Universidad de las Ciencias Informáticas

Facultad 2

Trabajo de Diploma para optar por el Título Ingeniero en Ciencias Informáticas

Título:

"Proveedor de metadatos para el Sistema ABCD"

Autores:
Dariel Elejalde Véliz
Yordan Herrera Venet

Tutores:
MSc. Madelis Pérez Gil
Ing. Luis Carlos Álvarez Fernández

La Habana, junio de 2017
"Año 59 de la Revolución"

“Un poco más de persistencia, un poco más de esfuerzo, y lo que parecía irremediablemente un fracaso puede convertirse en un éxito glorioso”

Elbert Hubbard

Declaración de autoría

Se declara que Dariel Elejalde Véliz y Yordan Herrera Venet son los únicos autores del trabajo de diploma titulado "**Proveedor de metadatos a través del protocolo OAI-PMH para el Sistema ABCD**" y se autoriza a la Facultad 2 de la Universidad de las Ciencias Informáticas de hacer su uso para su beneficio.
Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Tutores:

Ing. Luis Carlos Álvarez Fernández

_____
Firma del Tutor

MSc. Madelis Pérez Gil

_____
Firma del Tutor

Autores:

Dariel Elejalde Véliz

_____
Firma del Autor

.Yordan Herrera Venet

_____
Firma del Autor

Quiero dedicarle este resultado con mucho amor y cariño a mi bisabuela Victoria, quien deseaba estar presente en este momento tan crucial de mi vida. Para ella, donde quiera que se encuentre, le doy mil gracias por todo lo que aportó a mi personalidad. Un beso para ti: mi Bisabuela.

Para mi mamá las palabras no me alcanzarán nunca, eres mi guía, mi luz, mi razón de ser. Siempre a mi lado en los momentos tristes y felices de mi vida; tuve la oportunidad de ser tu alumno y la verdad que como tú, mi vieja, ninguna.

Papá de siempre, papá de todos los tiempos, de ti obtengo los mejores consejos y sugerencias. Yo quiero que sepas que eres un ídolo para mí y siempre seguiré tu ejemplo de padre, amigo y de hijo; te quiero papá.

Abuela, nunca te equivocas cuando observas un poco más lejos que nosotros. Estás llena de experiencia y sabiduría, claro, la obtuviste de mi bisabuela. Tus consejos para mí, siempre serán escuchados y de ti aprendo todos los días.

Mi hermanito, mi brazo derecho, tú sabes que somos los más jóvenes del familión. Me alegro mucho de tu graduación y quiero que te esfuerces cada día por ser mejor. Tú y yo, tenemos la tarea de defender lo que mamá, abuela y papá nos han enseñado.

Mi tía Clara, no me olvidé de ti, tan risueña como siempre, alegras la casa cada vez que vas, sabes que puedes contar con la familia para lo que sea.

También dedicarles con todo mi amor a mi abuelo por parte de madre Eusebio y a mi abuela por parte de padre Wilma.

Un abrazo fuerte para mi tía Ana Julia y tío Israel, sé que son otro eslabón familiar que me vio nacer. Les debo muchísimo, y saben cuánto los quiero. Sus hijos Yanet y Arnaldo son unos primos que jamás pensé tenerlos. Para ellos un besote bien grande.

Para mi tío Papo y Jorge, sangre de mi sangre, son hombres de bien, no importa como actúen, siempre los quiero.

A mi madrina Hilda, que hace mucho tiempo que no la veo, deseo algún día tenerte de frente y darte un besote bien grande porque te lo mereces, porque aun de pequeño, recuerdo tus caricias; para ti mi madrina te dedico todo mi cariño.

También dedico este momento a Yuyú, Gerardo, y sus hijos, al profesor Eduardo Peña, a mi vecina Digna y a su hija Maité, a mis amigos de ajedrez en los tiempos de la EIDE, a Juan quien me enseña a hacer ejercicios físicos, a Nena, amiga íntima de mi abuela, a Alejandro y a Gustavo, los hermanos que me enseñaron lo poco que se de programación, a William, a su mamá hermana y tía, a la profe María de los Ángeles, a Robertico Chang, a Mario, a Coloma, a la China, su esposo y su nené, a mis doctoras Nairobis y Naomi y a su madre y tía.

