiza una breve descripción de las distintas herramientas y tecnologías que serán utilizadas en el desarrollo del módulo.

## 1.1 Gestión archivística. Conceptos fundamentales y características

Los **archivos** generalmente son conocidos como los edificios que contienen los documentos o el mueble que los guarda. Sin embargo, este término presenta tres significados: como contenido documental, como institución y como lugar de conservación.

El diccionario de terminología archivística, lo define con tres acepciones (3):

- Conjunto orgánico de documentos producidos y/o recibidos en el ejercicio de sus funciones por las personas físicas o jurídicas, públicas y privadas.
- La institución cultural donde se reúnen, conservan, ordenan y difunden los conjuntos orgánicos de documentos para la gestión administrativa, la información, la investigación y la cultura.
- El archivo también es el local donde se conservan y consultan los conjuntos orgánicos de documentos.

El **documento de archivo** se define como el testimonio material de un hecho o acto realizado en el ejercicio de sus funciones por personas físicas o jurídicas, públicas o privadas, de acuerdo con unas características de tipo material y formal (2).

La **gestión documental** es un área de la administración general que se encarga de garantizar la economía y la eficiencia en la creación, mantenimiento, uso, y disposición de los documentos administrativos durante todo su ciclo de vida. Esta se extiende al ciclo de vida complejo del documento, desde su producción hasta la eliminación final y su envío al archivo para su conservación permanente. Está dirigida a asegurar una documentación adecuada, evitar lo no esencial, mejorar la forma de cómo se organizan y recuperan los documentos (2).

Por otra parte la **descripción archivística** consiste en el análisis realizado por el archivero sobre los fondos y los documentos de archivo agrupados natural o artificialmente, a fin de sintetizar y condensar

la información en ellos contenida para ofrecerla a los interesados (4). La descripción persigue dos objetivos: dar información a los demás y facilitar el control al archivero (5).

Las descripciones archivísticas son realizadas a través de los instrumentos de descripción, que no son más que las representaciones de los documentos o de sus agrupaciones, por cuanto los transforman mediante una forma distinta de la original (4). Atendiendo al nivel de detalle, los instrumentos tradicionales de descripción son: guías, inventarios, catálogos e índices.

En los últimos años un instrumento descriptivo que está ganando mucha aceptación es La Norma Internacional General de Descripción Archivística ISAD (G), la cual es una normativa internacional para la descripción de documentos. Esta norma está concebida una guía general para la elaboración de descripciones archivísticas.

ISAD (G) contiene reglas generales para la descripción archivística que pueden aplicarse con independencia del tipo documental o del soporte físico de los documentos de archivos. Los objetivos del cumplimiento de estas reglas son:

- Garantizar la elaboración de descripciones coherentes, pertinentes y explícitas.
- Facilitar la recuperación y el intercambio de información sobre los documentos de archivos.
- Compartir los datos de autoridad.
- Hacer posible la integración de las descripciones procedentes de distintos lugares en un sistema unificado de información.

Esta norma está constituida por 26 elementos que permiten la descripción archivística, de los cuales sólo 6 se consideran de carácter obligatorio para el intercambio internacional de la información descriptiva, estos son (6):

- Código de referencia
- Titulo
- Productor(es)
- Fecha(s)
- Extensión de la unidad de descripción
- Nivel de descripción

Las reglas ISAD (G) se estructuran en 7 áreas de información descriptiva:

- Área de Identificación: contiene la información esencial para identificar la unidad de descripción.
- Área de Contexto: contiene la información relativa al objeto y organización de la unidad de descripción.
- Área de acceso y utilización: contiene la información relativa a la accesibilidad de la unidad de descripción.

- Área de documentación asociada: contiene la información relativa a aquellos documentos que tienen una relación significativa con la unidad de descripción.
- Área de notas: contiene información especial y aquella otra que no ha podido incluirse en ninguna de las demás áreas.
- Área de control de la descripción: contiene la información relativa al cómo, cuándo y quién ha elaborado la descripción archivística.

## 1.2 Protocolos para la comunicación entre sistemas. Definición y características.

### 1.2.1 Protocolo

Un protocolo es un conjunto de normas que definen la comunicación entre sistemas. FTP (Protocolo de Transferencia de Ficheros) y HTTP (Protocolo de Transferencia de Hipertexto) son ejemplos de otros protocolos utilizados para la comunicación entre sistemas a través de Internet (7).

### 1.2.2 Interoperabilidad

La interoperabilidad es “la capacidad de un sistema de información de comunicarse y compartir datos, información, documentos y objetos digitales de forma efectiva (con una mínima o nula pérdida de su valor y funcionalidad), con uno o varios sistemas de información (siendo generalmente estos sistemas completamente heterogéneos, distribuidos y geográficamente distantes), mediante una interconexión libre, automática y transparente, sin dejar de utilizar en ningún momento la interfaz del sistema propio” (8).

### 1.2.3 Metadatos

Debido a la enorme cantidad y variedad de las fuentes y recursos de información disponibles en Internet, surgió el problema de clasificarla e identificarla de manera rápida y eficiente por lo que se hizo necesario crear un mecanismo que fuese capaz de etiquetar, catalogar, describir y clasificar los recursos presentes en la Web con el fin de facilitar la búsqueda y recuperación de la información. Este mecanismo es a lo que hoy se le conoce con el nombre de metadatos.

Los metadatos son la información estructurada acerca de recursos (digitales y no digitales) (9). Uno de los objetivos que presentan los metadatos es ayudar a darle soporte y publicitad a los datos que una persona u organización han producido.

Existen diversas clasificaciones de los metadatos atendiendo a distintos aspectos como su forma,

funcionalidad, nivel de estructuración de los datos, persona o entidad que los origina, etc. La calificación que se ofrece a continuación es según la función que proporcionan (10):

- **Metadatos descriptivos:** son aquellos que sirven para la descripción e identificación de los recursos de información, permiten la búsqueda y recuperación de la información, como también distinguir un recurso de otro y entender el asunto o contenido del mismo.
- **Metadatos estructurales:** son los que más influyen en la recuperación de la información electrónica, facilitan la navegación y presentación de los recursos electrónicos. Ofrecen la información sobre la estructura interna de los recursos, estableciendo las relaciones entre ellos, de manera que pueden incluso unir los archivos de imagen y textos que están relacionados.
- **Metadatos administrativos:** son de carácter más técnico porque incluyen datos sobre la creación y control de calidad, datos sobre la gestión de derechos, requisitos del control de acceso y utilización, información sobre la preservación y permiten la gestión a largo y corto plazo.

### 1.2.4 Protocolos de intercambio y recuperación de la información

Cualquier desarrollo informático que involucre un procesamiento de datos e información, y que apoye a las organizaciones en la optimización, mejoramiento y desarrollo de sus procesos puede ser considerado un sistema de información (8). Los Sistemas de Gestión de Documentos Históricos corresponden a un tipo especial de sistema de información. A partir de la evolución de las TIC, surge la necesidad de establecer la interoperabilidad entre los sistemas de información, y por ende entre los Sistemas de Gestión de Documentos Históricos, con el objetivo de posibilitar la transferencia e intercambio de información entre ellos. Para que esta comunicación pueda establecerse de una manera organizada surgieron algunos protocolos dentro de los cuales se pueden encontrar:

- **Protocolo Dienst**

Permite la comunicación entre servicios que conforman una biblioteca digital. Fue diseñado y probado con el sistema NCSTRL (*Networked Computer Science Technical Report Library*) (11). Parte de cuatro tareas básicas (búsqueda y recuperación de documentos, navegación, agregación de nuevos documentos y registro de usuarios) y propone seis categorías de servicios: repositorio, indexado, mediadores, información, colección y registro. En la actualidad este protocolo es muy poco usado por las instituciones.

- **Protocolo Guildford**

Proporciona un conjunto de reglas para la publicación y el intercambio de documentos en el Internet, este puede ser implementado tanto individualmente como en grupos y es especialmente

adecuado si los metadatos se codifican en REDIF (por su siglas en inglés, *Research Document Information Format*) (12). Su surgimiento se remonta a una declaración realizada por William L. Goffe el 15 de julio de 1995, donde expresó la necesidad que existía de un sistema distribuido con cualquier número de sitios, donde cada uno fuera el reflejo del otro. Este protocolo fue creado con el objetivo de facilitar el acceso a los últimos resultados de investigaciones en las diferentes áreas de la economía, lo cual no quiere decir que no pueda ser aplicable a otros sectores de la sociedad.

- **Protocolo Simple para la Interoperabilidad de Bibliotecas Digitales (SDLIP)**

SDLIP es un protocolo para la integración de múltiples fuentes, de información heterogénea. Ha sido desarrollado conjuntamente por la Universidad de Stanford, UC Berkeley, UC Santa Bárbara, el San Diego Supercomputer Center, y el Proyecto de Biblioteca Digital de California. Los clientes utilizan SDLIP para solicitar realizar búsquedas sobre las fuentes de información (13).

- **Protocolo Z39.50**

Z39.50 es un protocolo para la recuperación de información basado en la estructura cliente/servidor que facilita la interconexión de sistemas informáticos. Es un protocolo desarrollado y mantenido por bibliotecarios y se adopta como la norma ISO 23950. Su principal objetivo consiste en permitirle al usuario realizar búsquedas en bases de datos que cuenten con un servidor Z39.50, sin tener que conocer las sintaxis de búsqueda que utilizan dichos sistemas. La funcionalidad que realiza este protocolo es muy compleja, por ejemplo, trata cuestiones como el manejo de sesiones, gestión de conjuntos de resultados y permite la especificación de predicados para filtrar los resultados obtenidos, proporcionando así un incremento en la complejidad de la implementación y por ende un aumento de los costos. Este protocolo ha sido utilizado mayoritariamente por bibliotecarios y a partir del empleo de estándares como el formato MARC (14).

- **Iniciativa de Archivos Abiertos –Protocolo para la Recolección de Metadatos (OAI-PMH)**

El marco diseñado por OAI (por sus siglas en inglés, *Open Archives Initiative* o Iniciativa de Archivos Abiertos) es bastante simple y posee como objetivo una mínima complicación para las instituciones que en algún momento deseen implementarlo. Si bien OAI–PMH surgió en el seno de la comunidad académica y científica para la búsqueda y recuperación de textos electrónicos, es perfectamente aplicable en cualquier contexto documental.

Sus bases se encuentran sentadas en el encuentro realizado en Santa Fe, Estados Unidos. Este encuentro se realizó en octubre de 1999, con la idea de que la interoperabilidad de los archivos de las comunidades de publicaciones electrónicas (eprints) era fundamental para aumentar su impacto entre la comunidad académica (15). Con ella se podrían unir varios archivos, intercambiar registros o realizar búsquedas en disciplinas relacionadas al mismo tiempo. Las personas que asistieron a esta reunión

fueron especialistas en bibliotecas digitales, así como representantes de los principales archivos existentes en ese momento.

Después de largas horas de debates, los resultados alcanzados en la reunión fueron un conjunto de acuerdos técnicos y organizativos conocidos como la Convención de Santa Fe. Los aspectos técnicos que allí se definieron incluían tres puntos fundamentales: un formato para los metadatos, un protocolo y un sistema de identificación; pero no fue hasta enero de 2001 que se concretó lo planteado, con la publicación del protocolo para la comunicación de metadatos denominado OAI-PMH versión 1.0. Desde ese momento, muchas empresas comenzaron la implementación del protocolo por lo que no demoraron mucho tiempo en aparecer las primeras instituciones que lo utilizaron para poner en internet sus metadatos. La adopción del protocolo fue lenta y progresiva, elaborándose en 2002 la versión 2 del mismo.

Después de haber realizado un estudio de algunos de los protocolos para lograr interoperabilidad entre sistemas, con especial énfasis en Z39.50, se decide emplear OAI-PMH para el intercambio de la información almacenada en ArchiVenHIS y a su vez recuperar la de otros Sistemas de Gestión de Documentos Históricos. La selección de este protocolo estuvo basada en los siguientes aspectos:

- Simplicidad a la hora de implementar.
- Cantidad de instituciones de archivo que lo han desarrollado.
- Presenta mayor interoperabilidad que otros protocolos pues a través de uno de los verbos del protocolo es posible obtener los formatos que implementa el proveedor de datos, así como realizarle peticiones en cualquiera de ellos.
- Está basado en estándares ampliamente utiilzados, como son el HTTP para la transmisión de datos y órdenes y XML (Lenguaje de Marcas Extensible) para la codificación de metadatos.

### 1.2.5 Flujo actual de los procesos de OAI-PMH

OAI-PMH no es un protocolo de búsqueda, pero su utilización puede servir de apoyo para los servicios de búsqueda, es una capa sobre la que construir otros servicios (16). Este protocolo utiliza transacciones HTTP para emitir preguntas y obtener respuestas entre un proveedor de servicios (PS) y un proveedor de datos (PD). El primero puede pedir al segundo que le envíe metadatos según determinados criterios. En respuesta, el segundo devuelve un conjunto de registros en formato XML, incluyendo identificadores de los objetos descritos en cada registro.

Las peticiones se realizan 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 tipos de peticiones que

un proveedor de servicio puede realizar a un proveedor de datos, las cuales se conocen como verbos (17):

- **Identify**

Esta petición se utiliza para que el PS tenga conocimiento de la información tanto técnica como administrativa del PD con el objetivo de establecer cómo se va a realizar el proceso de recolección de metadatos. La información remitida por el PD al PS debe incluir una instancia de los siguientes elementos:

- Nombre del repositorio.
- Base url.
- Versión del protocolo OAI-PMH por el cual se va a realizar el cosechado.
- earliestDatestamp: expresa la fecha del último cambio de adición, eliminación o modificación de los objetos dentro del repositorio.
- deletedRecord: establece la forma como se efectúa el proceso de eliminación de los objetos dentro del repositorio, que puede ser de tres tipos: no se realiza, transitoria o persiste.
- Granularidad: define el formato de espacio y tiempo bajo el cual está soportado el repositorio, tiene que estar definida en la norma ISO 8601 31.
- Correo electrónico del administrador del repositorio.

- **ListMetadataFormats**

Esta petición se realiza para conocer bajo qué formatos de metadatos el PS puede recuperar la información dada por el PD, por lo general este protocolo se basa en los metadatos Dublin Core que será descrito más adelante. Con los resultados de esta petición, el PS elegirá la sintaxis de metadatos que utilizará para recuperar la información del PD y se convertirá en un argumento metadataPrefix, con el cual el PD remitirá los objetos resultantes para la posterior cosecha.

- **ListSets**

Esta petición se utiliza para recuperar un conjunto de registros. Estos conjuntos son creados de forma opcional por el servidor para proveer una recuperación selectiva de los registros. Estos conjuntos pueden ser simples listas o estructuras jerárquicas.

- **ListIdentifiers**

Esta petición se utiliza para obtener la información del verbo ListRecords de una forma más resumida, pero con la diferencia de que aquí solamente se recuperarán los encabezamientos de los registros, en lugar de los registros completos, por esta razón es posible afirmar que los argumentos requeridos por este verbo son iguales a los que se solicitan cuando se hace una petición del verbo ListRecords.

Con estas cabeceras, el PS obtiene más información sobre el identificador OAI del objeto a recolectar, el datestamp o fecha de ingreso o modificación del objeto dentro del sistema y los setSpec o identificadores de los set relacionados, que son por los cuales se agrupan los objetos del repositorio

- **ListRecords**

Esta petición es la que se utiliza para recolectar los objetos que están dentro de un repositorio, con el fin de darle un resultado al PS relacionado con las necesidades del mismo; este verbo puede tener como argumentos las siguientes variables:

- Fechas (superior e inferior) dentro de las cuales se pueden agrupar los metadatos de los objetos a cosechar.
- Set: es la agrupación por alguna característica descriptiva de los contenidos dentro del repositorio; si no se envía el o los set dentro de los argumentos del verbo ListRecords, entonces el resultado de esta petición serán todos los objetos que se encuentran dentro del repositorio, sin excepción alguna.
- resumptionToken: es un argumento exclusivo que se maneja dentro del control de intercambio de información de este protocolo. Es importante destacar que se utiliza cuando la lista de metadatos de objetos del repositorio es muy grande, cuando esto sucede se realiza un envío por lotes comunicándole al PS cuántos son, cuántos se han transmitido y si desea cosechar todos los metadatos asociados a esta petición.

- **GetRecord**

Esta petición se utiliza para recuperar un registro concreto, para lograr esto, se le debe realizar la petición al PD con los siguientes argumentos:

- Identifier: es la llave o número único dentro del repositorio que identifica al objeto dentro del sistema.
- MetadataPrefix: es la sintaxis de metadatos que se va a utilizar para realizar el cosechado del objeto.

OAIPMH, soporta múltiples formatos para expresar los metadatos, no obstante, requiere que todos los servidores ofrezcan los registros utilizando Dublin Core codificado en XML (15). Además, cabe destacar que cada servidor puede ofrecer los registros en otros formatos adicionales. Un cliente puede solicitar que los registros se le brinden en cualquier otro formato que sea soportado por el servidor.

Uno de los aspectos que no trata el protocolo son por ejemplo las cuestiones de gestión o autorización para el acceso de los clientes. El servidor deberá acudir a métodos externos si desea restringir los clientes a los que sirva información, en relación con este punto está la utilización que hagan los clientes de los datos. Finalmente, nunca trata el tema de cómo los clientes pueden localizar

aquellos servidores que contengan los datos que necesitan.

### 1.2.6 Proveedores de datos y de servicio

De la sección anterior se desprende que la arquitectura que presenta el protocolo OAI-PMH se basa en el modelo cliente/servidor que transmiten preguntas y respuestas entre un PD y un PS, los primeros son los archivos que suministran la información y los segundos son los recolectores que toman los datos, con el objetivo de incorporarles algún valor añadido y presentarlos a los usuarios finales.

Cuando un proveedor de datos recibe una petición de metadatos, este primeramente la procesa a través de una capa de aplicación programada, la cual contiene una implementación de OAI-PMH para realizar una consulta a su sistema de gestión de base de datos que contiene los metadatos relativos a los recursos que se encuentran en el repositorio. Una vez hecha la consulta la aplicación debe haber recopilado los metadatos, compone la respuesta a la petición realizada en formato OAI-PMH con sintaxis XML y posteriormente se envía a través de HTTP (18).

Luego la capa de aplicación del proveedor de servicio recolecta la respuesta y se encarga de introducir en su propia base de datos los metadatos recibidos para componer a continuación los servicios de valor añadido que presentará al usuario final.

En los anexos (Ilustración 1) se describe la estructura y el flujo de trabajo del sistema de forma detallada, especificando el tipo de peticiones HTTP.

Los Proveedores de Datos manejan el depósito y la publicación de los recursos en un repositorio y exponen los metadatos de los recursos del repositorio para que puedan ser recolectados. Ellos son los creadores y conservadores de los metadatos y de los repositorios de recursos (16). Estos deben cumplir con varios requisitos tales como (18):

- Soporte de almacenamiento de metadatos: Es recomendable que los metadatos se almacenen en una base de datos relacional a la que se pueda acceder mediante consultas SQL, es necesario que exista un único identificador para cada recurso. También se puede utilizar un sistema de ficheros tipo LDAP (Protocolo ligero de acceso a directorios).
- Soporte web: Un servidor web.
- 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.
- URL: Mantenimiento de un identificador para el repositorio o revista único, basado en su URL.
- URI: Mantenimiento de un identificador único para cada recurso que forma parte del repositorio.

- Formato de los metadatos: Se pueden utilizar todos los formatos de metadatos que se desee. Como mínimo debe soportar Dublín Core sin cualificar.
- *Datestamps*: Se debe mantener una marca de tiempo para la creación del recurso y otra para su modificación si fuera necesario.
- Soporte lógico para una jerarquía de conjuntos (sets): Permite el mantenimiento de clasificaciones de cara a una ordenación temática y/o conceptual de los recursos.
- Control de flujo: Se consigue a través de una implementación de la reanudación de la señal en la que se basa el diálogo entre el proveedor de metadatos y el servidor de datos. Este requisito no es obligatorio pero sí muy recomendable en cualquier proveedor de metadatos.

Los componentes que debe presentar un PD son (18):

- 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 un PS.
- Control de flujo que realice una lista de secuencias incompleta cuando los repositorios sean muy grandes de cara a que el envío de la respuesta se haga poco a poco en función de la reanudación de la señal.