A todo este arsenal de familia, les dedico con todo el corazón este resultado y espero que de ahora en lo adelante pueda obtener mejores resultados: todo gracias a ustedes.

Quiero agradecer primeramente a mi familia, especialmente a mi madre que ha sido y será siempre mi motivación, mi hermanita a la que quiero con locura y mi padre que me ha apoyado en todo, muchas gracias, porque sin ustedes no lo habría logrado.

Por supuesto, no podían faltar mi familia universitaria: Lourdes, Luis Ernesto, Francisco, Hansel, Maykel, Pedro, Ennery, Juan Pablo, Javier Nonell, Armando y Beatriz. Gracias chicos por todos esos momentos inolvidables que vivimos. Si existe algo que espero dure para siempre, ese algo tiene que ser nuestra amistad.

Finalmente, pero no menos importante, a esos que supieron tenerme paciencia y apoyarme. Muchas gracias a mis profesores de la UCI por prepararme como futuro ingeniero en especial a Siomara, Omar Garrido, Maydelis, Yoiler, a mis tutores por confiar en nosotros y nunca dejar que nos desanimáramos, y a mi compañero de tesis por compartir conmigo este tiempo de trabajo y esfuerzo.

Agradecimientos de Yordan

Quiero agradecer de por vida a mi tutora Madelis, mi madre en la escuela. Gracias por soportar todos mis caprichos de estudiante, supiste en el momento exacto como manejar la situación. Tan risueña pero exigente, evidencia su afán por el trabajo bien hecho; por eso mis respetos de hoy y siempre: la quiero mi profe.

También mis agradecimientos al profe Luis Carlos, Leandro, José Javier, Deylert, Alberto Arias, Rafael, el Butty y a todos los muchachones del proyecto CIGED. Saludos a los profes Yoiler, Abel, Omar, Maidelis y Reneé,

Para los amigos de escuela, los del 2101, 2201, 2301, 2401 y 2501, siempre juntos, inseparables. Mis saludos para Mahel, hermano de vida, compañeros de cuarto, de trabajo y estudio, amigo. Saludos para Andy, Jessica, Frank, Lourdes, Luis Ernesto, Pedro, Hansel, Liomar, Beatriz Piñar, Beatriz, Licor, Yissel, Liliet, Armando, Daymara, Betsy, Laura, Michel, Lianet, Luis Daniel, Lisandra, Zuleyma , para "my friend Yoandy" y especialmente para mi amigo y compañero de tesis Dariel, que siempre confió en mí y dio lo mejor de él para que este trabajo saliera adelante.

No puedo dejar de mencionar a Ernesto, Dario, Raidel, Alberto Emilio, Javier, Ennery, Yadir, Jossué, Osciel, Devorat, Eileen, Jessica Almanza, Bárbara, Claudia Marrero, Leyanet, Jipsy, Areleen, Claudia Alonso, Betsy Colas, Eliany, Yunexy, Yassel, Damir, Randy, Marlon, Ronald, Osmany, Solanch, Stefany, Roiby, Anna Laura, Anamelys, Maryeris, Dianelis, Elizabeth, Gelnda, a mis tías del comedor y a todos los amigos que de algún modo conocí y pasé excelentes momentos junto a ellos.

A todo los quiero mucho, mucho y mucho porque ustedes, sin excepción de ninguno, son los mejores

Resumen

En el presente trabajo investigativo se desarrolló un componente informático que permitió el intercambio de información bibliográfica del Sistema de Automatización de Bibliotecas y Centros de Documentación (ABCD), perteneciente a la Universidad de las Ciencias Informáticas (UCI), con otros sistemas. Para esto, se hizo uso del protocolo de búsqueda e intercambio de información bibliográfica *OAI-PMH*. El funcionamiento del componente informático se basa en la elaboración de peticiones que ofrece el estándar *OAI-PMH* a través de la barra de navegación del navegador *Web* instalado en el ordenador (proceso conocido también como método *GET*), por parte del usuario consultante de información. Como respuesta final hacia el usuario, se muestra una página *Web* con estructura *XML* según las especificaciones hechas en la petición.