En los anexos (Ilustración 2) se muestra el diagrama de interacción de los componentes del proveedor de datos.

Los PS recolectan los metadatos de los PD. Ellos emplean los metadatos recolectados con el fin de proporcionar servicios (16). Al igual que el PD el PS también presenta sus requisitos los cuales se enuncian a continuación (18):

- 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 (Internet Information Server) u 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 PS son (18):

- Gestión del archivo: Sistema de administración web que permita dar de alta a nuevas revistas y repositorios (PD) 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 proveedores de datos 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 partes 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 (normalizer): 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: 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.
- Módulo 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 proveedores de datos.

En los anexos (Ilustración 3) se muestra la interacción de los componentes del proveedor de servicio.

### 1.2.7 Dublin Core

Dublin Core es un modelo de metadatos elaborado por la DCMI (*Dublin Core Metadata Initiative*), una organización dedicada a fomentar la adopción extensa de los estándares interoperables de los metadatos y a promover el desarrollo de los vocabularios especializados de metadatos para describir

recursos. Las implementaciones de Dublin Core usan generalmente XML y se basan en el Resource Description *Framework* (19).

Dublin Core tiene como objetivo otorgarle un significado semántico a las quince definiciones descriptivas que presenta. Cada definición es opcional y puede repetirse. Además, estas pueden aparecer en cualquier orden. Este sistema de definiciones fue diseñado especialmente para proporcionar un vocabulario de características base, que fuese capaz de facilitar la información descriptiva básica referente a cualquier recurso, sin que importe cual es el formato de origen, el área de especialización o el origen cultural.

Los elementos de Dublin Core pueden clasificarse en tres categorías según el tipo de información que contengan (20):

- Sobre el contenido del recurso

| Etiquetas DC | Descripción |
|---|---|
| DC.Title | Título. El nombre dado al recurso. |
| DC.Subject | Materias y palabras clave. El tema del contenido del recurso. |
| DC.Description | Descripción del contenido del recurso. Puede incluir un resumen, una tabla de contenidos, etc. |
| DC.Source | Fuente. Referencia al recurso del que deriva el documento actual. |
| DC.Languaje | Lengua. El idioma del contenido del recurso. |
| DC.Relation | Relación. Una referencia a un recurso relacionado con el contenido. |
| DC.Coverage | Cobertura. Ámbito del contenido del recurso. Puede tratarse de una especificación geográfica, temporal o legal. |

- Sobre la propiedad intelectual del recurso

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

- Sobre la instancia del recurso

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

### 1.2.8 Análisis de soluciones existentes

El auge tecnológico ha traído consigo un mayor avance en los sistemas de gestión de archivos históricos, existiendo en la actualidad diferentes *software* que le permiten a estos difundir sus fondos en internet. Esto se puede lograr implementando un protocolo para el intercambio y recuperación de metadatos, siendo el más adecuado para la gestión archivística OAI-PMH. A continuación se muestran algunas aplicaciones existentes que implementan este protocolo (Archivo 3000, Pares) con el objetivo de determinar si su código fuente puede ser reutilizado para la implementación del módulo a desarrollar.

**Archivo 3000**

Archivo 3000 es un *software* para la gestión de archivos que utiliza normas ISAD (G) y un formato MARC para la introducción de datos y está capacitado para cubrir las necesidades de la mayoría de estas instituciones. Fue desarrollado por 3000 Informática S.L, en el lenguaje de programación Java. Este sistema utiliza la norma ISO-15489 para la gestión de registros y la Especificación MoReq para registros electrónicos. Es multiplataforma y funciona con Oracle y PostgreSQL. Dentro de las características generales que posee Archivo 3000 se pueden encontrar las siguientes (21):

- Utiliza la norma ISAD (G) para los niveles previstos: fondo, subfondo, serie, unidad documental compuesta y unidad documental simple, además Archivo 3000 incluye depósito, grupo o sección y subserie o parte de serie.
- Posibilidad de integrar imágenes y sonidos en los documentos descritos.

- Acceso a los distintos niveles de descripción a partir de múltiples puntos: nombres, fechas, títulos, materias, términos geográficos, fecha de entrada, fecha de producción, tipo diplomático, búsquedas truncadas, dígitos de clasificación, etc.
- Importación y exportación de datos.
- Permite realizar préstamos a investigadores tanto para uso interno como para uso externo.
- Posibilidad de búsqueda en línea por los usuarios y la personalización de las consultas de ellos.

Cabe destacar que este *software* a nivel mundial es considerado un líder en el mundo de la archivística puesto que ofrece disimiles productos y servicios. Cada uno de estos posee un valor monetario lo cual lo hace inalcanzable para algunos usuarios, además de que no revela su código fuente.

**PARES**

El Portal de Archivos Españoles (PARES), es un proyecto del Ministerio de Cultura Español destinado a la difusión en Internet del Patrimonio Histórico Documental. Fue desarrollado en Java y emplea tecnologías XML. Ofrece un acceso libre y gratuito para cualquier usuario que se encuentre interesado en acceder a los documentos con imágenes digitalizadas de los Archivos Españoles. Funciona sobre una base de datos Oracle. Este portal les brinda la posibilidad a los usuarios de realizar búsquedas simples y avanzadas de los archivos históricos a través de una interfaz amigable y sencilla, además permite la interconexión con otros archivos (22). Actualmente el contenido de PARES no esta disponbile a través del protocolo OAI, pero el equipo de desarrollo tiene previsto que lo estén a mediano plazo.

A pesar de ser un sitio bien diseñado resulta imposible reutilizar su código fuente puesto que el mismo es cerrado.

## 1.3 Tecnologías para el desarrollo

La selección de las herramientas, lenguajes y tecnologías utilizadas para la implementación del módulo de interconexión y búsqueda en proveedores de datos según el estándar OAI-PMH para el Sistema de Gestión de Documentos Históricos ArchiVenHIS se encuentra limitada ya que una parte de ellas fueron seleccionadas durante el desarrollo del sistema.

### 1.3.1 Metodología de desarrollo de *software* RUP

RUP es un proceso de desarrollo de *software*, además de ser una metodología para la ingeniería de *software*, que define claramente quién, cómo, cuándo y qué debe hacerse en el proyecto (23). Se caracteriza por ser dirigido por casos de uso, estar centrado en la arquitectura y por ser iterativo e incremental. En los anexos (ilustración 4) se puede presenciar en cuantas etapas se divide RUP (24):

- **Inicio:** el objetivo de esta etapa es determinar la visión del proyecto.
- **Elaboración:** el objetivo de esta etapa es determinar la arquitectura óptima.
- **Construcción:** el objetivo de esta etapa es llevar a obtener la capacidad operacional inicial.
- **Transición:** el objetivo de esta etapa es llegar a obtener el reléase del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, el 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.

### 1.3.2 Lenguajes

Para la implementación del módulo sé utilizarán los siguientes lenguajes:

**PHP**

PHP es un acrónimo recursivo que significa PHP Hypertext Pre-processor. Fue creado originalmente por Rasmus Lerdorf en 1994. Dentro de las operaciones que pueden realizarse con este lenguaje se encuentran: procesar la información de formularios, generar páginas con contenidos dinámicos, o enviar y recibir *cookies*, entre otras. Para su funcionamiento necesita tener instalado Apache o IIS con las librerías de PHP. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas.

Entre sus principales ventajas se puede mencionar (25):

- El código fuente escrito en PHP es invisible al navegador web y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- Posee una amplia documentación en su sitio Web oficial.
- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.

- PHP generalmente es utilizado como módulo de Apache, lo que lo hace extremadamente veloz. Está completamente escrito en C, así que se ejecuta rápidamente utilizando poca memoria.
- Puede interactuar con varios motores de bases de datos tales como Oracle, PostgreSQL y MySQL.

**HTML**

HTML (lenguaje de marcado de hipertexto) es un lenguaje de marcado diseñado para estructurar textos y presentarlos en forma de hipertextos, que es el formato estándar de las páginas web. Fue creado por Tim Berners-Lee en 1986.

HTML presenta diversas ventajas dentro de las cuales se pueden encontrar las siguientes (26):

- No necesita de grandes conocimientos cuando se cuenta con un editor de páginas web.
- Archivos pequeños.
- Despliegue rápido.
- Lenguaje de fácil aprendizaje.
- Lo admiten todos los navegadores web.

Sus principales desventajas son las siguientes (26):

- Lenguaje estático.
- La interpretación de cada navegador puede ser diferente.
- Guarda muchas etiquetas que pueden convertirse en “basura” y dificultan la corrección.
- El diseño es más lento.
- Las etiquetas son muy limitadas.

**CSS**

Las Hojas de Estilo en Cascada o simplemente CSS (por sus siglas en inglés, *Cascading Style Sheets*) es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas (27).

A continuación se describen algunas ventajas de usar CSS (28):

- Separación del contenido y presentación.

- Flexibilidad.
- Optimización de los tiempos de carga y de tráfico en el servidor.
- Precisión o elasticidad.
- Accesibilidad y estructuración.
- Limpieza del código fuente.
- Estandarización frente a especificaciones propietarias.
- Permite la diferenciación de estilos para imprimir / visualizar en pantalla.

**JavaScript**

JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos (29). Fue creado por Brendan Eich en la empresa Netscape Communications. Tiene como características principales las siguientes:

- Es utilizado principalmente para crear páginas web dinámicas.
- No es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de herencia.
- No es necesario declarar los tipos de variables que van a utilizarse.
- No puede escribir automáticamente al disco duro.

JavaScript presenta disimiles ventajas dentro de las cuales se destacan que es fácil de integrar con otros lenguajes y que es compatible con la mayoría de los navegadores modernos. Su principal desventaja está en que los usuarios pueden deshabilitar JavaScript en su navegador.

**AJAX**

AJAX, es el acrónimo para Asynchronous JavaScript + XML, en realidad no es una tecnología sino la combinación de muchas tecnologías como son: XHTML y CSS para crear una presentación basada en estándares, DOM (Document Object Model) para la interacción y manipulación dinámica de la presentación, XML, XSLT y JSON, para el intercambio y la manipulación de información, XMLHttpRequest para el intercambio asíncrono de información y JavaScript para unir todas las demás tecnologías (30).

Las razones para usar Ajax son las siguientes (31):

- Basado en los estándares abiertos.
- Válido en cualquier plataforma y navegador.
- Beneficia las aplicaciones web.
- No es difícil su utilización.
- Compatible con Flash.

- Es independiente del tipo de tecnología de servidor que se utilice.
- Mejora la estética de la Web.

**XML**

XML es un formato simple de texto muy flexible derivado del Estándar de Lenguaje de Marcado Generalizado (SGML, por sus siglas en inglés *Standard Generalized Markup Language*) que fue desarrollado por el Consorcio World Wide Web. XML está desempeñando un papel cada vez más importante en el intercambio de una amplia variedad de datos en la Web y en otros lugares (32).

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil (33).

## Lenguaje de modelado (UML)

Para evidenciar la eficiencia de la arquitectura propuesta, se necesita un lenguaje para el modelado de los procesos necesarios para el intercambio de descripciones de documentos. Los lenguajes de modelado son un conjunto estandarizado de símbolos y sus relaciones que permiten expresar un Sistema de Información mediante un esquema teórico, el cual representará su diseño (34).

El Lenguaje Unificado de Modelado o UML (por sus siglas en inglés, *Unified Modeling Language*) es un lenguaje para especificar, visualizar, construir y documentar los artefactos de los sistemas de *software*, así como para el modelado del negocio. Surgió en 1994 por iniciativa de Grady Booch y James Rumbaugh para combinar sus dos famosos métodos: el de Booch y el OMT (*Object Modeling Technique*), pero no fue hasta 1997 que fue adoptado como estándar por el OMG (*Object Management Group*).

Este lenguaje de modelado es útil para resolver una gran diversidad de problemas de ingeniería, desde procesos sencillos y aplicaciones de un sólo usuario a sistemas concurrentes y distribuidos. UML permite crear los diagramas que se van generando durante el proceso de ingeniería en el desarrollo de un sistema informático.

### 1.3.3 Sistema gestor de base de datos MySQL

MySQL es un sistema de gestión de bases de datos relacional (RDBMS). Este programa es capaz de almacenar una gran cantidad de datos y de distribuirlos para satisfacer las necesidades de cualquier tipo de organización. MySQL en sus inicios se consideraba la opción ideal para sitios web; pero actualmente incorpora muchas de las funciones necesarias para otros entornos sin perder su velocidad.

Algunas de las razones que conllevan a escoger a MySQL como gestor de datos son (35):

- **Coste:** el coste de MySQL es gratuito para la mayor parte de los usos.
- **Portabilidad:** MySQL se ejecuta en la gran mayoría de los sistemas operativos y en la mayor parte de los casos los datos se pueden trasladar de un sistema a otro sin presentar dificultad alguna.
- **Facilidad de uso:** MySQL es un sistema fácil de utilizar y de administrar. Las herramientas de MySQL son fuertes y flexibles, sin sacrificar su capacidad de uso.

### 1.3.4 Servidor web Apache

Apache es un Servidor HTTP de dominio público basado en el sistema operativo Linux. Fue desarrollado en 1995 y es actualmente uno de los servidores HTTP más utilizados en la red (36). Este servidor web en la actualidad es usado por múltiples razones dentro de las que se puede encontrar que es una tecnología gratuita de código fuente abierto, que presenta diseño modular, además les da la posibilidad a los usuarios de personalizar la respuesta ante los posibles errores que se puedan dar en el servidor.

Este servidor está diseñado para ser un servidor web flexible y potente que puede funcionar sobre varias plataformas y entornos, además de ser compatible con una amplia variedad de sistemas operativos (Unix, Windows, Macintosh, entre otros).

### 1.3.5 Herramientas

**NetBeans**

NetBeans es un proyecto de código abierto con una gran base de usuarios, una comunidad en constante crecimiento y con cerca de cien socios en todo el mundo. Es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas (37). Tiene soporte para varios lenguejes incluyendo PHP, JavaScript, HTML y CSS.

Dentro de las características de NetBeans se pueden encontrar las siguientes:

- Posee una integración completa en términos de administración básica y avanzada de MySQL.
- El depurado de las aplicaciones es más sencillo.
- Autocompleta código de los lenguejes que soporta.
- Compatibilidad total con PHP 5.3.
- En multiplataforma

**Visual Paradigm**

Visual Paradigm es una herramienta CASE de modelado de UML que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite graficar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. También proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Presenta licencia gratuita y comercial. Es fácil de instalar y actualizar y compatible entre ediciones.

Dentro de las principales características de esta herramienta se pueden encontrar las siguientes (38):

- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- Ingeniería inversa - Código a modelo, código a diagrama.
- Generación de código - Modelo a código, diagrama a código.
- Generación de bases de datos: transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos: desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.
- Generador de informes para generación de documentación.
- Distribución automática de diagramas: reorganización de las figuras y conectores de los diagramas UML.
- Importación y exportación de ficheros XMI.
- Integración con Visio: dibujo de diagramas UML con plantillas (stencils) de MS Visio.
- Editor de figuras.

### 1.3.6 *Framework* de desarrollo

**JQuery**

JQuery es un *framework* de Javascript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, permitiendo manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX al sistema (39).

JQuery consiste en un único fichero JavaScript que contiene las funcionalidades comunes de DOM, eventos, efectos y AJAX. La característica principal de la biblioteca es que permite cambiar el contenido de una página web sin necesidad de recargarla, mediante la manipulación del árbol DOM y peticiones AJAX (40).

**CodeIgniter**

CodeIgniter es un conjunto de herramientas para personas que construyen su aplicación web usando PHP. Su principal objetivo es brindarle la facilidad a los usuarios de desarrollar proyectos con mayor rapidez de lo que alcanzaría si lo tuvieran que escribir desde cero, proporcionándole una serie de librerías para tareas comúnmente necesarias como: acceder a una base de datos, enviar *email*, validar datos de un formulario, mantener sesiones, manipular imágenes, entre otras.

Es importante destacar que este *framework* implementa el patrón arquitectónico Modelo-Vista-Controlador (MVC), que es un estándar de programación que puede ser utilizado tanto para la realización de sitios web como para programas tradicionales. Además, muchas de sus utilidades y modos de funcionamiento son opcionales, lo que hace que el usuario goce de una mayor libertad a la hora de desarrollar aplicación web.

CodeIgniter sin dudas presenta muchas características, pero en comparación con otros *frameworks* sus características más generales son las que se presentan a continuación:

- **Versatilidad:** es capaz de trabajar en la mayoría de los entornos o servidores, incluso en sistemas de alojamiento compartido, donde sólo se tiene un acceso por FTP para enviar los archivos al servidor y donde no se posee acceso a su configuración.
- **Compatibilidad:** es compatible con la versión 4 de PHP, lo que hace que se pueda utilizar en una amplia gama servidores, incluso en algunos de los más antiguos. Por supuesto, funciona correctamente también en PHP 5.
- **Flexibilidad:** define una manera específica para trabajar, pero en muchos de los casos se pueden seguir o no. Algunos módulos como el uso de plantillas son totalmente opcionales. Esto ayuda muchas veces también a que la curva de aprendizaje sea más sencilla al principio.

- **Ligereza:** el núcleo de CodeIgniter es bastante ligero, lo que permite que el servidor no se sobrecargue interpretando o ejecutando grandes porciones de código. La mayoría de los módulos o clases que ofrece se pueden cargar de manera opcional, sólo cuando se van a utilizar realmente.
- **Documentación en forma de tutorial:** la documentación que este *framework* presenta es fácil de seguir y de asimilar, porque está escrita en modo de tutorial. Esto no facilita mucho la referencia rápida, cuando ya se sabe acerca del *framework* y lo que se quiere es consultar sobre una función o un método determinado, pero para las personas que deseen iniciarse sin duda es muy beneficioso (41).

## Conclusiones del capítulo

Con la realización del presente capítulo han quedado definidos los conceptos más importantes para el correcto entendimiento de los términos usados en la presente investigación. Se realizó un estudio de los principales protocolos de intercambio y recuperación de información entre sistemas de información documental, donde se llegó a la conclusión de que el protocolo OAI-PMH es el más idóneo para darle solución al problema propuesto en el presente trabajo de diploma. Se realizó un estudio de las herramientas, lenguajes y metodología a utilizar en el desarrollo del módulo, fundamentando la selección de cada una de ellas.

# Capítulo 2. Características del módulo para la interconexión y búsqueda en proveedores de datos

Actualmente el Sistema de Gestión de Documentos Históricos ArchiVenHIS no es capaz de brindar información sobre el material custodiado por otras instituciones de archivo aun cuando estas presenten un sistema informático con mecanismos que garanticen la consulta de esta información.

Para darle solución al problema anterior se propone incorporarle al ArchiVenHIS una nueva funcionalidad la cual consiste en recuperar información descrita en otros sistemas de gestión de archivos históricos bajo las especificaciones del protocolo para el intercambio y recuperación de información OAI-PMH. Para esto se implementará un módulo el cual permitirá que ArchiVenHIS se comporte como un proveedor de servicios OAI-PMH.

En el presente capítulo se presenta el modelo de dominio para ayudar a la compresión del contexto del sistema, capturando los componentes más importantes con los que se trabajará. Se muestra los requerimientos funcionales a partir de los cuales se conforman los casos de uso del sistema. Además, se presentan los diagramas relacionados con el análisis y el diseño de la solución que se propone.

## 2.1 Modelo de dominio

Un modelo del dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las cosas que existen o los eventos que suceden en el entorno en que trabaja el sistema. El modelo del dominio se describe mediante diagramas de UML (especialmente mediante diagramas de clases) (42).

En el presente trabajo de diploma se realiza el modelo de dominio debido a que no existe un negocio definido, por lo cual, no se pueden determinar los procesos y roles del proceso de negocio, haciéndose engorroso y poco exacto la descripción de los mismos.

### 2.1.1 Diagrama de clases del modelo del dominio

Los diagramas de clases del modelo del dominio muestran a los clientes, usuarios, revisores y a otros desarrolladores las clases del dominio y como se relacionan unas con otras mediante asociaciones (42).