Para el desarrollo investigativo, se utilizaron métodos científicos como el analítico-sintético y análisis documental, los cuales permitieron la organización en la investigación y la selección adecuada de las herramientas informáticas para el desarrollo de la aplicación. Posteriormente, se describió la arquitectura del sistema mediante el empleo de estilos y patrones arquitectónicos así como los patrones de diseño para asignar responsabilidades. Finalmente, se hace uso de estrategias de pruebas para comprobar el comportamiento y rendimiento en tiempo de ejecución de la aplicación.

Palabras claves: *protocolo OAI-PMH*, *interoperabilidad*, *metadatos, Servicio Web*

Índice General

Índice de tablas

Índice de figuras

Introducción

En el mundo actual, el desarrollo indetenible de las Tecnologías de la Información y las Comunicaciones (TICs), ha permitido gestionar, almacenar e intercambiar grandes colecciones de información bibliográfica en formato digital. Esto fue posible gracias a la creación de sistemas bibliotecarios digitales, los cuales proporcionaron servicios encargados de su divulgación.

Con el uso de los sistemas bibliotecarios digitales se aspiraba que el nuevo conocimiento generado fuese intercambiado por todo el mundo. Se tenía como objetivo quebrar cualquier barrera regional que existiese: una limitante hacia aquellas personas que necesitasen del acceso a la información bibliográfica generada fuera de sus sitios de origen o residencia. Algunos ejemplares de estos sistemas son los Sistemas Integrados de Gestión de Bibliotecas (SIGB) y los repositorios digitales.

"Los Sistemas Integrados de Gestión de Bibliotecas son *software* dedicados a la automatización de las operaciones bibliotecarias como la catalogación, circulación, administración y la adquisición de materiales bibliográficos" (Gavilán, 2008). En la actualidad existen diversos ejemplares de SIGBs, pero entre estos se destacan: *Koha*, *Evergreen*, *PHPMyBibli*, *OPALS*, *NewGenLib*, Sistema de Automatización de Bibliotecas y Centros de Documentación (ABCD) y *OpenBiblio* (Prieto, 2012).

"Los repositorios digitales son locaciones para gestionar, difundir y facilitar el acceso a los recursos bibliográficos que alberga una institución determinada. Se han creado herramientas que permiten el desarrollo de repositorios digitales. Algunos ejemplares destacados son: *DSpace*, *Bepress*, *ETD-db*, *DIVA*, *E-Prints*, *Open Publications System* (*OPUS*, por sus siglas en inglés) y *Fedora*" (Zubiri, y otros, 2012).

Conceptualmente, a este proceso de intercambio de información bibliográfica, se le conoce en la actualidad con el término de 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 uno o varios sistemas de información, mediante una interconexión libre, automática y transparente" (Gómez, 2007). Esta capacidad de interoperar es a través de protocolos de búsqueda e intercambio de información bibliográfica. Ejemplos de estos son: el protocolo Z39.50, el protocolo *OAI-PMH* y el protocolo *SWORD.*

Cuba no se encuentra ajena a estas tecnologías. La Universidad "Hermanos Saíz Montes de Oca", perteneciente a la provincia de Pinar del Río, posee un repositorio científico denominado "Alma" implementado con la herramienta *DSpace*. El objetivo que se persigue es reunir, archivar, preservar y compartir toda la producción intelectual resultante de la actividad académica e investigativa de la comunidad universitaria. Para el intercambio de información bibliográfica se hace uso del protocolo *OAI-PMH*. Estos recursos bibliográficos se encuentran disponibles en la dirección *http://rc.upr.edu.cu/oai*.

Sin embargo, en el país existen entidades que poseen sistemas bibliotecarios digitales que aún no son capaces de interoperar con otros sistemas debido a que estas tecnologías no han sido implementadas. La Universidad de las Ciencias Informáticas (UCI) es una unidad docente donde se combina el estudio, la producción y la investigación. Su misión es: "*Formar profesionales comprometidos con su Patria y altamente calificados en la rama de la Informática,*

además de producir aplicaciones y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación" (UCI, 2017).
Adscritos a diferentes estructuras administrativas, funcionan 15 centros que desarrollan actividades de Investigación + Desarrollo + Innovación (I + D + I), encargados de las aplicaciones informáticas, el desarrollo tecnológico y las investigaciones asociadas. De conjunto, conforman una red de trabajo colaborativo que opera bajo normas y procedimientos comunes, posibilitando la reutilización de componentes y eficiencia industrial. Los centros de desarrollo de *software* realizan más de 200 proyectos al año, con gran impacto en la informatización de la sociedad cubana.
Uno de estos centros es el Centro de Informatización de la Gestión Documental (CIGED) perteneciente a la Faculta 2. Aquí se desarrollan sistemas y servicios informáticos integrales de alta calidad y competitividad en la informatización o mejora de la Gestión Documental. Entre estos se destaca el sistema de Automatización de Bibliotecas y Centros de Documentación (ABCD). Actualmente la información bibliográfica que se gestiona en este sistema se encuentra almacenada en el sistema informático *J-ISIS*.
Es importante que esta información sea intercambiada para que otros sistemas, tanto dentro como fuera de la institución universitaria, puedan consultarla cada vez que fuese necesario. Para el centro CIGED constituye una desventaja que el Sistema ABCD no tenga implementado tecnologías de interoperabilidad. Esto trae como consecuencia la pérdida de promoción de los resultados intelectuales y académicos (tesis, trabajos científicos, artículos, documentación de producción de *software*) que se han realizado: un aspecto muy importante para toda la comunidad productiva universitaria. También provocaría el estancamiento de su información, pues puede ocurrir que un sistema bibliotecario determinado contenga documentación obsoleta y requiera de su búsqueda más actualizada en el Sistema ABCD.

Descrita la situación problemática, se plantea como **problema a resolver**: ¿Cómo intercambiar la información bibliográfica del Sistema ABCD con otros sistemas? Se define como **objeto de estudio**: la interoperabilidad del Sistema ABCD con otros sistemas y se considera como **campo de acción**: los protocolos destinados a la búsqueda e intercambio de información bibliográfica entre sistemas. Para solucionar el problema mencionado, se plantea como **objetivo general**: desarrollar un componente informático que permita intercambiar la información bibliográfica del Sistema ABCD con otros sistemas informáticos.

Para dar cumplimiento al objetivo general se proponen los siguientes **objetivos específicos**:

1. Analizar el marco teórico conceptual de la investigación y realizar un estudio referente a las características de los protocolos de búsqueda e intercambio de información bibliográfica y las funcionalidades asociadas al sistema informático *J-ISIS* para el desarrollo del componente informático.

2. Definir la propuesta de solución de la investigación relacionada al desarrollo del componente informático.

3. Implementar y validar las funcionalidades asociadas al componente informático mediante la aplicación de estrategias de pruebas para corroborar su comportamiento y rendimiento en tiempo de ejecución.

Con el propósito de cumplir los objetivos específicos propuestos, la presente investigación fue guiada por **Métodos científicos**. Para ello se utilizaron:

Métodos empíricos:

- **Análisis documental**: Propone pautas para expresar la información, adquirida de fuentes bibliográficas, de un modo preciso y exponiendo las ideas fundamentales. En la presente investigación se hace uso de este método para el estudio documental referente a los protocolos de búsqueda e intercambio de información bibliográfica y sobre las características del sistema informático *J-ISIS*.