**Diagrama de clases del modelo del dominio**

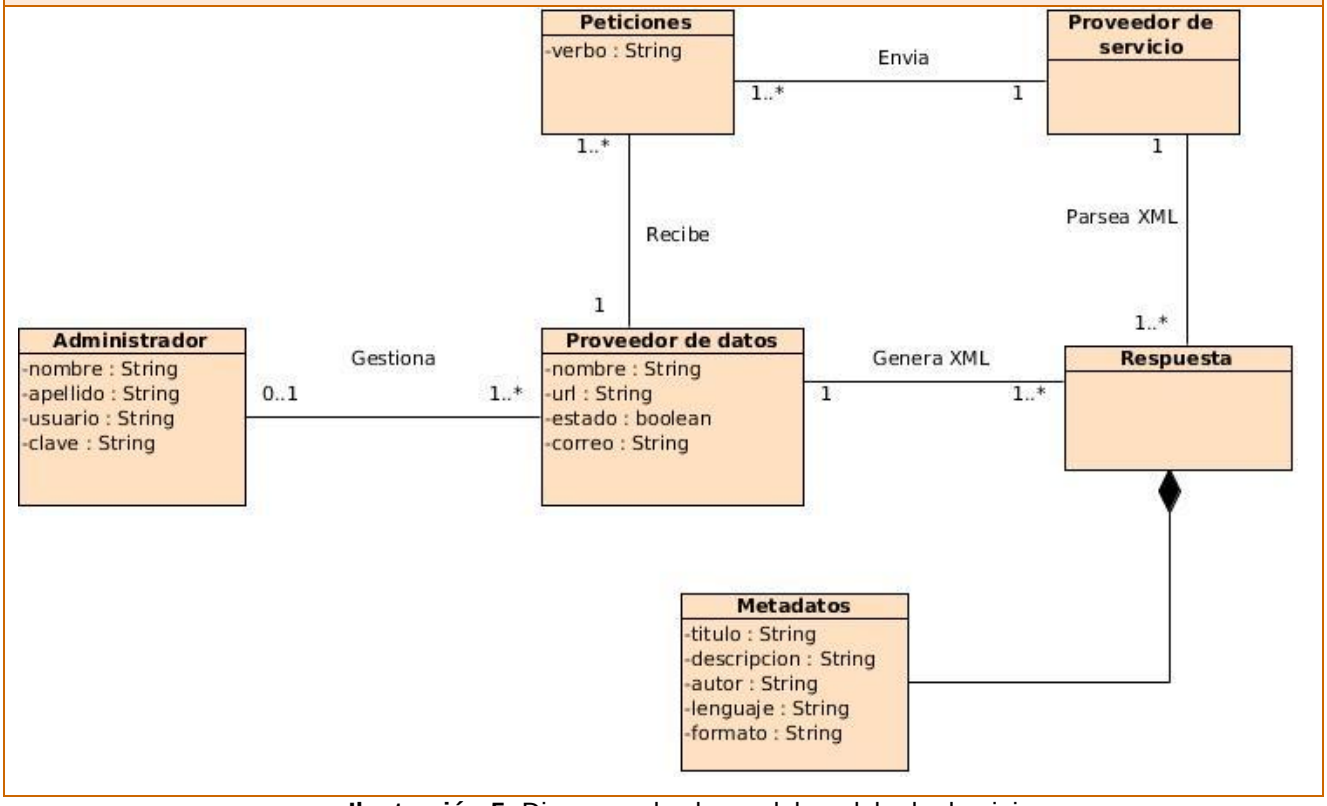



**Ilustración 5:** Diagrama de clases del modelo de dominio

## 2.2 Modelado del Sistema

### 2.2.1 Requerimientos funcionales

Los requerimientos funcionales son condiciones o capacidades que el sistema debe cumplir. A continuación se pueden observar los requerimientos que el sistema debe cumplir:

**RF1-** Adicionar repositorios OAI-PMH: permitir adicionar un nuevo repositorio oai-pmh.

**RF2-** Modificar repositorios OAI-PMH: permitir modificar los datos del repositorio oai-pmh seleccionado.

**RF3-** Eliminar repositorios OAI-PMH: permitir eliminar uno o varios repositorios.

**RF4-** Listar repositorios OAI-PMH: permite listar los repositorios que han sido adicionados.

**RF5-** Modificar estado del repositorio: permite modificar el estado del repositorio.

**RF6-** Actualizar los datos del repositorio: permite actualizar los datos del repositorio introducido por el usuario con los datos obtenidos desde el PD.

**RF7-** Indexar documentos por repositorios: permite indexar los documentos de uno o varios repositorios

**RF8-** Indexar documentos por agrupaciones: permite indexar los documentos de una o varias agrupaciones de un determinado repositorio.

**RF9-** Permitir realizar búsquedas sencillas: permite realizar búsquedas sencillas sobre los documentos indexados de los repositorios

**RF10-** Permitir realizar búsquedas avanzadas: permite realizar búsquedas avanzadas (título, repositorio, clasificación) sobre los documentos indexados de los repositorios

**RF11-** Mostrar los resultados: el sistema muestra los resultados de la búsqueda realizada.

**RF12-** Visualizar descripciones: el sistema muestra una breve descripción del documento.

### 2.2.2 Requerimientos no funcionales

Los requisitos no funcionales especifican propiedades del sistema, como restricciones del entorno o de la implementación, rendimiento, dependencia de la plataforma, facilidad de mantenimiento, extensibilidad y fiabilidad (42).

**Usabilidad**

- Ofrecer interfaz amigable, interactiva e intuitiva.

**Fiabilidad**

- El sistema debe estar disponible las 24 horas del día.

**Eficiencia**

- El sistema debe garantizar la gestión de la información, así como responder en un tiempo relativamente rápido a las peticiones de los usuarios.

**Soporte**

- Se debe implementar el sistema siguiendo el estándar de codificación definido en la línea de desarrollo.
- Los componentes de *software* que integran la solución se organizarán de forma modular.

**Restricciones del diseño**

- Lenguaje de programación: PHP 5.3
- *Framework* de desarrollo: CodeIgniter 1.6.3
- Entorno de desarrollo integrado: Netbeans 7.0
- Bibliotecas de clases: jQuery 1.4.4

**Requisitos para la documentación y ayuda del sistema.**

- El sistema notifica a los usuarios acerca de los resultados, exitosos o no, de las acciones

realizadas.

**Requisitos de *software*.**

- Tener instalado como sistema operativo en cada uno de los servidores Debían 6+.
- Tener instalado en el servidor de bases de datos el SGBD MySQL 5.1
- Tener instalado en el servidor web Apache 2.2 configurado adecuadamente para trabajar con CodeIgniter.

***Hardware***

- Servidor de bases de datos y servidor web con las siguientes prestaciones:
  - **RAM:** 2 GB+.
  - **Procesador:** Dos núcleos a 2.4 GHZ +.

**Requisitos de licencia**

- Se emplearán herramientas libres para el desarrollo del sistema, por lo que no será necesario adquirir licencias o patentes para su uso.

**Seguridad**

- El sistema debe garantizar la seguridad a través de la autenticación de los usuarios.
- El acceso a cada una de las funcionalidades que presenta la aplicación deberá estar protegido por permisos de acceso según los roles definidos en el sistema.

### 2.2.3 Actores del sistema

Los actores del sistema representan el rol que juega una o varias personas, un equipo o un sistema automatizado, son parte del sistema y pueden intercambiar información con él o ser recipientes pasivos de información. En la tabla que aparece a continuación se definen los actores que interactúan con el sistema.

| Actores | Justificación |
|---|---|
| Administrador del sistema | El administrador es el encargado de gestionar los usuarios internos del sistema. Además, podrá gestionar los repositorios OAI-PMH y realizar algunas funcionalidades sobre ellos como son: modificar estado, sincronizar los datos e indexar documentos de estos. |
| Usuario externo | El usuario externo podrá realizar búsquedas sobre los documentos incorporados al sistema, puede visualizar las imágenes asociadas a ellos o visualizar su descripción. Solicita los servicios que se brindan en el Archivo visualizando el estado de la solicitud. Además, podrá realizar búsquedas |

| | |
|---|---|
| | sobre los documentos indexados por el sistema desde los repositorios OAI-PMH. |

**Tabla 1:** Actores del sistema

### 2.2.4 Diagrama de casos de uso (CU) del sistema.

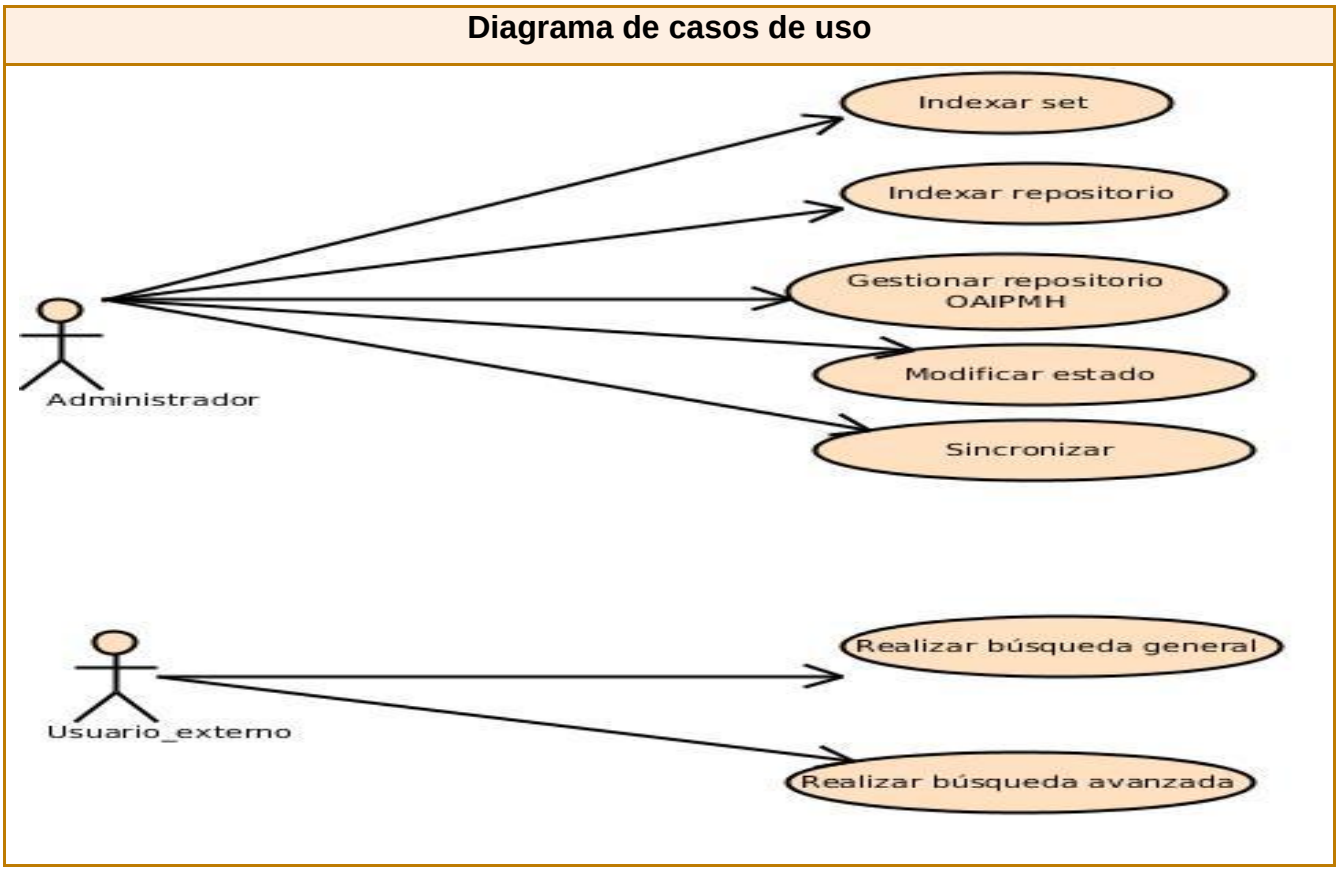



**Ilustración 6:** Diagrama de caso de uso

### 2.2.5 Patrón de caso de uso utilizado

El patrón de caso de uso utilizado en la solución del presente trabajo de diploma es el patrón CRUD Completo ya que permite de manera simple modelar los casos de uso que poseen las operaciones de crear, leer, actualizar y eliminar.

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

**Gestionar repositorio OAI-PMH antes de aplicar el CRUD Completo**

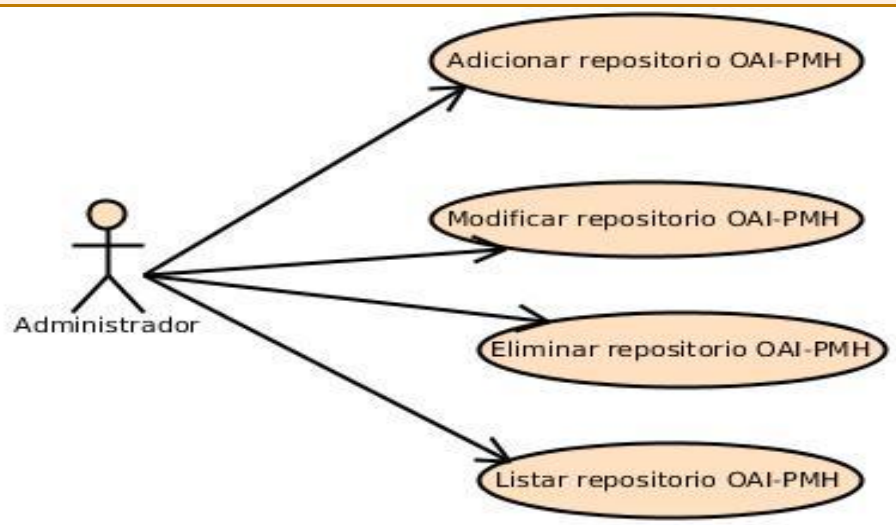



**Ilustración 7:** CU Gestionar repositorio OAI-PMH antes de aplicar el CRUD Completo

**Gestionar repositorio OAI-PMH después de aplicar el CRUD Completo**

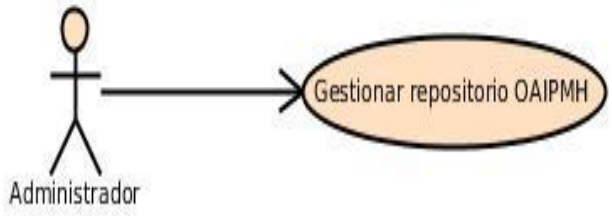



**Ilustración 8:** CU Gestionar repositorio OAI-PMH después de aplicar el CRUD Completo

### 2.2.6 Descripción de los Casos de Uso (CU) del Sistema

A continuación se ofrece una breve descripción de los casos de uso. Para ver una descripción más detallada de los casos de usos consultar el expediente del proyecto.

| | |
|---|---|
| **Caso de uso** | Gestionar repositorio OAI-PMH |
| **Actores** | Administrador |
| **Resumen** | El caso de uso inicia cuando el actor se autentica en el sistema y accede a gestionar repositorios. El actor procede a adicionar, modificar y eliminar un repositorio OAI-PMH y una vez ejecutada la acción por parte del sistema, finaliza el |

| | |
|---|---|
| | caso de uso. |
| **Referencias** | RF1, RF2, RF3, RF4 |

**Tabla 2:** Descripción del CU Gestionar repositorio OAI-PMH

| | |
|---|---|
| **Caso de uso** | Modificar estado |
| **Actores** | Administrador |
| **Resumen** | El caso de uso inicia cuando el actor se autentica en el sistema y accede a modificar estado. El actor procede a activar o desactivar un repositorio OAI-PMH y una vez ejecutada la acción por parte del sistema, finaliza el caso de uso. |
| **Referencias** | RF5 |

**Tabla 3:** Descripción del CU Modificar estado

| | |
|---|---|
| **Caso de uso** | Sincronizar |
| **Actores** | Administrador |
| **Resumen** | El caso de uso inicia cuando el actor se autentica en el sistema y procede a sincronizar los datos entrados por el usuario con los obtenidos del repositorio y una vez ejecutada la acción por parte del sistema, finaliza el caso de uso. |
| **Referencias** | RF6 |

**Tabla 4:** Descripción del CU Sincronizar

| | |
|---|---|
| **Caso de uso** | Indexar repositorio |
| **Actores** | Administrador |
| **Resumen** | El caso de uso inicia cuando el actor se autentica y decide indexar los metadatos de algunos de los repositorios registrados en el sistema. |
| **Referencias** | RF7 |

**Tabla 5:** Descripción del CU Indexar repositorio

| | |
|---|---|
| **Caso de uso** | Indexar set |
| **Actores** | Administrador |
| **Resumen** | El caso de uso inicia cuando el actor se autentica y decide indexar los metadatos de algunos de los set de los repositorios registrados en el sistema. |
| **Referencias** | RF8 |

**Tabla 6:** Descripción del CU Indexar set

| | |
|---|---|
| **Caso de uso** | Realizar búsqueda general |
| **Actores** | Usuario externo |
| **Resumen** | El caso de uso inicia cuando el actor se autentica en el sistema y procede a realizar una búsqueda general sobre los documentos indexados de los repositorios y una vez ejecutada la acción por parte del sistema, finaliza el caso de uso. |
| **Referencias** | RF9, RF11, RF12 |

**Tabla 7:** Descripción del CU Realizar búsquedas general

| | |
|---|---|
| **Caso de uso** | Realizar búsqueda avanzada |
| **Actores** | Usuario externo |
| **Resumen** | El caso de uso inicia cuando el actor se autentica en el sistema y procede a realizar una búsqueda avanzada sobre los documentos indexados filtrando la búsqueda por repositorio y por clasificación, y una vez ejecutada la acción por parte del sistema, finaliza el caso de uso. |
| **Referencias** | RF10, RF11, RF12 |

**Tabla 8:** Descripción del CU Realizar búsqueda avanzada

## 2.3 Análisis

El análisis se desarrolla fundamentalmente dentro de la fase de elaboración. Puede considerarse como una primera aproximación al modelo de diseño. Este consiste en obtener una visión del sistema que se preocupa de ver qué hace, de modo que sólo se interesa por los requisitos funcionales (43).

Un diagrama de clases del análisis es un artefacto en el que se representan los conceptos en un dominio del problema. Representa el funcionamiento del mundo real, no de la implementación automatizada del mismo. A continuación se nombran los tipos de clases que son utilizados en el modelo de análisis (42):

**CI_ [Nombre de la Clase]:** estas clases modelan la interacción entre los usuarios y el sistema, es decir, ventanas, formularios, dispositivos, sistemas externos, etc.

**CC_ [Nombre de la Clase]:** estas clases encapsulan el comportamiento de cada caso de uso y coordinan el trabajo de las clases interfaz y entidad.

**CE_ [Nombre de la Clase]:** estas clases modelan toda la información del sistema que posee una vida larga y que puede ser persistente.

A continuación se muestran los diagramas de clases del análisis de los casos de usos críticos.