Métodos teóricos:

- **Analítico - sintético**: Es utilizado en la reproducción del objeto de estudio en su totalidad en un plano teórico. La creación de lo concreto se efectúa sobre la base de la síntesis, reduciendo a la unidad las diversas propiedades y relaciones descubiertas en el objeto de que se trate. En la presente investigación se hace uso de este método para la extracción y análisis de los conceptos más importantes que guían la investigación. Además, ayuda a identificar las herramientas y tecnologías de desarrollo de *software* posibles a ser empleadas en el desarrollo de la aplicación.

El presente Trabajo de Diploma presenta la siguiente **estructura capitular**:

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA: En este capítulo se describen los conceptos por los cuales se rige la presente investigación. Luego, se realiza un estudio sobre las características y funcionalidades de los protocolos de búsqueda e intercambio de información bibliográfica y sobre el sistema informático *J-ISIS*. Finalmente, se justifica la selección de las herramientas y tecnologías de desarrollo de *software* que intervienen en la elaboración la aplicación.

CAPÍTULO II. PROPUESTA DE SOLUCIÓN: En este capítulo se describe la propuesta de solución del sistema mediante la elaboración del modelo de domino ya que ayuda a adquirir una mejor comprensión del funcionamiento de la aplicación. Luego, se hace uso de estilos y patrones arquitectónicos y los patrones de diseño para argumentar sobre la arquitectura de la aplicación.

CAPÍTULO III. IMPLEMENTACIÓN Y ETRATEGIAS DE PRUEBAS: En este capítulo se describe el proceso de implementación de la aplicación. Posteriormente, se efectúan estrategias de pruebas para comprobar su comportamiento y rendimiento en tiempo de ejecución. Finalmente, se argumentan los resultados observados de la aplicación de las pruebas.

Capítulo 1 Fundamentación teórica

1.1 Introducción

En toda investigación científica es necesario partir de un cúmulo de información conocida la cual compone el marco teórico que sustenta las soluciones propuestas. En el presente capítulo se describen los conceptos fundamentales por los cuales se rige la presente investigación. Posteriormente, se realiza un estudio sobre las características y funcionalidades de los protocolos de búsqueda e intercambio de información bibliográfica y sobre el sistema informático *J-ISIS*. Finalmente, se justifica la selección de las herramientas y tecnologías de desarrollo de *software* que intervienen en la elaboración la aplicación.

1.2 Conceptos fundamentales

Para adquirir una mayor comprensión del presente proceso investigativo es necesario puntualizar varios conceptos fundamentales.

1.2.1 Protocolo

“Conjunto de normas y procedimientos útiles para la transmisión de datos, que permiten que dos o más entidades se comuniquen entre ellas” (Estrada Corona, 2004).

1.2.2 Dato

“Representación simbólica, bien sea mediante números o letras, de una recopilación de información la cual puede ser cualitativa o cuantitativa, que facilitan la deducción de una investigación o un hecho” (Venemedia, 2014). Por ejemplo: José, 14, masculino, etc…

1.2.3 Metadato

“Son datos altamente estructurados que describen la información, el contenido, la calidad, la condición y otras características de los datos” (Nacional de Colombia, 2017). Por ejemplo: **Nombre**: José, **Edad**: 14, **Género**: Masculino, donde las palabras resaltadas son los metadatos.

1.2.4 Información

“Grupo de datos supervisados y ordenados, que se utilizan para construir un mensaje basado en un determinado fenómeno o situación” (Definicion.de, 2017).

1.2.5 Registro bibliográfico

"Representación de un documento mediante una serie de datos descriptivos establecidos por normas internacionales. Estos datos permiten identificar el documento y establecer puntos de acceso para su posterior recuperación entre un conjunto de registros del mismo tipo. De esta manera, el registro bibliográfico va a interactuar entre el usuario y el documento, informando sobre aspectos como el autor, el título, el editor o el contenido" (Bárzaga, y otros, 2014).

1.2.6 Recurso bibliográfico

"Un recurso es algo que tiene identidad. Son ejemplos típicos: un documento electrónico, una imagen, un servicio y una colección de otros recursos" (Powell, y otros, 2003).

1.2.7 Proveedor

"Entidad física o virtual que tiene el fin de ofrecer servicios a otras entidades" (DefinicionABC, 2010).

1.2.8 Base de datos

"Colección de datos lógicamente relacionados entre sí. Poseen una definición y descripción común y están estructurados de una forma particular" (Fuentes, 2013).

1.2.9 Sistema Gestor de Bases de Datos

"Los Sistemas Gestores de Bases de Datos (SGBD) son sistemas computacionales de administración de bases de datos que controlan el almacenamiento, organización, recuperación, integridad y seguridad de los datos y manejan todas las solicitudes de acceso a la base de datos" (Regalado Martínez, 2012).

1.3 Estudio acerca de protocolos de búsqueda e intercambio de información bibliográfica

La creación, desarrollo y utilización de estándares de comunicación para la obtención de información bibliográfica es una necesidad para los actuales sistemas de información. Esto hace imprescindible su implementación en las plataformas de gestión de información (SIGBs, repositorios digitales, etc…) que traigan integrados módulos específicos para acceder a estos servicios. Para el correcto desarrollo del presente estudio, se evidencian las características que son de gran importancia en los estándares de interoperabilidad.

1.3.1 Protocolo *OAI-PMH*

"El protocolo *OAI-PMH* (*Open Archives Initiative – Protocol Metadata Harvesting*, por sus siglas en inglés) fue diseñado específicamente para transmitir datos de un sistema de información hacia otros a través de Internet, tratando de ser sencillos en su implementación pero siendo muy claros en cuanto a las reglas de uso para promover el estándar y ser aplicado eficientemente al facilitar la difusión de contenidos sobre la *Web*, con el fin de convertirse en un mecanismo altamente usado para compartir recursos documentales a través de Internet" (Ramírez, 2007).

Historia de las versiones del protocolo *OAI-PMH*

Las especificaciones técnicas modificadas fueron hechas públicas en el año 2001 con la publicación del protocolo *OAI-PMH* versión 1.0 (v1.0), momento inicial de su implementación surgiendo las primeras instituciones que lo utilizaron para poner en Internet sus metadatos, pero con una adopción lenta y progresiva. Es por ello que en el año 2002 se implementa el protocolo *OAI-PMH* v2.0 que suprime los problemas que existían en su primera versión.

Estructura del protocolo *OAI-PMH*

Dentro del protocolo *OAI-PMH* participan dos actores fundamentales: el Proveedor de Servicios (*SP*, *Service Provider*, por sus siglas en inglés) y el Proveedor de Datos (*DP*, *Data Provider*, por sus siglas en inglés).