**Diagrama de clases del análisis CU Gestionar repositorio OAI-PMH**

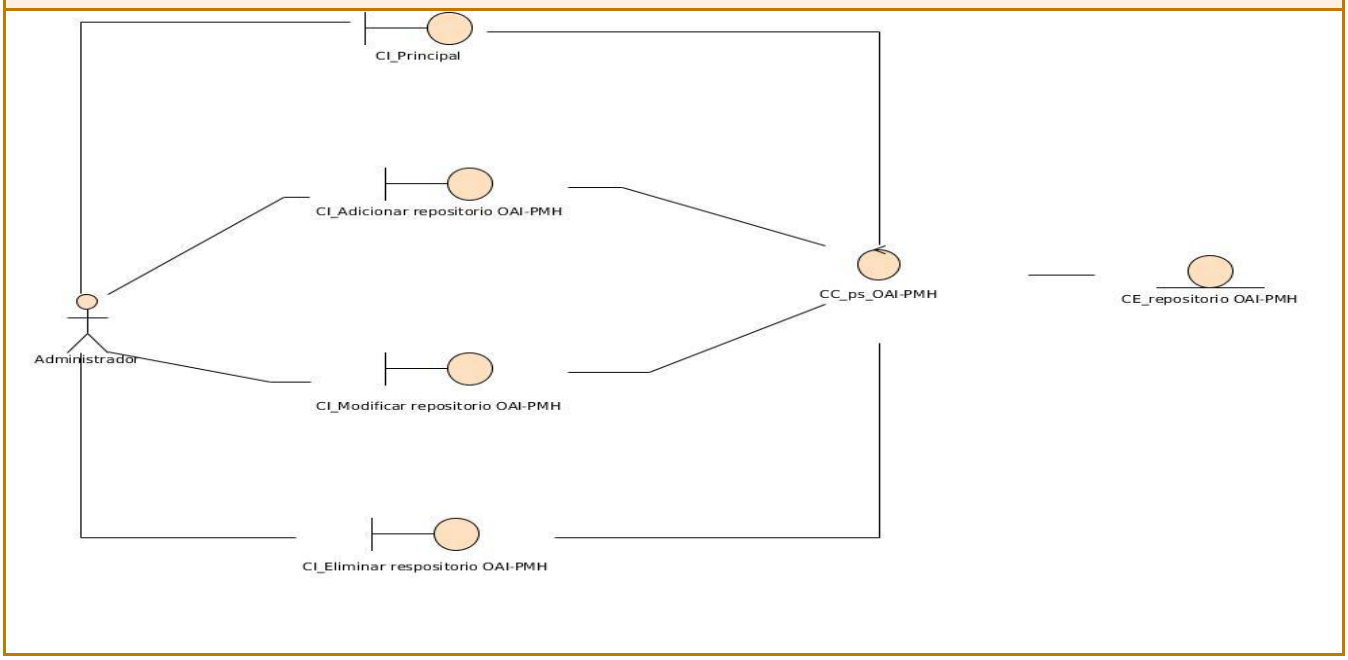



**Ilustración 9:** Diagrama de clases del análisis CU Gestionar repositorio OAI-PMH

**Diagrama de clases del análisis CU Modificar estado**

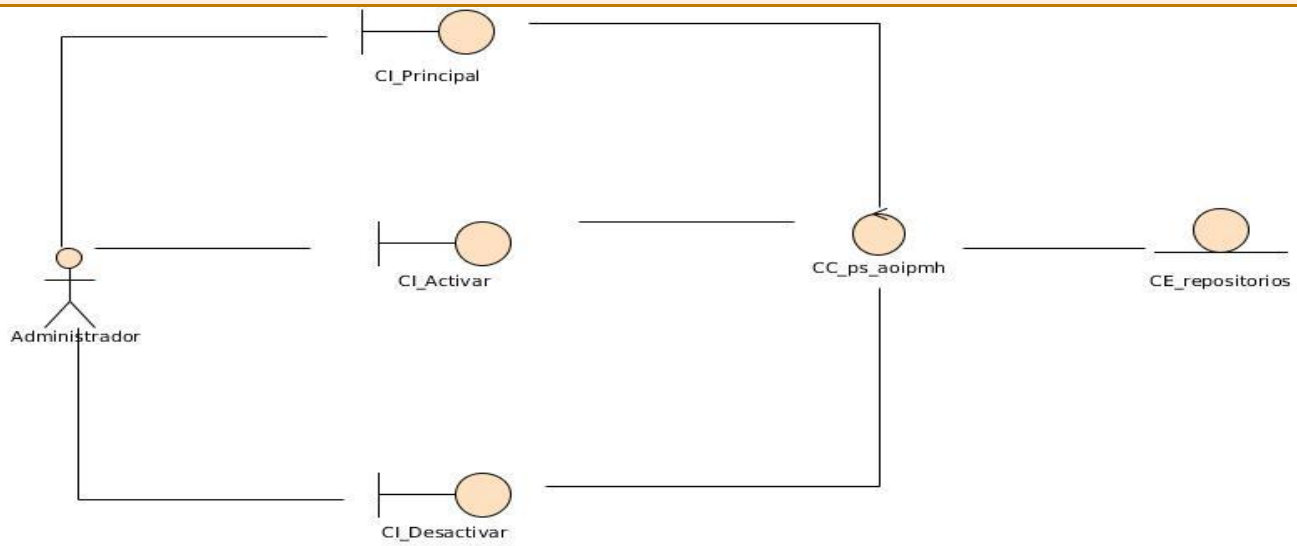



**Ilustración 10:** Diagrama de clases del análisis CU Modificar estado

**Diagrama de clases del análisis CU  Indexar repositorio**

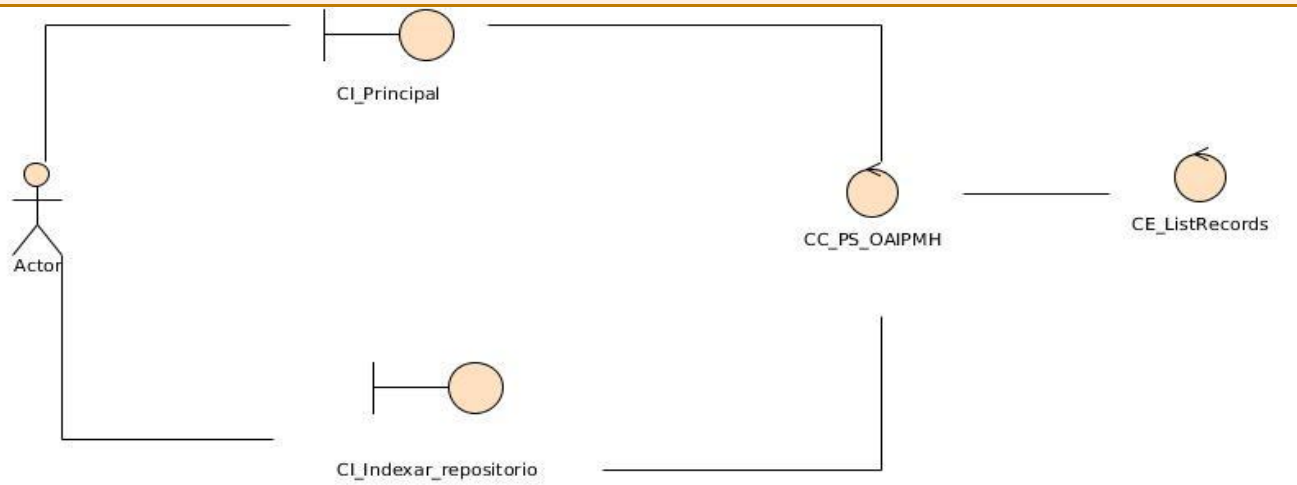



**Ilustración 11:** Diagrama de clases del análisis CU  Indexar repositorio

**Diagrama de clases del análisis CU  Indexar set**

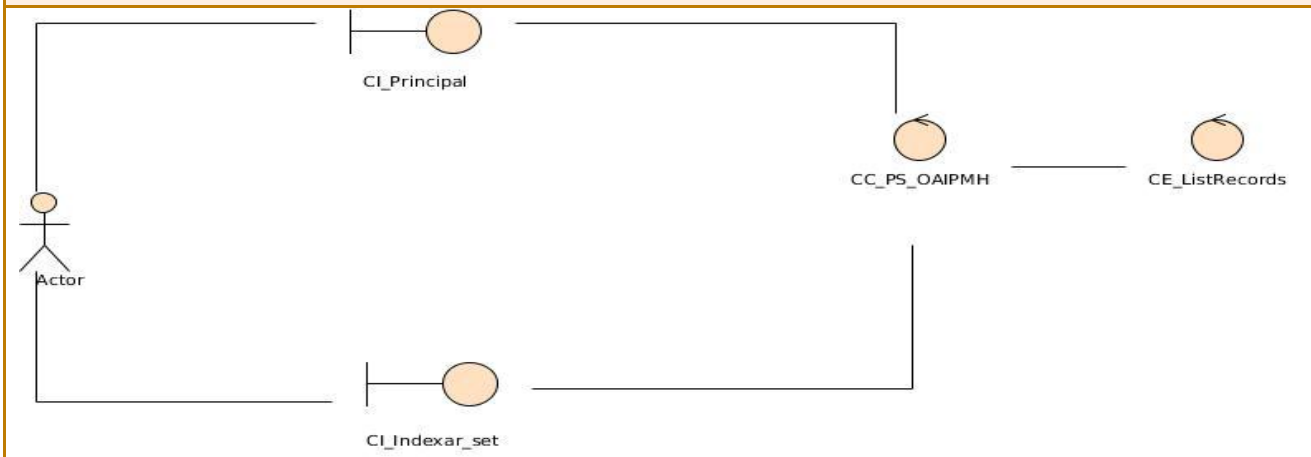



**Ilustración 12:** Diagrama de clases del análisis CU  Indexar set

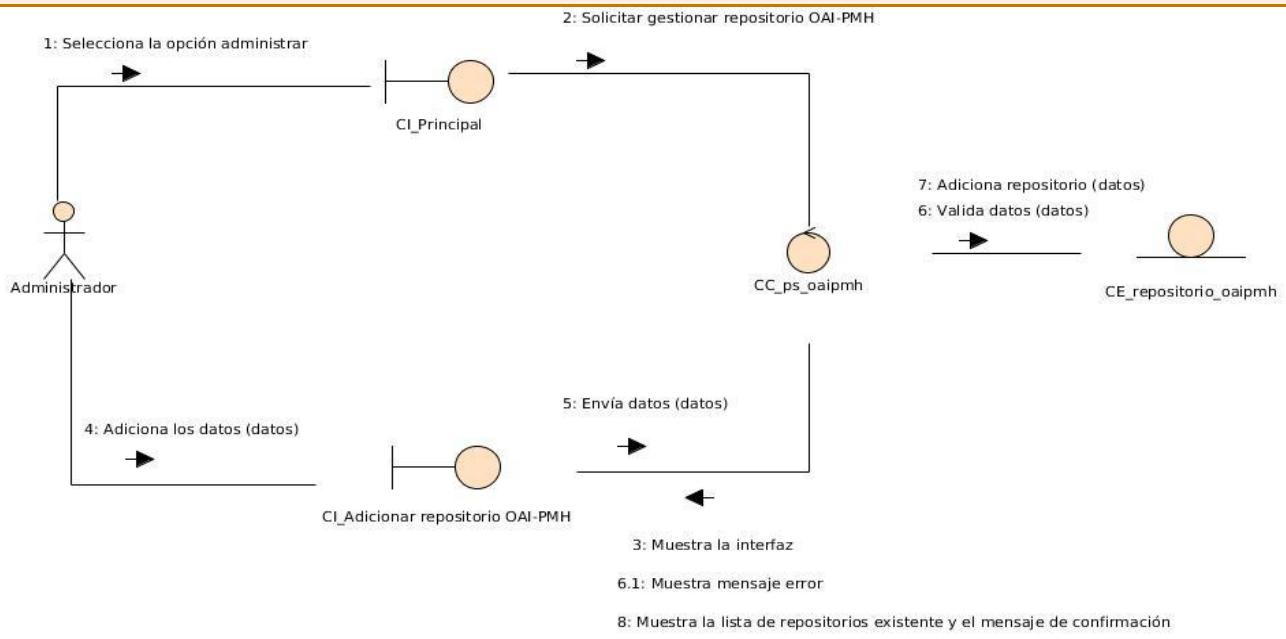
## 3.1 Diagrama de interacción

El término diagrama de interacción es una generalización de dos tipos de diagramas, los de colaboración y los de secuencia, ambos pueden utilizarse para representar de forma similar la interacción entre los mensajes.

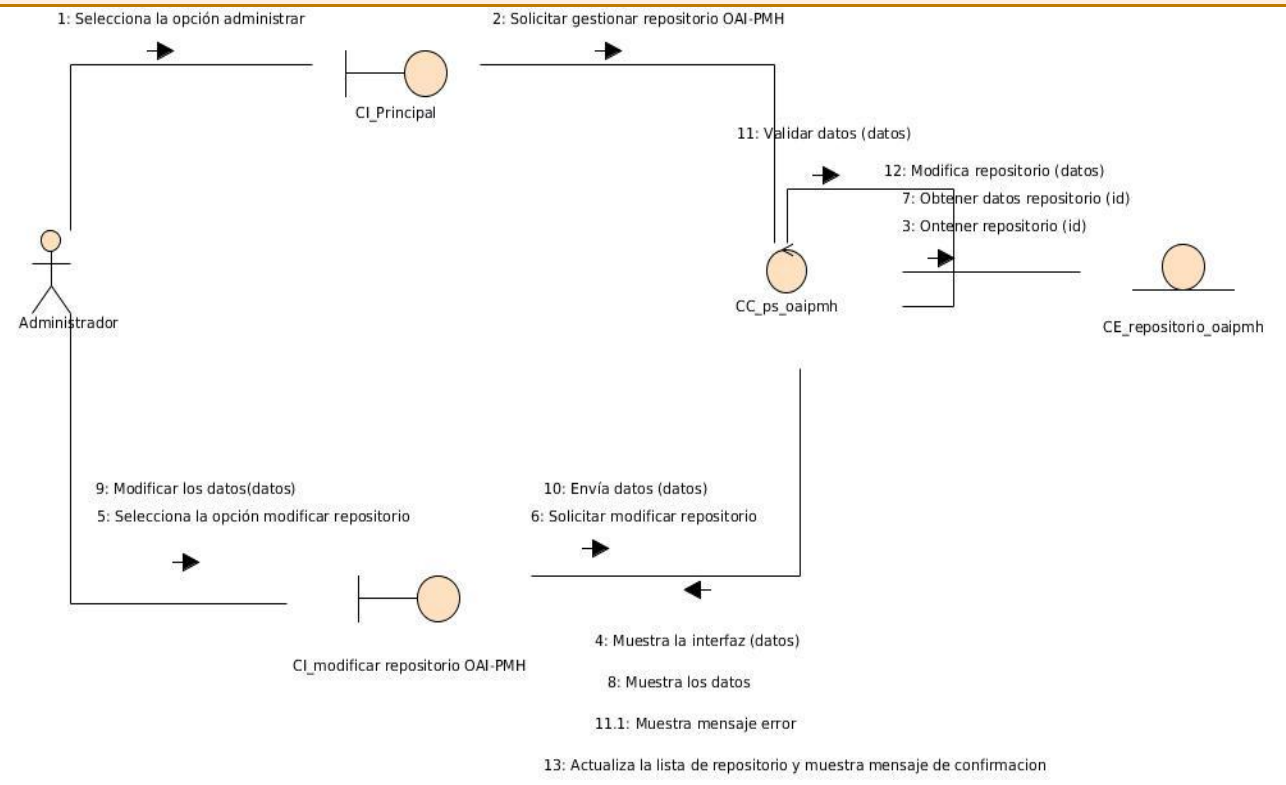
Los diagramas de colaboración destacan la organización estructural de los objetos que envían y reciben mensajes. A continuación se pueden observar los diagramas de colaboración correspondientes a los casos de usos críticos.

**Diagrama de interacción del CU Gestionar repositorio OAI-PMH**

**Sección "Adicionar repositorio OAI-PMH"**




**Sección "Modificar repositorio OAI-PMH"**



**Sección "Eliminar repositorio OAI-PMH"**

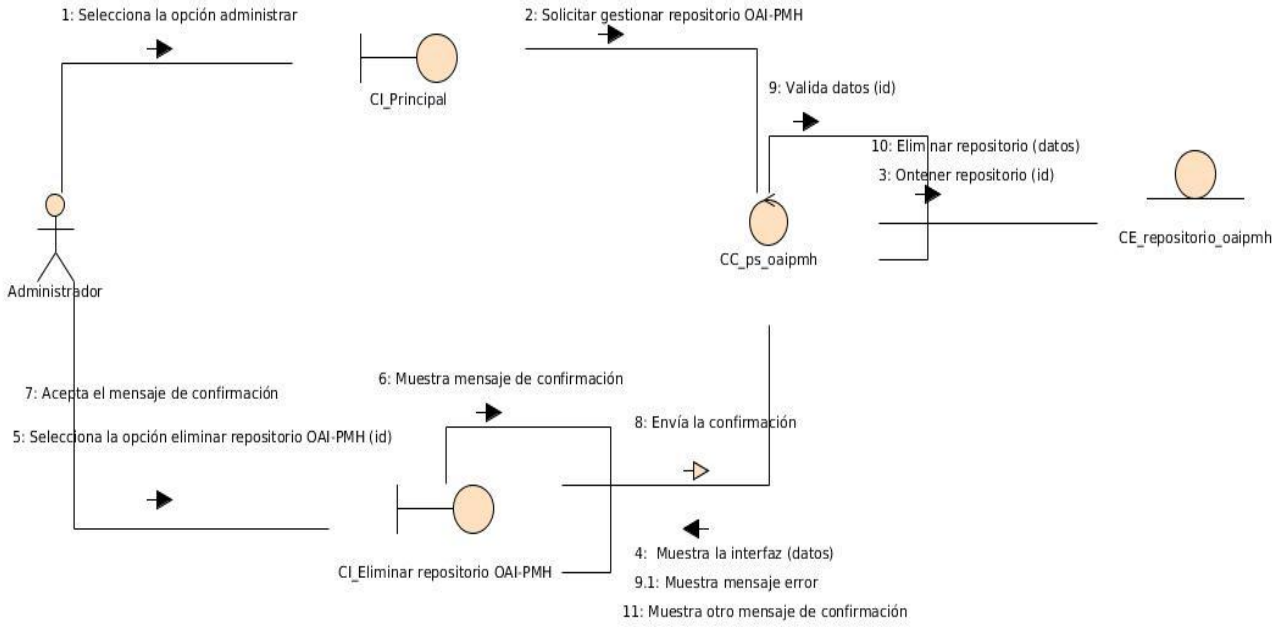
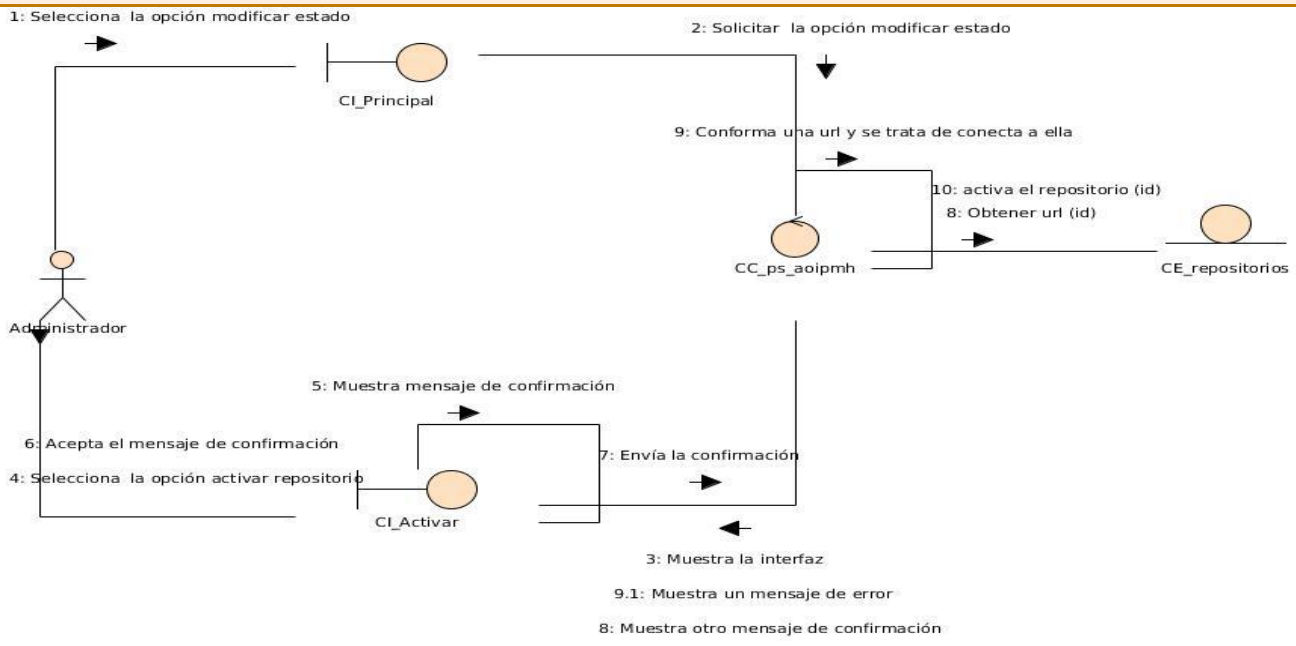