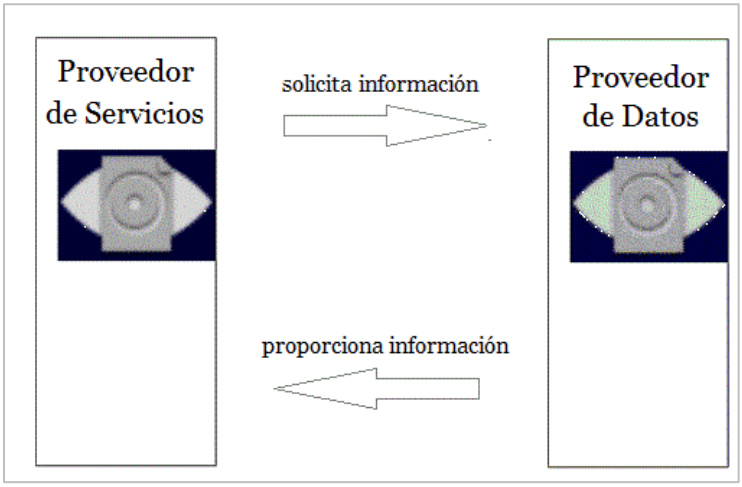


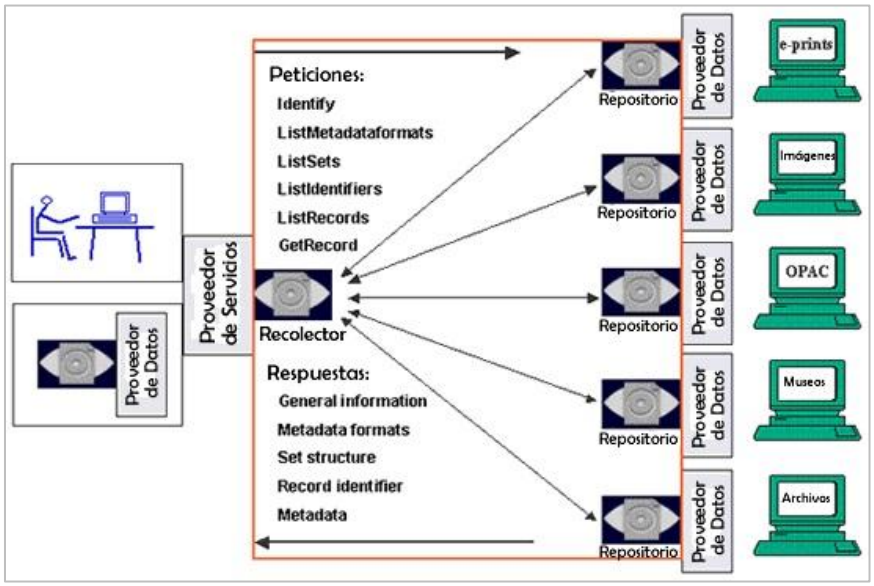
Figura 1.1 Estructura del protocolo *OAI-PMH*

El Proveedor de Servicios, desde el punto de vista informático, es una herramienta de investigación científica que permite realizar búsquedas de información sobre consultas específicas. Por tanto, puede ser un buscador estupendo para cualquier usuario o sistema que necesiten solicitar determinada documentación (contenido de revistas científicas, páginas de investigadores, cursos, patentes, repositorios institucionales, etc…).

Los Proveedores de Datos son entidades (sistemas bibliotecarios, repositorios digitales, etc…) que proporcionan información solicitada por usuarios o entidades determinadas (os).

Funcionamiento del protocolo *OAI-PMH*

El funcionamiento básico del protocolo *OAI-PMH* consta en formular peticiones (denominadas en ocasiones como "*verbo*") y obtener respuestas, entre el Proveedor de Servicios y el Proveedor de Datos, basado en la arquitectura Cliente-Servidor. Las peticiones y respuestas se administran en el Proveedor de Servicios mediante el Protocolo de Transferencia de Hipertexto (*HTTP*, *Hypertext Transfer Protocol*, por sus siglas en inglés) a través del método ***GET***. Las respuestas a las peticiones *OAI-PMH* las proporcionan los Proveedores de Datos. La figura 1.2 muestra funcionamiento del protocolo *OAI-PMH*, respectivamente.



Fuente: https://www.oaforum.org/tutorial/english/page3.htm

Figura 1.2 Funcionamiento del protocolo *OAI-PMH*

Primeramente, el usuario elabora de manera manual las peticiones *OAI-PMH* mediante el método ***GET*** o un Localizador Uniforme de Recursos (*URL*, *Uniform Resource Locator*, por sus siglas en inglés) sobre el navegador *Web*. Automáticamente, las peticiones son recibidas por el Proveedor de Servicios (mayormente son servicios *Web*), encargado de verificar si estas se encuentran bien elaboradas y luego enviarlas al Proveedor de Datos. Posteriormente, el Proveedor de Datos tiene como tarea fundamental extraer la información bibliográfica necesaria según las especificaciones establecidas en la petición *OAI-PMH*. Finalmente, esta respuesta es enviada al Proveedor de Servicios, el cual se encargará de crear una página *Web* con formato *XML* (*Extensible Markup Language*, por sus siglas en inglés) en el navegador (*Opera*, *Mozilla Firefox*, *Internet Explorer*, *Chrome*, *Safari*, etc…).