**Ilustración 12:** Diagrama de interacción del CU Gestionar repositorio OAI-PMH

**Diagrama de interacción del CU Modificar estado**

**Sección "Activar"**



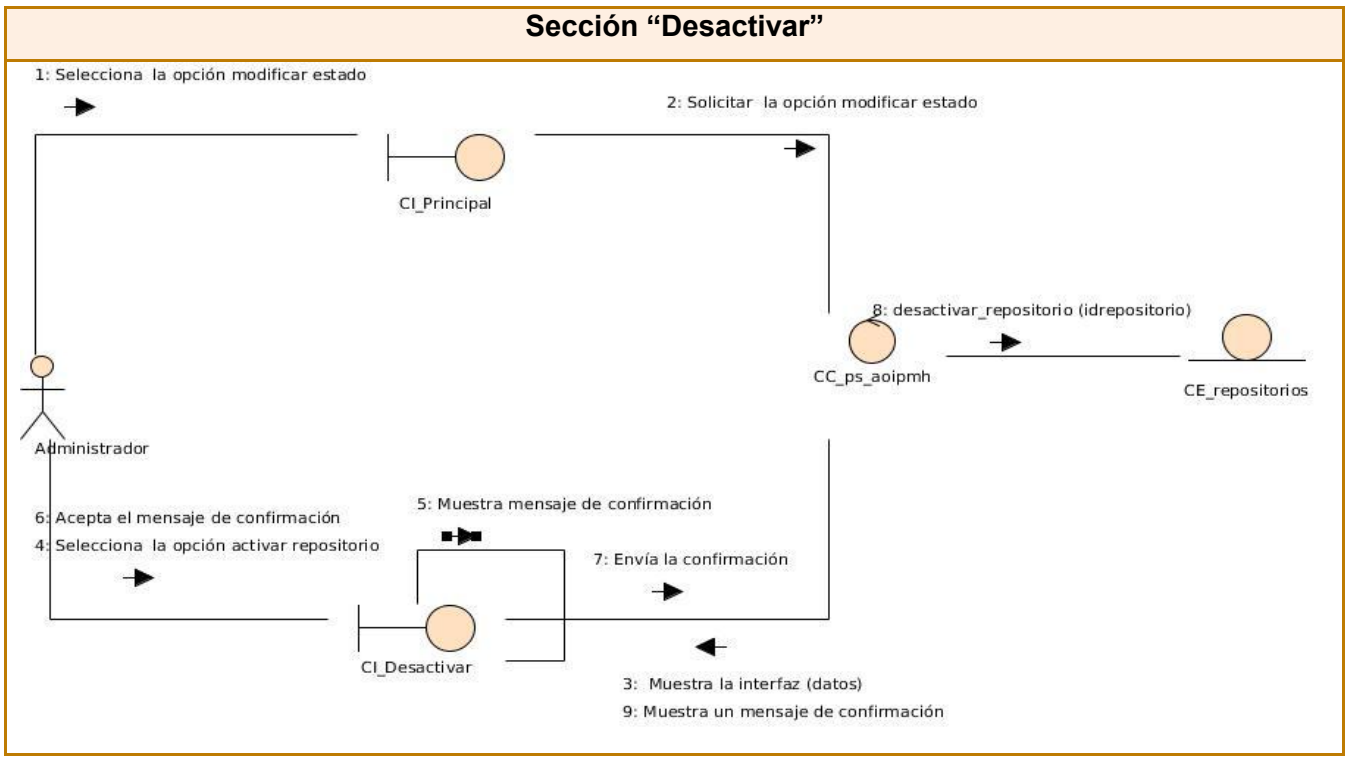



**Ilustración 13:** Diagrama de interacción del CU Modificar estado

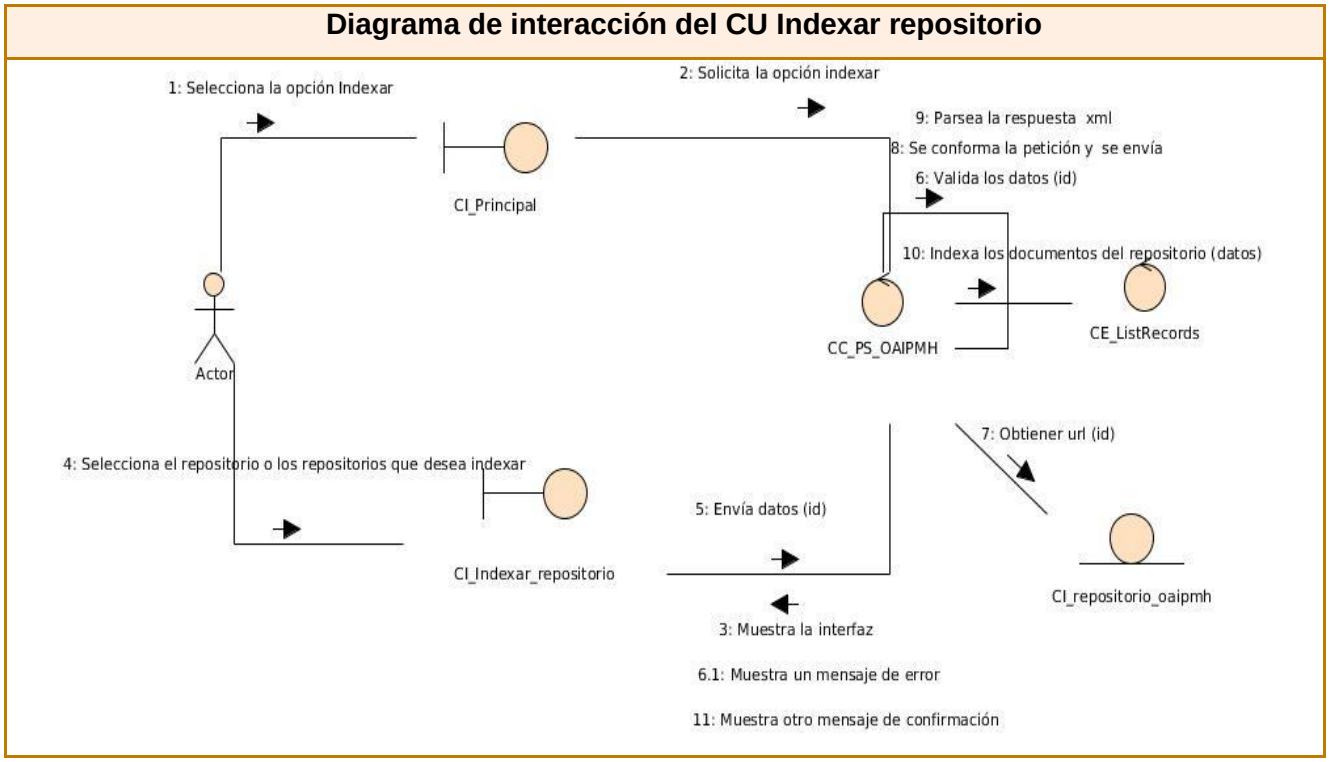



**Ilustración 14:** Diagrama de interacción del CU Indexar repositorio

**Diagrama de interacción del CU Indexar set**

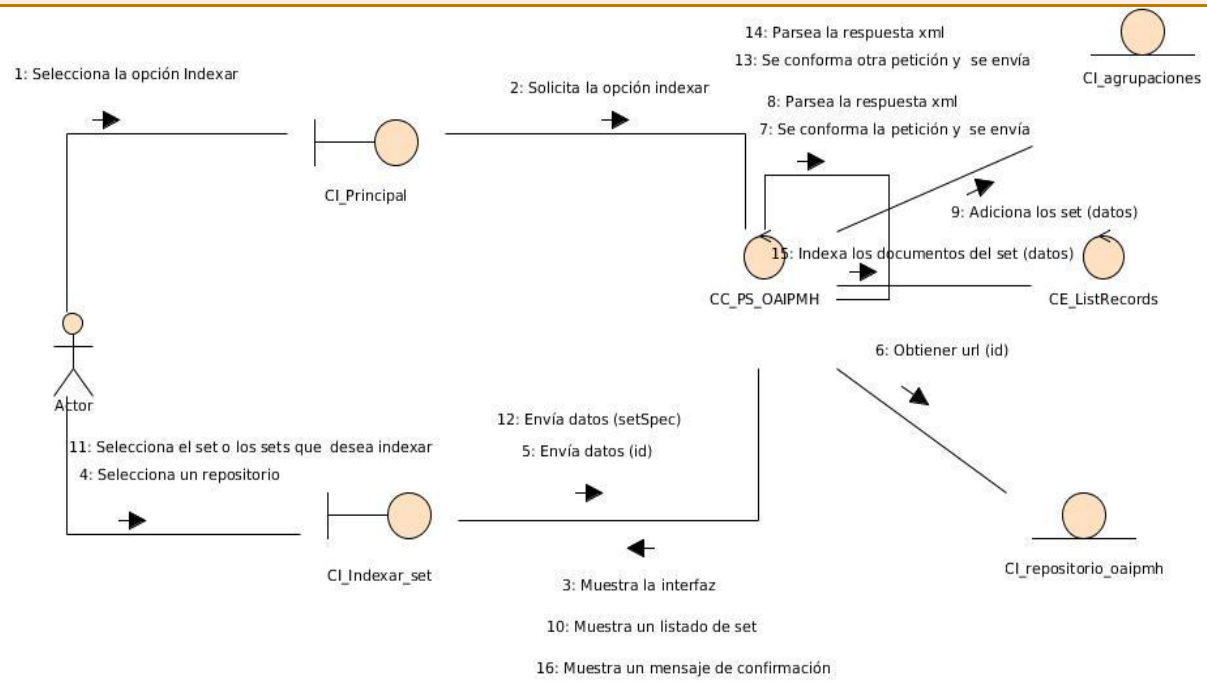



**Ilustración 15:** Diagrama de interacción del CU Indexar set

## 2.4 Diseño

El diseño se realiza al final de la fase de elaboración después del análisis. Este es un refinamiento del análisis que permite mediante el modelo de diseño describir la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar (42).

Un diagrama de clase del diseño describe gráficamente las especificaciones de las clases de *software*, de las interfaces, así como sus relaciones en una aplicación. A continuación se muestran los diagramas de clases del diseño de los casos de usos críticos.

**Diagrama de clase del diseño del CU Gestionar repositorio OAI-PMH**

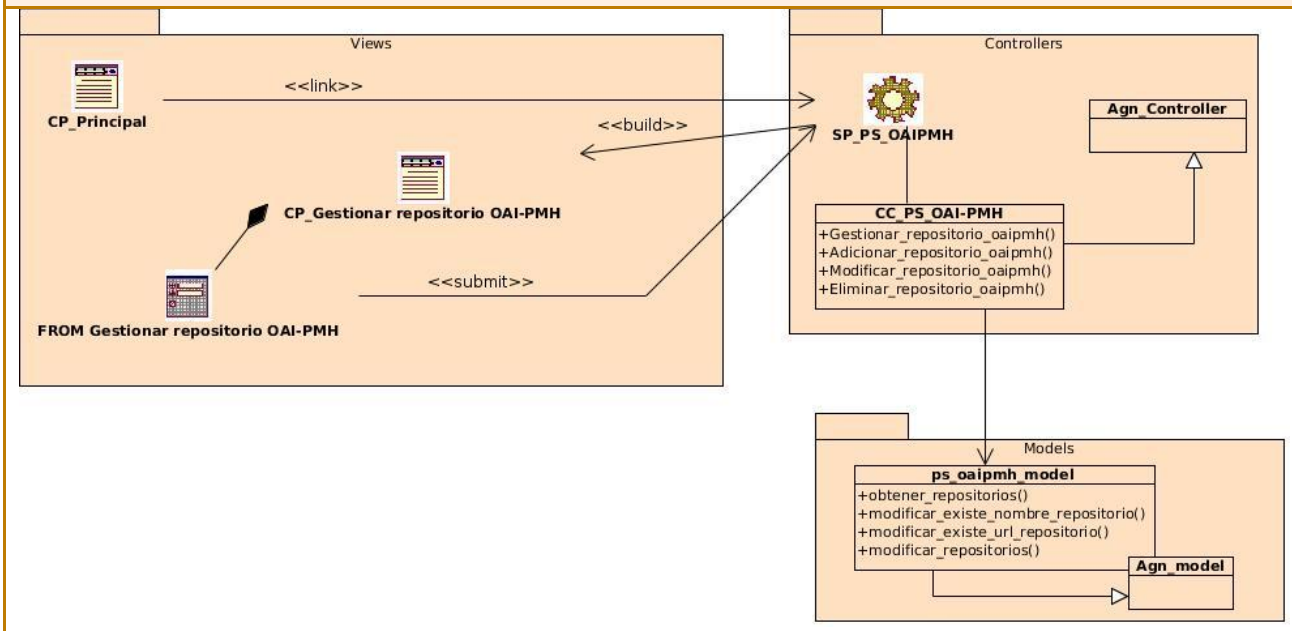



**Ilustración 16:** Diagrama de clase del diseño del CU Gestionar repositorio OAI-PMH

**Diagrama de clase del diseño del CU Modificar estado**

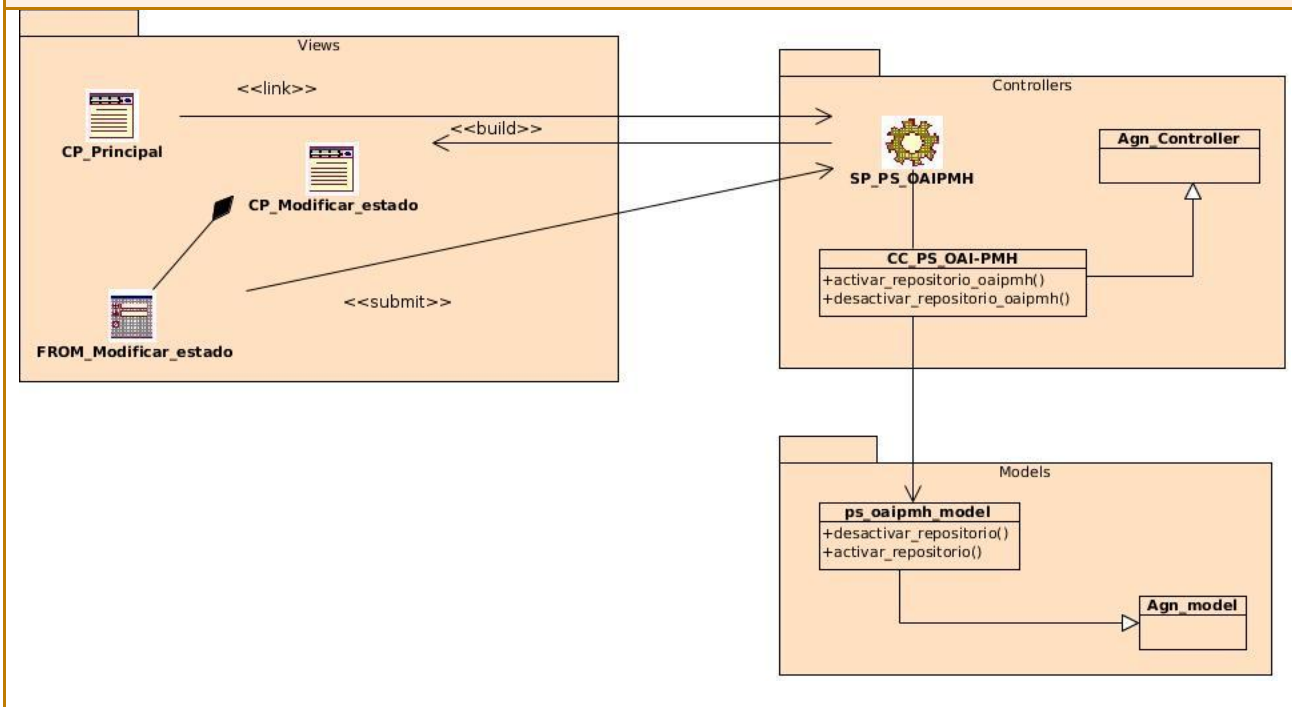



**Ilustración 17:** Diagrama de clase del diseño del CU Modificar estado

**Diagrama de clase del diseño del CU Indexar repositorio**

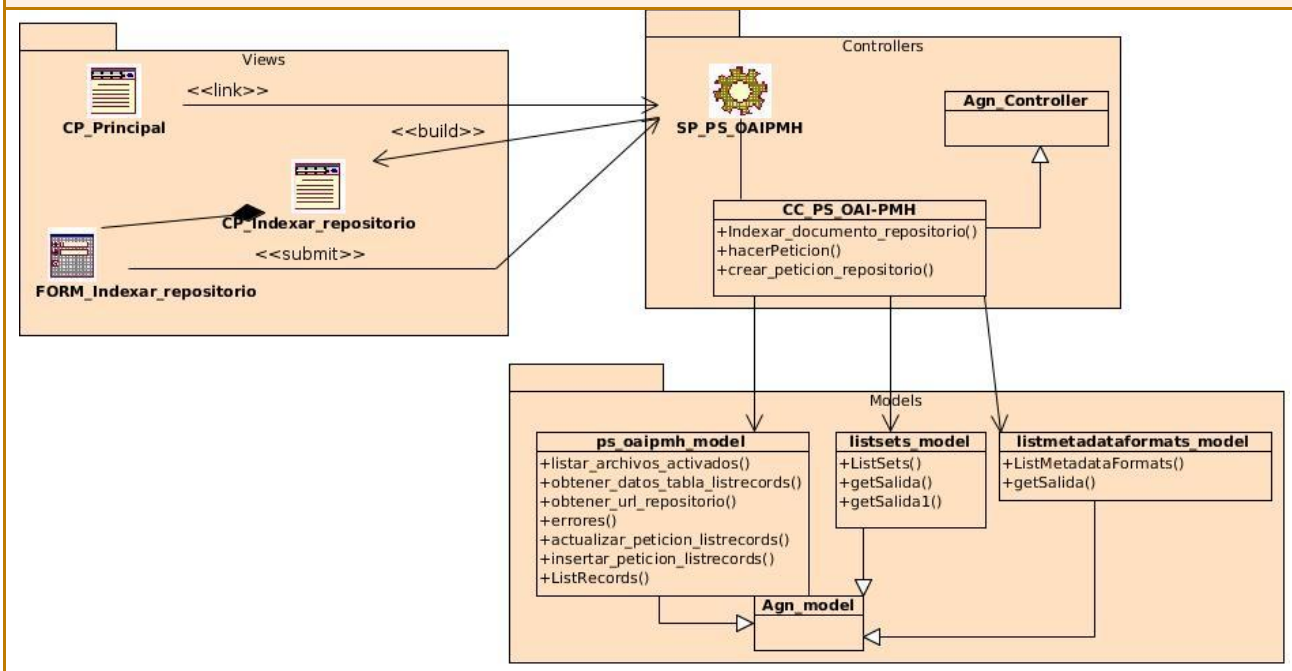



**Ilustración 18:** Diagrama de clase del diseño del CU Indexar repositorio

**Diagrama de clase del diseño del CU Indexar set**

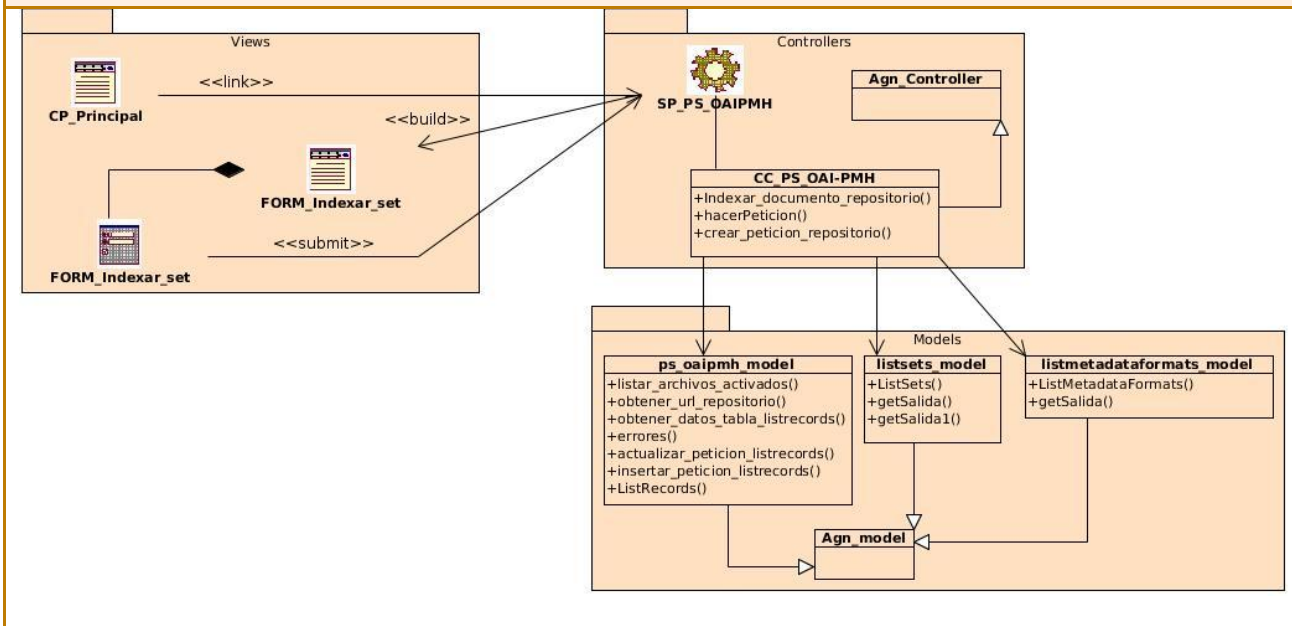



**Ilustración 19:** Diagrama de clase del diseño del CU Indexar set

## 2.5 Modelo de datos

Un modelo de datos es una colección de herramientas conceptuales para describir los datos, las relaciones que existen entre ellos, semántica asociada a los datos y restricciones de consistencia.

**Modelo de datos**

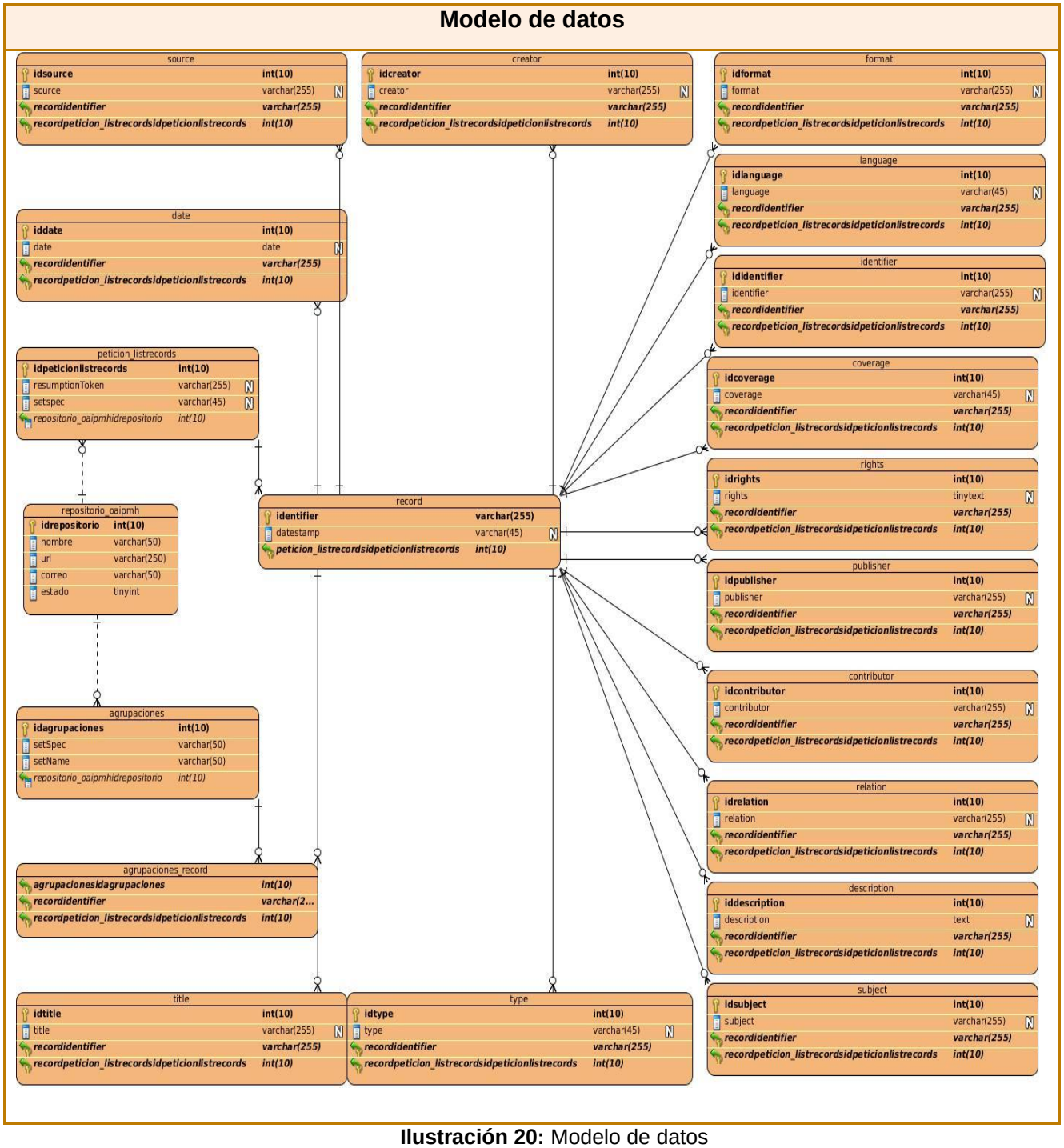



**Ilustración 20:** Modelo de datos

### 2.5.1 Descripción del modelo de datos

| **Tabla** | repositorio_oaipmh | |
|---|---|---|
| **Descripción** | En esta tabla se guardan los datos de los repositorios OAI-PMH | |
| **Atributos** | **Tipo** | **Descripción** |
| idrepositorio | int(10) | Identifica las tupas en la tabla, valor generado auto incrementalmente por el gestor |
| nombre | varchar(50) | Nombre del repositorio |
| url | varchar(250) | URL del repositorio |
| correo | varchar(50) | Correo del administrador del repositorio |
| estado | tinyint | Estado del repositorio. Toma valor uno (1) si está activado (disponible para indexar) o cero (0) en caso contrario |

**Tabla 9:** Modelo de datos Entidad: repositorio_oaipmh

| **Tabla** | peticion_listrecords | |
|---|---|---|
| **Descripción** | En esta tabla se guardan los datos que se utilizan para realizar las peticiones de este tipo | |
| **Atributos** | **Tipo** | **Descripción** |
| idpeticionlistrecord | int(10) | Identifica las tupas en la tabla, valor generado auto incrementalmente por el gestor |
| resumptiontoken | varchar(255) | Parámetro de reanudación |
| setSpect | varchar(50) | Identificador de un *set* |
| repositorio_oaipmhidrepocitorio | int(10) | Identifica el repositorio al que se le está haciendo la petición |

**Tabla 10:** Modelo de datos Entidad: peticion_listrecords

| **Tabla** | agrupaciones | |
|---|---|---|
| **Descripción** | Guarda los datos de las distintas agrupaciones | |
| **Atributos** | **Tipo** | **Descripción** |
| idagrupaciones | int(10) | Identifica las tupas en la tabla, valor generado auto incrementalmente por el gestor |
| setSpect | varchar(50) | Identificador del set |
| setName | varchar(50) | Nombre del set |
| repositorio_oaipmhidrepositorio | int(10) | Identifica el repositorio al que pertenece |

**Tabla 11:** Modelo de datos Entidad: agrupaciones

| **Tabla** | record | |
|---|---|---|
| **Descripción** | En esta tabla se guardan los datos de la cabecera que devuelven las peticiones de tipo ListRecords | |
| **Atributos** | **Tipo** | **Descripción** |
| recordidentifier | varchar(255) | Identificador del registro |
| datestamp | timestamp | Fecha de creación del registro |
| peticion_listrecordsidpeticionlistrecords | int(10) | Identifica a la petición que recolectó el *record* |

**Tabla 12:** Modelo de datos Entidad: record

| **Tabla** | agrupaciones_record | |
|---|---|---|
| **Descripción** | Permite determinar a cual o cuales *set* pertenecen los record recolectados | |
| **Atributos** | **Tipo** | **Descripción** |

| | | |
|---|---|---|
| agrupacionesidagrupaciones | int(10) | Identifica a la agrupación a la que pertenece el record |
| recordidentifier | varchar(255) | Identificador del registro |
| recordpeticion_listrecordsidpeticionlistrecords | int(10) | Identifica a la petición que recolectó el record |

**Tabla 13:** Modelo de datos Entidad: agrupaciones_record

| **Tabla** | title | |
|---|---|---|
| **Descripción** | En esta tabla se guardan los títulos asociados con un documento | |
| **Atributos** | **Tipo** | **Descripción** |
| idtitle | int(10) | Identifica las tupas en la tabla, valor generado auto incrementalmente por el gestor |
| title | varchar(255) | Título del documento |
| recordidentifier | varchar(255) | Identificador del registro |
| recordpeticion_listrecordsidpeticionlistrecords | int(10) | Identifica a la petición que recolectó el *record* |

**Tabla 14:** Modelo de datos Entidad: title

**Nota:** Las restantes tablas que aparecen en el modelo de datos presentan una estructura similar a la descrita para *title*, estas son empleadas para almacenar cada uno de los campos de Dublin Core: contributor, coverage, creator, date, description, format, identifier, language, publisher, relation, rights, source, subject, type. Para ver sus descripciones consultar el expediente de proyecto.

## 2.6 Patrones de arquitectura

Un patrón de arquitectura es una plantilla para una arquitectura de aplicaciones, especifica las propiedades generales a la estructura del sistema y repercute en la arquitectura de sus subsistemas.

La selección de un patrón de arquitectura es por lo tanto una decisión fundamental al desarrollar un sistema de *software* (44).

El sistema ArchiVenHIS se encuentra implementado sobre el framemork CodeIgniter, el cual se basa en el patrón arquitectónico MVC, por lo que el módulo a desarrollar debe ajustarse a este patrón.

MVC es un patrón de arquitectura de *software* encargado de separar la lógica de negocio de la interfaz del usuario, separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. Es el más utilizado en aplicaciones web, ya que facilita la funcionalidad, el mantenimiento y escalabilidad del sistema, de forma simple y sencilla (45)**.**

MVC divide las aplicaciones en tres niveles de abstracción:

- **Modelo:** representa la lógica de negocios. Es el encargado de acceder de forma directa a los datos actuando como "intermediario" con la base de datos.
- **Vista:** es la encargada de mostrar la información al usuario de forma gráfica y "humanamente legible".
- **Controlador:** es el intermediario entre la vista y el modelo. Es quien controla las interacciones del usuario solicitando los datos al modelo y entregándolos a la vista para que esta, los presente al usuario, de forma "humanamente legible".

## 2.7 Patrones de diseño

Un patrón de diseño define un esquema de refinamiento de los subsistemas o componentes dentro de un sistema, o las relaciones entre estos. Este describe una estructura común y recurrente de componentes interrelacionados, que resuelve un problema general de diseño dentro de un contexto particular (44).

Para la implementación del módulo se utilizaron los patrones de diseño GRASP, acrónimo de *General Responsibility Asignment Software Patterns* (patrones generales de *software* para asignar responsabilidades). A continuación se describen el uso de estos en el módulo.

- **Creador:** se utiliza en las clases controladoras al instanciar, a través del objeto load de clase loader, las vistas y las modelos.
- **Controlador:** consiste en asignar la responsabilidad de controlar el flujo de eventos del sistema. Este patrón se utiliza en las clases controladoras que son las que se encargan de obtener datos y enviarlos a las vistas.
- **Bajo acoplamiento y Alta Cohesión:** la propia implementación del *framework* utilizado contiene estos patrones nivelados puesto que permite el uso de los componentes de forma individual, evidenciando el bajo acoplamiento así como la dependencia entre ellos o alta

cohesión.

## Conclusiones del capítulo

En el presente capítulo teniendo en cuenta que el flujo de trabajo de análisis y diseño es uno de los más importantes que se lleva a cabo en el ciclo de vida de cualquier *software,* se modelaron algunos de sus artefactos, los cuales permitieron obtener una mayor comprensión del módulo, así como definir los principios que guiaron su implementación.

# Capítulo 3: Implementación y prueba del módulo para la interconexión y búsqueda en proveedores de datos

En el presente capítulo se modela el diagrama de componentes, haciendo una representación de la implementación de las clases del diseño en términos de componentes y cómo estos se organizan de acuerdo con los nodos específicos en el modelo de despliegue. Además, recoge el resultado de un conjunto de pruebas realizadas al *software* para verificar su funcionamiento y como es capaz de responder a los requerimientos para el cual fue concebido.

## 3.1 Diagrama de componentes

Un diagrama de componentes muestra cómo se organizan los elementos constituyentes del sistema y las dependencias existentes entre ellos. A continuación se muestra el diagrama de componentes de forma global:

**Diagrama de componentes general**

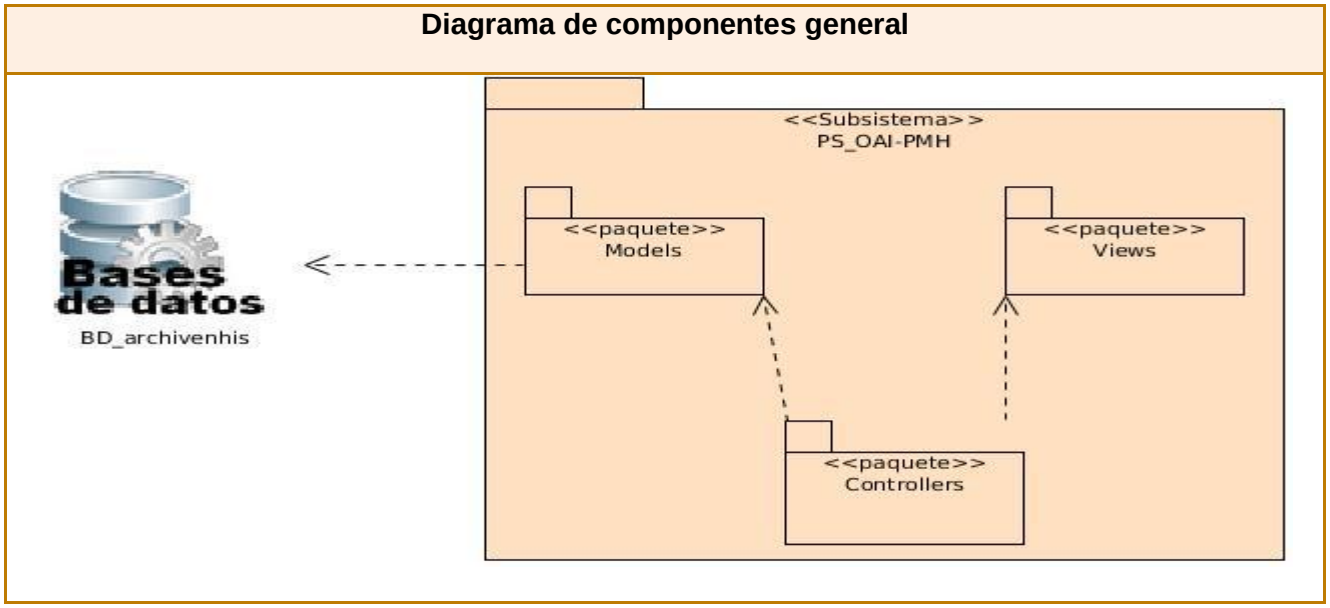



**Ilustración 22:** Diagrama de componente general

En el paquete Vistas se agrupan los componentes necesarios para la interacción del usuario con el sistema, los cuales son manejados por el paquete de Controladoras. A continuación se representan de forma detallada los componentes de dicho paquete.

**Diagrama de componente del paquete Vistas**

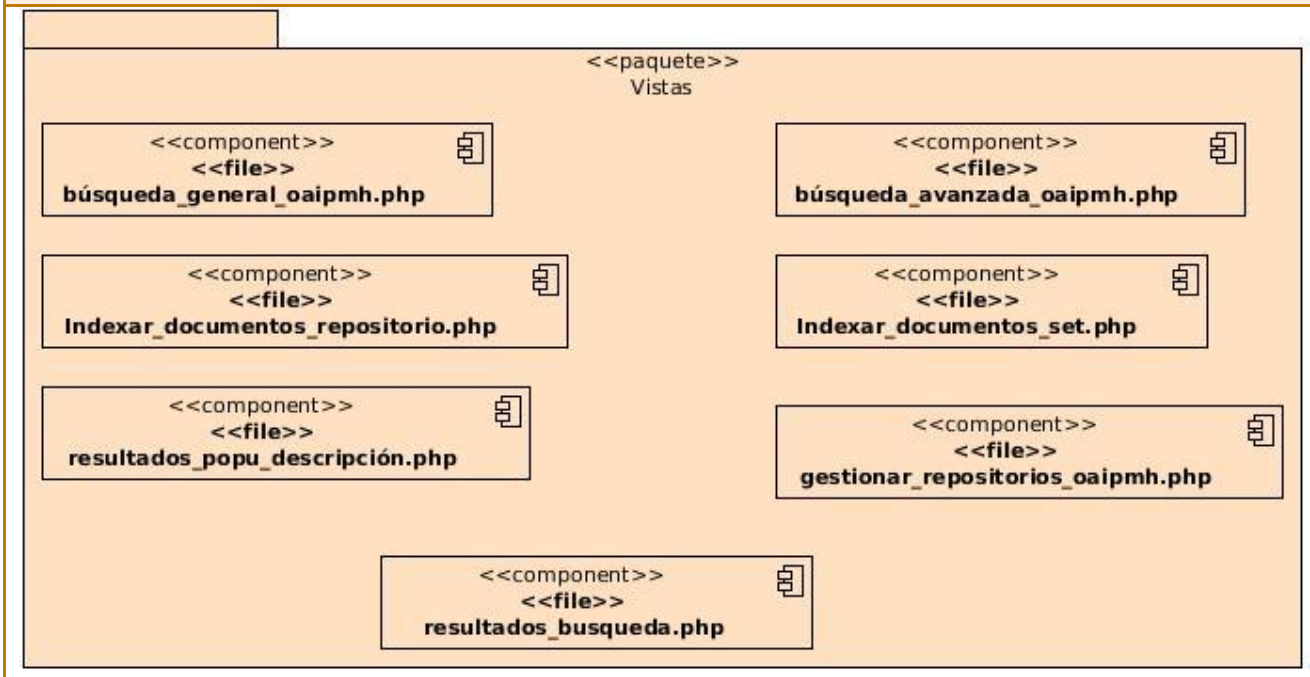



**Ilustración 23:** Diagrama de componente del paquete Vistas

El paquete Controladoras es el rector de las actividades de la aplicación, este contiene los ficheros de código fuente que interactúan con los demás paquetes, coordinando las acciones del *software*. A continuación se representan de forma detallada los componentes de dicho paquete.

**Diagrama de componente del paquete Controladoras**

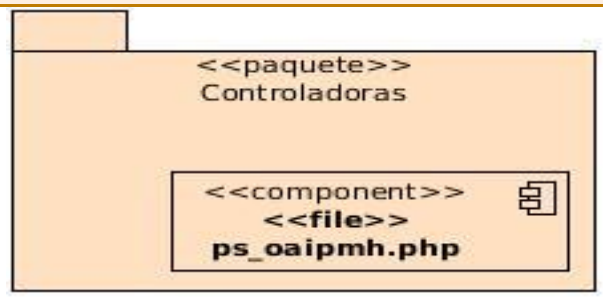



**Ilustración 24:** Diagrama de componente del paquete Controladoras

El paquete Modelos es el responsable de interactuar con la capa de almacenamiento de datos, para de esta forma gestionar la información con la que trabaja el sistema. A continuación se presentan de forma detallada los componentes de dicho paquete.

**Diagrama de componente del paquete Modelos**

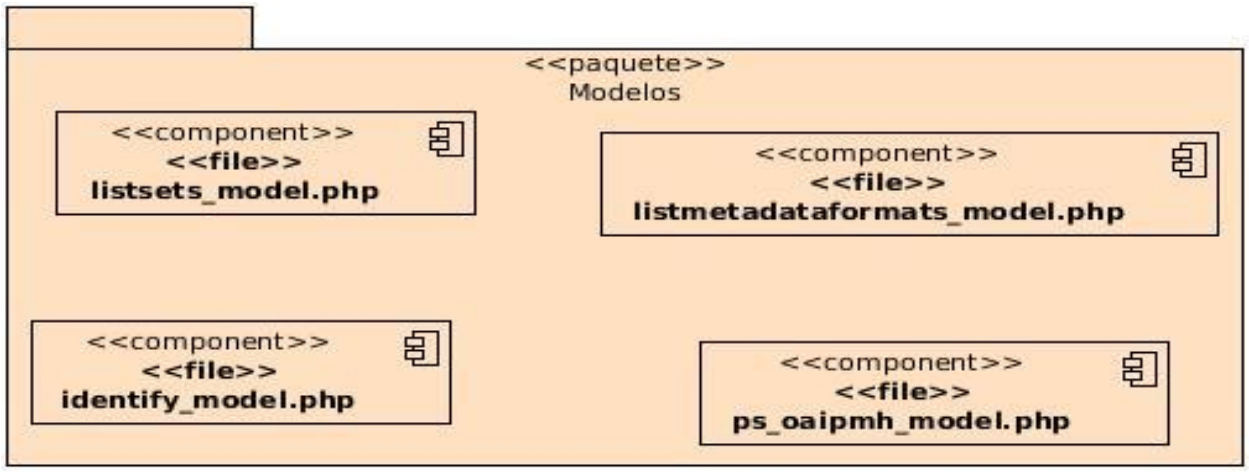



**Ilustración 25:** Diagrama de componente del paquete Modelos.

## 3.2 Diagrama de despliegue

Los diagramas de despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados mediante enlaces de comunicación (46). A continuación se describen los elementos que componen el diagrama de despliegue correspondiente al presente trabajo de diploma.

- Nodo PC Cliente: se refiere a las computadoras que utilizarán los usuarios para interactuar con la aplicación. Se comunica con el Servidor de Aplicación a través del protocolo HTTP.
- Nodo Servidor de Aplicación: representa el servidor Apache donde se encuentra instalado el sistema.
- Nodo Servidor de Base de Datos: es el servidor MySQL donde se almacena la base de datos del sistema.

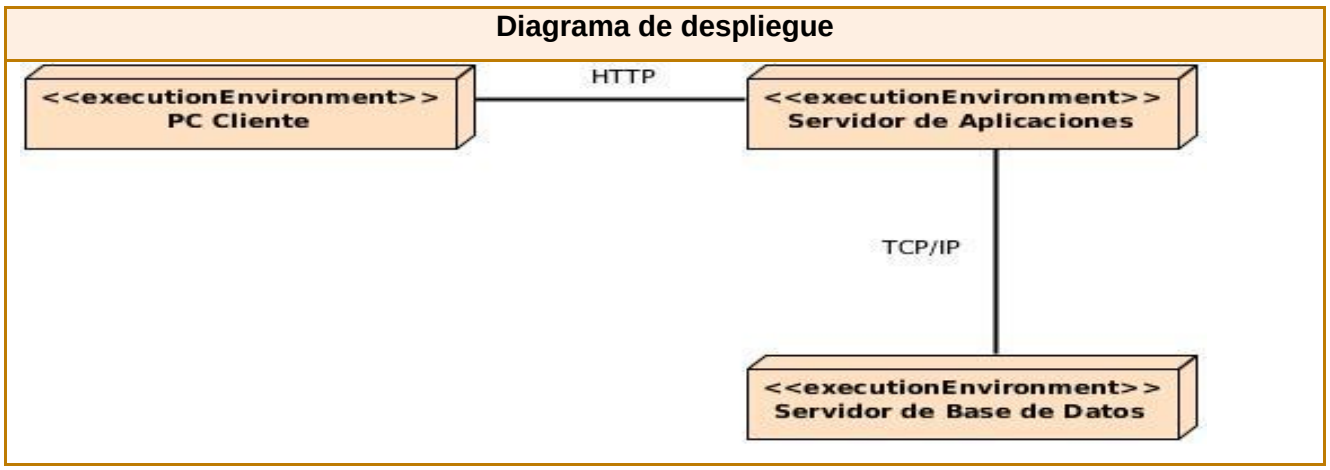



**Ilustración 21:** Diagrama de despliegue

## 3.3 Pruebas

Las pruebas de *software* son procesos que permiten verificar y revelar la calidad de un producto. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un programa de ordenador.

Las pruebas de caja negra, también denominadas prueba de comportamiento, se centran en los requisitos funcionales del *software*. Está prueba intenta encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos externas, errores de rendimiento y errores de inicialización y de terminación (23).

Existen diversos métodos de prueba de caja negra, pero uno de los más usados es la partición equivalente, este método divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar (23).

A continuación se muestran los casos de prueba de los casos de usos críticos, el resto pueden ser consultados en el expediente de proyecto.

| **Caso de prueba:** Gestionar repositorio OAI-PMH | | |
|---|---|---|
| **Gestionar repositorio OAI-PMH (**Adicionar repositorio OAI-PMH**)** | | |
| **Entrada** | **Resultados** | **Condiciones** |
| El usuario introduce todos los datos válidos | El sistema notifica al actor "El repositorio ha sido adicionado satisfactoriamente". | El actor debe estar autenticado en el sistema. |
| El usuario introduce el nombre incorrecto y todos los demás datos válidos | El sistema notifica al actor "El nombre no es válido". | |
| El usuario introduce el nombre incorrecto (vacío) y todos los demás datos válidos | El sistema notifica al actor "El nombre no puede estar vacío" (Javascript) y con el mensaje "Los campos obligatorios no pueden estar vacíos". | |
| El usuario introduce la dirección url incorrecto (vacío) y todos los demás datos válidos | El sistema notifica al actor "La url no puede estar vacía" (Javascript) y con el mensaje "Los campos obligatorios no pueden estar vacíos". | |
| El usuario introduce la dirección url incorrecto y todos los demás datos válidos | El sistema notifica "Por favor, introduzca una URL". | |
| El usuario introduce el correo incorrecto y todos los demás datos válidos | El sistema notifica "Por favor, introduzca una dirección de correo". | |
| El usuario introduce un nombre que ya existe y todos los demás datos válidos | El sistema notifica "Existe un repositorio con el nombre especificado". | |

| | | |
|---|---|---|
| El usuario introduce una dirección url que ya existe y todos los demás datos válidos | El sistema notifica “Existe un repositorio con la url especificada”. | |
| **Gestionar repositorio OAI-PMH (**Modificar repositorio OAI-PMH**)** | | |
| **Entrada** | **Resultados** | **Condiciones** |
| El usuario introduce todos los datos válidos | El sistema notifica al actor “El repositorio ha sido modificado satisfactoriamente”. | El actor debe estar autenticado en el sistema. |
| El usuario introduce el nombre incorrecto y todos los demás datos válidos | El sistema notifica al actor “El nombre no es válido”. | |
| El usuario introduce el nombre incorrecto (vacío) y todos los demás datos válidos | El sistema notifica al actor “El nombre no puede estar vacío” (Javascript) y con el mensaje “Los campos obligatorios no pueden estar vacíos”. | |
| El usuario introduce la dirección url incorrecto (vacío) y todos los demás datos válidos | El sistema notifica al actor “La url no puede estar vacía” (Javascript) y con el mensaje “Los campos obligatorios no pueden estar vacíos”. | |
| El usuario introduce la dirección url incorrecto y todos los demás datos válidos | El sistema notifica “Por favor, introduzca una URL”. | |
| El usuario introduce el correo incorrecto y todos los demás datos válidos | El sistema notifica “Por favor, introduzca una dirección de correo”. | |

| | | |
|---|---|---|
| El usuario introduce un nombre que ya existe y todos los demás datos válidos | El sistema notifica “Existe un repositorio con el nombre especificado”. | |
| El usuario introduce una dirección url que ya existe y todos los demás datos válidos | El sistema notifica “Existe un repositorio con la url especificada”. | |
| El usuario introduce el nombre incorrecto y todos los demás datos válidos | El sistema notifica al actor “El nombre no es válido”. | |
| **Gestionar repositorio OAI-PMH (**Eliminar repositorio OAI-PMH**)** | | |
| **Entrada** | **Resultados** | **Condiciones** |
| El usuario selecciona el vínculo eliminar del repositorio | El sistema elimina el repositorio, actualiza el listado de repositorios sin la presencia de la colección eliminada y muestra un mensaje de confirmación “El repositorio ha sido eliminado satisfactoriamente”. | El actor debe estar autenticado en el sistema. |
| | Si el repositorio seleccionado esta activado, el sistema muestra un mensaje de error “El repositorio seleccionado no puede ser eliminado”. | |
| El usuario no selecciona ningún repositorio y oprime el botón Eliminar | El sistema muestra un mensaje de error “Debe seleccionar al menos un repositorio para eliminar”. | |

| | | |
|---|---|---|
| El usuario selecciona varios repositorios y oprime el botón Eliminar | Si de los repositorios seleccionados, existe al algún repositorio activado el sistema elimina sólo los repositorios que no están activados y muestra un mensaje “Existe: 1 repositorios que no puede ser eliminado”. | |
| | Si todos los repositorios seleccionados están activados, el sistema muestra un mensaje “Existían: 2 repositorios que no pueden ser eliminados”. | |
| | El sistema elimina los repositorios, actualiza el listado de repositorios y muestra un mensaje de confirmación “Los repositorios ha sido eliminado satisfactoriamente”. | |

**Tabla 15:** Diseño de caso de prueba del CU Gestionar repositorio OAI-PMH

| **Caso de prueba:** Modificar estado | | |
|---|---|---|
| **Modificar estado (**Activar**)** | | |
| **Entrada** | **Resultados** | **Condiciones** |
| Se escoge la opción activar | El sistema verifica que la url este activada, en caso positivo activa el repositorio y muestra un mensaje “El repositorio ha sido activado satisfactoriamente”. | El actor debe estar autenticado en el sistema. |

| | | |
|---|---|---|
| Se escoge la opción activar | El sistema verifica que la url este activada, en caso negativo muestra un mensaje “La url no está activa y el directorio no puede ser activado”. | |
| **Modificar estado (**Desactivar**)** | | |
| **Entrada** | **Resultados** | **Condiciones** |
| Se escoge la opción desactivar | El sistema desactiva el repositorio y muestra un mensaje ”El repositorio ha sido desactivado satisfactoriamente” | El actor debe estar autenticado en el sistema. |

**Tabla 16:** Diseño de caso de prueba del CU Modificar estado

| **Caso de prueba:** Indexar repositorio | | |
|---|---|---|
| **Indexar repositorio** | | |
| **Entrada** | **Resultados** | **Condiciones** |
| El usuario selecciona el vínculo indexar del repositorio | El sistema notifica al actor “Los documentos se han indexado correctamente”. | El actor debe estar autenticado en el sistema. |
| | El sistema notifica al actor “No se indexó ningún documento”. | |
| El usuario no selecciona ningún repositorio y oprime el botón Indexar | El sistema notifica al actor “Debe seleccionar al menos un repositorio”. | |
| El usuario selecciona varios repositorios y oprime el botón Eliminar | El sistema notifica al actor un mensaje por cada uno de los repositorios que el seleccionó. | |

**Tabla 17:** Diseño de caso de prueba del CU Indexar repositorio

| **Caso de prueba:** Indexar set | | |
|---|---|---|
| **Indexar set** | | |
| **Entrada** | **Resultados** | **Condiciones** |
| El usuario no selecciona ningún repositorio | El sistema notifica al actor “Debe seleccionar un repositorio”. | El actor debe estar autenticado en el sistema. |
| El usuario selecciona un repositorio y oprime el botón Indexar | El sistema notifica al actor “Debe seleccionar al menos una agrupación”. | |
| El usuario selecciona un repositorio y selecciona las agrupaciones que desea indexar | El sistema notifica al actor un mensaje por cada uno de los repositorios que el seleccionó. | |

**Tabla 18:** Diseño de caso de prueba del CU Indexar set

## Conclusiones del capítulo

En el presente capítulo se abordaron los aspectos fundamentales de los flujos de trabajo de implementación y prueba, donde se realizó la modelación del diagrama de despliegue, especificando la distribución física por nodos del sistema; además de la modelación del diagrama de componentes, obteniéndose una visión mucho más clara sobre la estructura del módulo. Por su parte en el flujo de trabajo prueba se le realizaron un conjunto de casos de prueba a la aplicación detectando varias no conformidades, las cuales fueron solucionadas para lograr un mejor funcionamiento del sistema.

.

# Conclusiones

Una vez finalizada la investigación se puede arribar a las siguientes conclusiones:

- El protocolo para el intercambio y recuperación de información OAI-PMH es el más idóneo para el Sistema de Gestión de Archivos Históricos ArchiVenHIS.
- La implementación del proveedor de servicios para el Sistema de Gestión de Documentos Históricos ArchiVenHIS le aporta un importante valor agregado, al permitirle difundir la memoria histórica salvaguardada por otras instituciones, que emplean para gestionarla, sistemas que implementan el proveedor de datos según el protocolo OAI-PMH.
- El módulo para la interconexión y búsquedas en proveedores de datos implementado presenta un correcto funcionamiento de acuerdo a los resultados obtenidos a partir de las pruebas realizadas.

# Recomendaciones

Para el desarrollo de futuras investigaciones y proyectos que guarden relación con este trabajo se proponen las siguientes recomendaciones:

- Automatizar el proceso de indexar documentos.
- Recuperar metadatos en otros formatos.
- Internacionalizar el *software*.

# Bibliografía referenciada


1. Gaceta oficial de la repubica de Cuba Ministerio de Justicia. [En línea] 5 de MAYO de 2009. [Disponible en: http://www.gacetaoficial.cu/. ISSN 1682-7511].
2. **Mena Mugica, Mayra.** Gestión documental y organización de archivos. La Habana : Félix Varela, 2005. ISBN 959-258-950-X.
3. Diccionario de Terminología archivística. [En línea] [Citado el: 28 de novienbre de 2012.] [Disponible en: http://www.mcu.es/archivos/MC/DTA/Diccionario.html].
4. **HEREDIA HERRERA, ANTONIA.** ARCHIVíSTICA GENERAL.TEORíA Y PRACTICA. SEVILLA: s.n., 1991. I.S.B.N. 84 - 7798 - 056 – X.
5. Descripción en archivos. [En línea] [Citado el: 3 de abril de 2012.] [Disponible en: http://www.vicentgimenez.net/curs/descrip.htm].
6. Consejo Internacional de Archivos. Normal Internacional General de Descripción Archivística ISAD (G). Madrid : s.n., 2000.
7. **Carpenter, Leona.** OAI para principiantes: Introducción. [En línea] [Citado el: 10 de diciembre de 2012.] [Disponible en: http://travesia.mcu.es/portalnb/jspui/html/10421/1823/page1.htm].
8. **Gómez, Laureano Felipe.** Interoperabilidad en los Sistemas de Información Documental (SID): la información debe fluir. 2007.
9. **Carpenter, Leona.** Glosario. [En línea] [Citado el: 10 de diciembre de 2012.] [Disponible en: http://travesia.mcu.es/portalnb/jspui/html/10421/1823/page6.htm#section16].
10. **García, Nelida Elba y Caballero, Sergio.** Metadatos : necesidad e importancia de integrar estándares. [Citado el: 20de febrero de 2012.]
11. Dienst. [En línea] [Citado el: 17 de abril de 2012.] [Disponible en: http://www.sedic.es/autoformacion/acceso_abierto/4-definicion5.html].
12. **Krichel, Thomas.** Guildford Protocol. [En línea] [Citado el: 17 de abril de 2012.] [Disponible en: http://openlib.org/acmes/root/docu/guilp_1999-12-27.html].
13. Simple Digital Library Interoperability Protocol (SDLIP-Core). [En línea] [Citado el: 17 de abril de 2012.] [Disponible en: http://mallikarjunahb.blogspot.com/2006/10/simple-digital-library-interoperability.html].
14. **Guajardo Salinas, Aldo.** Z39.50 y OAI-PMH: Protocolos de Transferencia y Recuperación de Información. 2010.

15. **Barrueco, José Manuel y Subirats Coll, Imma.** Open archives initiative. Protocol for metadata harvesting (OAI-PMH): descripción, funciones y aplicaciones de un protocolo. abril de 2003.
16. **Carpenter, Leona.** Historia y desarrollo del OAI-PMH. [En línea] 14 de Octubre de 2003. [Citado el: 22 de febrero de 2012.] [Disponible en: http://travesia.mcu.es/portalnb/jspui/html/10421/1823/page2.htm].
17. **SIZA RAMÍREZ, JUAN PABLO.** Estudio, análisis, evaluación e implementación de un protocolo de intercambio de contenidos sobre internet para la interoperabilidad de repositorios de la Biblioteca Digital en la Universidad Nacional de Colombia con un proveedor de servicios de contenido. 2010.
18. **PEMAU ALONSO, JULIO y BARROSO CORROTO, JOSE ANGEL.** INTRODUCCIÓN A OAI-PMH Y SU IMPLANTACIÓN EN EL PORTAL E-REVISTAS.
19. Dublin Core. [En línea] 2010. [Citado el: 20 de mayo de 2012.] [Disponible en: http://www.metadatos-xmlrdf.com/metadatos/dublin-core].
20. **Ayllón Bonet, Julio César.** Recuperación y acceso a la Información. Metadatos y Documentos XML/RDF para Recuperación. 2007.
21. Archivo 3000. [En línea] [Citado el: 20 de octubre de 2012.] [Disponible en: http://archivo3000.com/general/archivo3000.htm#pestanas].
22. Portal de Archivos Españoles. [En línea] [Citado el: 4 de febrero de 2012.] [Disponible en: http://pares.mcu.es/].
23. **Pressman, Roger S.** Ingeniería de *Software*. Un enfoque práctico. Madrid y Carchelejo (España): s.n., 2001.
24. **Mendoza Sanchez, María A.** Metodologías De Desarrollo De *Software*. 2004.
25. Ventajas y desventajas del Personal Home Page PHP. [En línea] [Citado el: 20 de enero de 2012.] [Disponible en: http://www.creargratisunapaginaweb.com/PHP/Ventajas-y-desventajas-del-Personal-Home-Page-4/].
26. Los diferentes lenguajes de programación para la web. [En línea] [Citado el: 20 de octubre de 2012.] [Disponible en: http://www.maestrosdelweb.com/principiantes/los-diferentes-lenguajes-de-programacion-para-la-web/].
27. **Eguíluz Pérez, Javier.** Introducción a CSS. [Disponible en: http://sunshine.prod.uci.cu/gridfs/sunshine/books/introduccion_css.pdf].
28. 10 razones para usar CSS. [En línea] [Citado el: 26 de enero de 2012.] [Disponible en: http://www.maestrosdelweb.com/editorial/usarcss/].

29. **Eguíluz Pérez, Javier.** Introducción a JavaScript. [Disponible en: http://sunshine.prod.uci.cu/gridfs/sunshine/books/introduccion_javascript_2caras.pdf].
30. **Eguíluz Pérez, Javier.** Introducción a AJAX. [Disponible en: http://sunshine.prod.uci.cu/gridfs/sunshine/books/introduccion_ajax.pdf].
31. 10 Razones para usar Ajax. [En línea] [Citado el: 26 de enero de 2012.] [Disponible en: http://www.tufuncion.com/ventajas-ajax].
32. Extensible Markup Language (XML). [En línea] [Citado el: 20 de mayo de 2012.] [Disponible en: http://www.w3.org/XML/].
33. **Cruz Toirac, Damaris y García Martín, Jorge.** MULTIMEDIA CURSO XML. Ciudad de la Habana: s.n., 2007. [Disponible en: http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_0728_07/1/TD_0728_07.pdf].
34. **Herranz, Raúl.** Lenguajes de Modelado. [En línea] 2010. [Disponible en: http://www.utopicainformatica.com/2010/12/lenguajes-de-modelado.html].
35. **Gilfillan, Ian.** La Biblia de MySQL. [Disponible en: http://sunshine.prod.uci.cu/gridfs/sunshine/books/La_Biblia_De_Mysql.pdf].
36. Definición de apache. [En línea] [Citado el: 26 de enero de 2012.] [Disponible en: http://www.definicion.org/apache].
37. Bienvenido a NetBeans y a www.netbeans.org, Portal del IDE Java Open Source [En línea] [Citado el: 25 de noviembre de 2012.] [Disponible en: http://netbeans.org/community/articles/welcome-template_es.html].
38. Visual Paradigm para UML. [En línea] [Citado el: 22 de noviembre de 2012.] [Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/].
39. **Murphey, Rebecca.** jQuery Fundamentals. [En línea] [Citado el: 25 de noviembre de 2012.] [Disponible en: http://www.etnassoft.com/biblioteca/jquery-fundamentals].
40. JQuery. [En línea] [Citado el: 25 de enero de 2012.] [Disponible en: http://iie.fing.edu.uy/ense/asign/tap/obrar09/GastonDeLeon_comp/web/partes/jquery/descripcion/home.html].
41. **Alvarez, Miguel Angel.** Manual de CodeIgniter. [Disponible en: http://sunshine.prod.uci.cu/gridfs/sunshine/books/manual-codeigniter.pdf].
42. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** El Proceso Unificado de Desarrollo de *Software*. Madrid: s.n., 2000. ISBN: 84-7829-036-2. [Disponible en:

http://sunshine.prod.uci.cu/gridfs/sunshine/books/El_Proceso_Unificado_de_Desarrollo_de_Software.pdf].

43. **Martínez, Alejandro y Martínez, Raúl.** Guía a Rational Unified Process.
44. **Parra, Jose David.** Guía de Patrones, Prácticas y Arquitectura .NET. [Disponible en: http://sunshine.prod.uci.cu/gridfs/sunshine/books/PPArquitecturaNET.pdf].
45. **Bahit, Eugenia.** El paradigma de la Programación Orientada a Objetos en PHP y el patrón de arquitectura de *Software* MVC. [Disponible en: http://sunshine.prod.uci.cu/gridfs/sunshine/books/MVC_con_PHP.pdf].
46. **Quisbert Limachi, Nancy Susana y Marca Huallpara, Hugo Michael.** Diagrama de Despliegue. [En línea] [Citado el: 20 de mayo de 2012.] [Disponible en: http://virtual.usalesiana.edu.bo/web/practica/archiv/despliegue.doc].

# Bibliografía


1. Diccionario de Terminología archivística. [En línea] [Citado el: 28 de novienbre de 2012.] [Disponible en: http://www.mcu.es/archivos/MC/DTA/Diccionario.html].
2. **HEREDIA HERRERA, ANTONIA.** ARCHIVíSTICA GENERAL.TEORíA Y PRACTICA. SEVILLA : s.n., 1991. I.S.B.N. 84 - 7798 - 056 - X.
3. **Cruz Mundet, José Ramón.** Manual de Archivística. 2da edición. Madrid : s.n., 1996. ISBN 8.1-86168-94-5 (F.G.S.R.), ISBN 83-368-0860-6 (Pirámide).
4. **Mena Mugica, Mayra.** Gestión documental y organización de archivos. La Habana : Félix Varela, 2005. ISBN 959-258-950-X.
5. Definición de Protocolo (en informática) - Concepto y Significado. [En línea] [Citado el: 04 de marzo de 2012.] http://www.carlospes.com/minidiccionario/protocolo.php.
6. **Gómez, Laureano Felipe.** Interoperabilidad en los Sistemas de Información Documental (SID): la información debe fluir. 2007.
7. **Sené, María Luisa.** LOS METADATOS Y SU LUGAR EN LA ARENA INTERNACIONAL. Ciudad de La Habana : s.n.
8. Dienst Protocol. [En línea] [Citado el: 17 de abril de 2012.] [Disponible en: http://www.cs.cornell.edu/cdlrg/dienst/protocols/dienstprotocol.htm].
9. **Sanz Domingo, Pedro y Martín, Pepi.** @absysnet.com. Z39.50 . [En línea] 2001-2005. [Disponible en: http://www.absysnet.com/tema/tema1.html].
10. Open Archives Initiative. [En línea] [Citado el: 25 de octubre de 2012.] [Disponible en: http://www.openarchives.org/].
11. **Guajardo Salinas, Aldo.** Z39.50 y OAI-PMH: Protocolos de Transferencia y Recuperación de Información. 2010.
12. Z39.50 Norma para la recuperación de información. [En línea] [Citado el: 25 de noviembre de 2012.] [Disponible en: http://caribe.udea.edu.co/~hlopera/Z3950.html].
13. Z39.50 - Wikipedia, la enciclopedia libre. [En línea] [Citado el: 25 de noviembre de 2012.] [Disponible en: http://es.wikipedia.org/wiki/Z39.50].
14. **Pérez Velandia, Mayerly y Silva, Luís Felipe.** COMO FUNCIONA EL PROTOCOLO OAI – PMH EN LA RECUPERACION DE INFORMACION.
15. **Carpenter, Leona.** Historia y desarrollo del OAI-PMH. [En línea] 2003. [Citado el: 22 de febrero de 2012.] [Disponible en: http://travesia.mcu.es/portalnb/jspui/html/10421/1823/page2.htm]

16. **SIZA RAMÍREZ, JUAN PABLO.** Estudio, análisis, evaluación e implementación de un protocolo de intercambio de contenidos sobre internet para la interoperabilidad de repositorios de la Biblioteca Digital en la Universidad Nacional de Colombia con un proveedor de servicios de contenido. 2010.
17. **PEMAU ALONSO, JULIO y BARROSO CORROTO, JOSE ANGEL.** INTRODUCCIÓN A OAI-PMH Y SU IMPLANTACIÓN EN EL PORTAL E-REVISTAS.
18. **Ayllón Bonet, Julio César.** Recuperación y acceso a la Información. Metadatos y Documentos XML/RDF para Recuperación. 2007.
19. **Sené, María Luisa.** LOS METADATOS Y SU LUGAR EN LA ARENA INTERNACIONAL. Ciudad de La Habana : s.n.
20. **Van de Sompel, Herbert y Lagoze, Carl.** The Santa Fe Convention of the Open Archives Initiative. [En línea] 2000. [Citado el: 20 de enero de 2012.] [Disponible en: http://www.dlib.org/dlib/february00/vandesompel-oai/02vandesompel-oai.html].
21. **Carpenter, Leona.** Implementando OAI-PMH. [En línea] [Disponible en: http://travesia.mcu.es/portalnb/jspui/html/10421/1823/page4.htm].
22. The Dublin Core® Metadata Initiative. [En línea] [Citado el: 14 de diciembre de 2012.] [Disponible en: http://www.dublincore.org/].
23. **ROJAS YEPES, SERBE LEÓN.** APLICACIÓN DEL ESQUEMA DE METADATOS DUBLIN CORE EN LA BIBLIOTECA DIGITAL DE LA UNIVERSIDAD DE ANTIOQUIA. 2010.
24. PHP - Wikipedia, la enciclopedia libre. [En línea] [Citado el: 14 de diciembre de 2012.] [Disponible en: http://es.wikipedia.org/wiki/PHP].
25. Ventajas de usar PHP. [En línea] [Citado el: 30 de noviembre de 2012.] [Disponible en: http://codigoprogramacion.com/programacionweb/50-ventajas-de-usar-php.html].
26. ¿Qué es Javascript? [En línea] [Citado el: 31 de octubre de 2012.] [Disponible en: http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/].
27. **Archivo 3000.** [En línea] [Citado el: 30 de Octubre de 2011.] [Disponible en: http://www.archivo3000.com/]
28. **Visual Paradigm for UML**. [En línea] 5 de Marzo de 2007. [Citado el: 25 de noviembre de 2011.] [Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(MÍ)_14720_p/].

# Glosario de términos

**A**

**API (**por sus siglas en inglés, **Application Program Interface):** conjunto de convenciones internacionales que definen cómo debe invocarse una determinada función de un programa desde una aplicación. Cuando se intenta estandarizar una plataforma, se estipulan unos APIs comunes a los que deben ajustarse todos los desarrolladores de aplicaciones.

Herramientas de programación para rutinas, protocolos y *software*.

**Arquitectura:** es una descripción de la organización, motivación y estructura de un sistema. Están implicados muchos niveles diferentes de arquitecturas en el desarrollo de los sistemas *software*, desde la arquitectura hardware física a la arquitectura lógica de un *b* de aplicación

**C**

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

**E**

**Estándar:** es un patrón, una tipificación o una norma de cómo realizar algo (AulaGlobal, 2007).

**F**

***Framework*:** es una herramienta de automatización basados en patrones para resolver los problemas más comunes en el desarrollo de aplicaciones. Su función es facilitar el proceso de concepción del *software* gracias a que es capaz de encapsular un conjunto de funcionalidades que pueden ser complejas en funciones sencillas, disminuyendo considerablemente el tiempo de desarrollo.

**G**

**GET:** es un método empleado para enviar datos desde el navegador al servidor Web, generalmente utilizado al enviar formularios y especificado mediante la directiva METHOD. Este método envía los parámetros por medio de la URL del fichero que carguemos. El método GET es bueno para recuperar información de los sitios favoritos o en la optimización del motor de búsqueda y la indexación.

**H**

**Herramienta CASE (**por sus siglas en inglés, ***Computer Aided Software Engineering*)**:** son un conjunto de programas destinados a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

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

**I**

**ISO (**por sus siglas en inglés, **International Organization for Standarization):** es una organización de carácter voluntario fundada en 1946 que es responsable de la creación de estándares internacionales en muchas áreas, incluyendo la informática y las comunicaciones. Está formada por las organizaciones de normalización de sus países miembro.

**M**

**MARC:** es un conjunto de estándares para manipular datos bibliográficos legibles por ordenador.

**Metodología:** es un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de un producto *software*.

**Modelo:** captura una vista de un sistema del mundo real. Es una abstracción de dicho sistema, considerando un cierto propósito. Así, el modelo describe completamente aquellos aspectos del sistema que son relevantes al propósito del modelo, y a un apropiado nivel de detalle.

**MoReq:** Modelo de Requisitos para la Gestión Electrónica de Documentos de Archivo

**O**

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

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

**Objetos:** En este contexto hace referencia a un conjunto de recursos digitales de cualquier tipo de formato.

**OMG (**por sus siglas en inglés, **Object Management Group)**: organización que promueve estándares para la industria.

**P**

**Plataforma:** sistema operativo o sistemas complejos, ya sea de hardware o *software*, sobre el cual un programa pueda ejecutarse.

**POST:** es un método empleado para enviar datos desde el navegador al servidor web, generalmente utilizado al enviar formularios y especificado mediante la directiva METHOD. Este método envía los datos sin que el usuario los vea, lo que lo hace más seguro, por ejemplo al enviar

información de un formulario. POST es bueno en los formularios donde se envían datos una sola vez.

**PostgreSQL:** es un servidor de bases de datos relacional orientado a objetos. Está dirigido por una comunidad de desarrolladores y organizaciones comerciales que se hacen llamar PGDG (por sus siglas en inglés, PostgreSQL Global Development Group).

**R**

**Reléase:** se refiere a una versión funcional de un producto *software***.**

**Repositorio:** en el contexto que es usado se refiere a un "proveedor de datos" y de forma genérica se puede definir como un lugar central donde se registran datos para su almacenamiento y conservación con propósitos diversos de seguridad o consulta posterior.

**S**

**SET:** en este contexto hace referencia a la agrupación de objetos según alguna característica de los mismos.

**Servidor web:** un servidor web es una máquina conectada a la red en la que están almacenadas físicamente las páginas que componen un sitio web. Dícese también del programa que sirve dichas páginas.

**Sistema de Gestión de Base de Datos (SGBD):** conjunto coordinado de programas, procedimientos, lenguajes, herramientas, etc., que suministra los medios necesarios para describir, actualizar y administrar los datos de una BD.

***Software*:** es el conjunto de programas de cómputo, documentos asociados y esquemas de configuración necesarios para que estos programas operen.

**U**

**URI :** Acrónimo de Uniform Resource Identifier (identificador uniforme de recurso).

**URL (**por sus siglas en inglés, **Uniform Resource Identifier):** es el identificador de cualquier tipo de objeto que pueda llegar a ser catalogado.

# Anexos

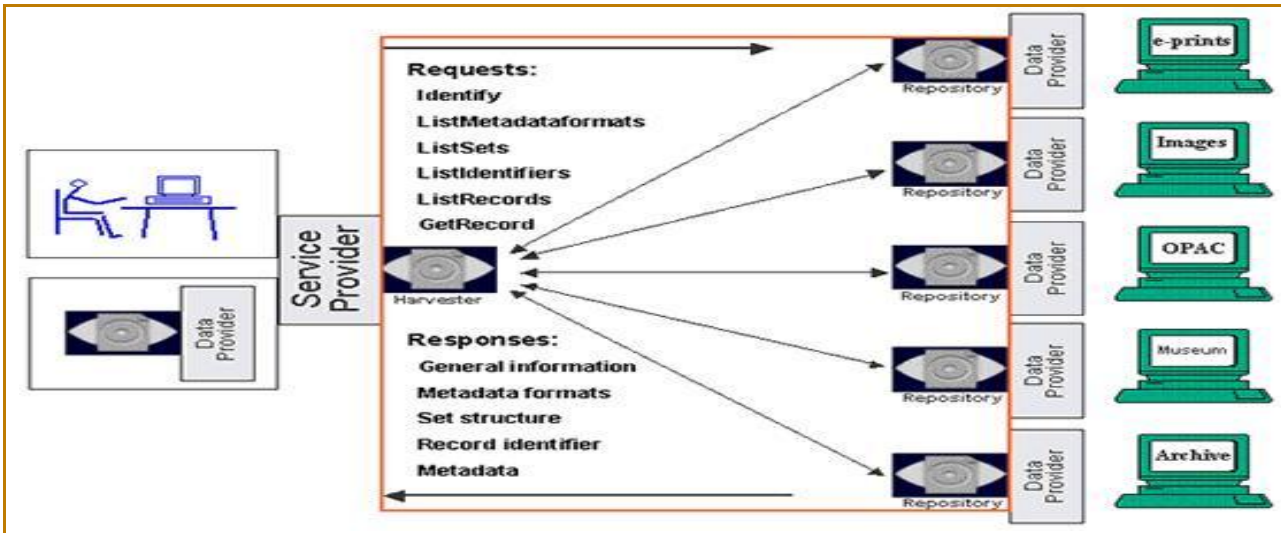



**Ilustración 1:** Describe la estructura y el flujo de trabajo.

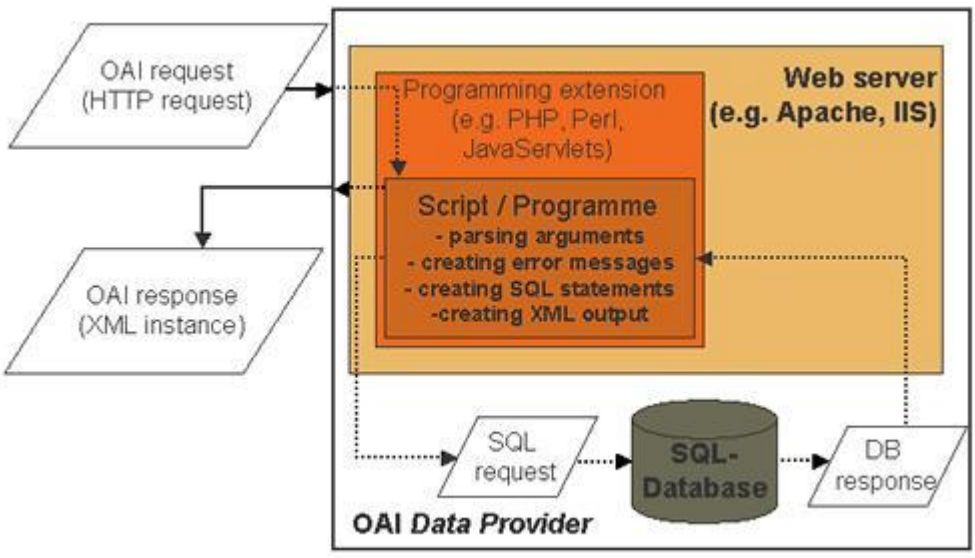



**Ilustración 2:** Iteración de los componentes del proveedor de datos

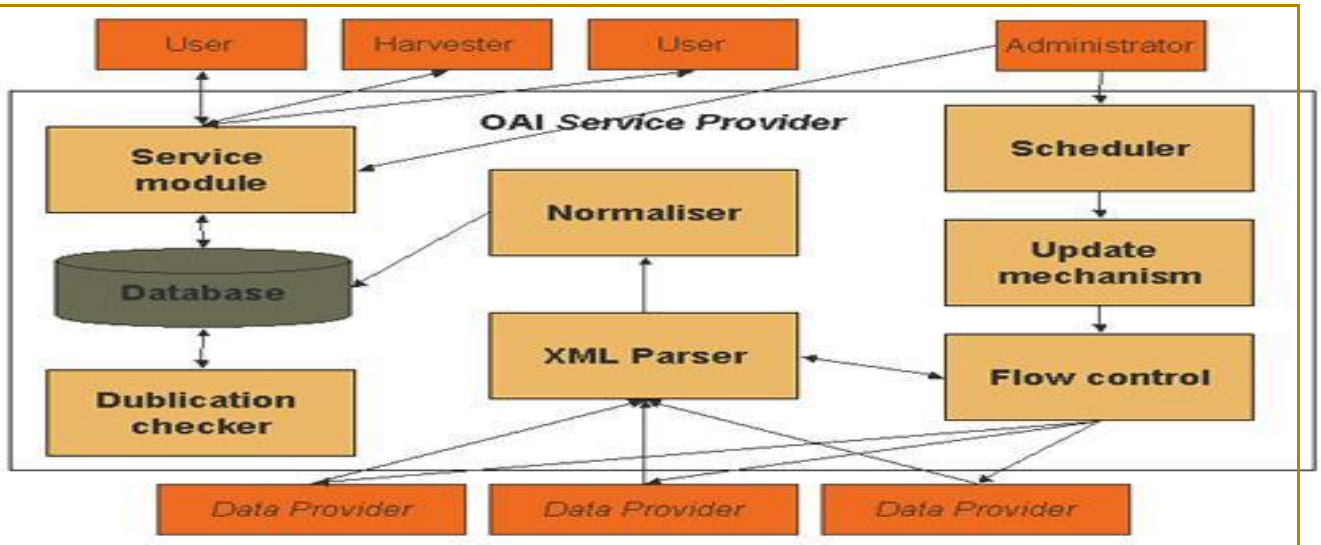



**Ilustración 3:** Iteración de los componentes del proveedor de servicio

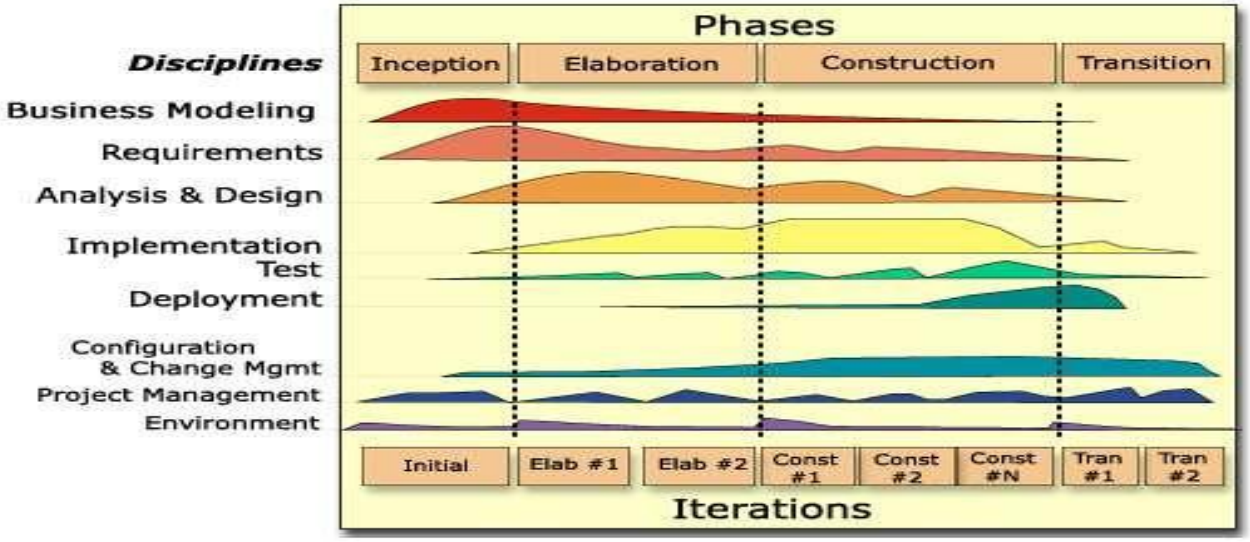



**Ilustración 4:** Etapas e iteraciones de RUP.