Sistemas bibliotecarios que utilizan el protocolo *OAI-PMH*

Existen numerosas instituciones bibliotecarias que hacen uso del estándar *OAI-PMH* para divulgar su información bibliográfica. Para mayor información, estos listados se encuentran disponibles en la dirección *https://www.openarchives.org/Register/BrowseSites*. A continuación, en la tabla 1.1 se evidencian varios ejemplos:

Tabla 1.1 Sistemas bibliotecarios que utilizan el protocolo *OAI-PMH*

Fuente: https://www.openarchives.org/Register/BrowseSites

Sistemas bibliotecarios	Disponible
Universidad Complutense	http://eprints.ucm.es/cgi/oai2
Digitala Vetenskapliga Arkivet	http://www.diva-portal.org/dice/oai
Universidad Europea	http://abacus.universidadeuropea.es/oai/request
Aberdeen University Research Archive	http://aura.abdn.ac.uk/dspace-oai/request
'Anil Islam	http://jurnal.instika.ac.id/index.php/AnilIslam/oai

1.3.2 Protocolo Z39.50

El protocolo *Information Retrieval* (Z39.50); *Application Service Definition and Protocol Specification* es utilizado para normalizar la comunicación entre dos computadores y recuperar información entre sí. Basado en la arquitectura Cliente-Servidor, especifica los procedimientos, formatos y funciones necesarias para recuperar la información en bases de datos (ANSI/NISO, 2003). Fue desarrollado por la Organización de Estándares de Información Nacional (*NISO*[1], *National Information Standards Organization*, por sus siglas en inglés) en Estados Unidos a partir del año 1970.

El éxito que presentó este protocolo fue proporcionado por la creación de catálogos bibliográficos durante la década de los años 80 y los 90. En aquel entonces se intentaba sistematizar la búsqueda, catalogación y gestión de los documentos de las instituciones, donde el tratamiento manual mediante fichas bibliográficas se tornaba tedioso. Pero a pesar de eso, se logró sistematizar utilizando metadatos en formato *MARC*. Es por ello que se desarrolla una aplicación nombrada "*Linked Systems Project*[2]" que agrupó a varias instituciones como la Biblioteca del Congreso de E.E.U.U (***Library of Congress***[3], en inglés), el Centro de Bibliotecas Computarizadas en Línea (*OCLC*[4], *Online Computer Library Center*, por sus siglas en inglés) y la Red de Información para Investigación en Bibliotecas.

Historia de las versiones del protocolo Z39.50

La primera versión del estándar de interoperabilidad Z39.50 se aprobó en al año 1988. En 1990 se establecieron dos importantes grupos que garantizan el desarrollo controlado y la continua evolución del protocolo: un grupo de implementadores *ZIG* (Z39.50 *Implementors Group*) y una agencia para su soporte del estándar (Z39.50 *Maintenance Agency*), fruto de su trabajo se aprueba la segunda versión en el año 1992 que, además de numerosas mejoras, evita las incompatibilidades con el protocolo de ISO "*Search and Retrieve*" (ISO 10162 y 10163). En el año 1997 la tercera versión del protocolo fue aceptada como estándar ISO (ISO 23950), incorporando mejoras cuyo principal propósito fue facilitar el desarrollo de un gran número de implementaciones de Z39.50 que corrieran sobre el modelo *TCP/IP*, de manera tal que fueran accesibles a través de Internet, convirtiendo esta versión del protocolo en una herramienta de gran presencia en la comunidad bibliotecaria hasta la actualidad.

Estructura del protocolo Z39.50

[1] *http://www.niso.org*

[2] *https://static-content.springer.com/lookinside/chp%3A10.1007%2F978-94-009-5452-6_31/000.png*

[3] *https://www.loc.gov*

[4] *https://www.oclc.org*

El protocolo Z39.50 está compuesto por:

- Proveedor de Servicios. Cliente (***origin***, en inglés), también conocido como Cliente Z (Z39.50), es el encargado de comunicarse con procesos auxiliares que se encargan de establecer conexión con el Servidor Z (Z39.50), enviar el pedido, recibir la respuesta, manejar las fallas y realizar actividades de sincronización y seguridad. El usuario interactúa con el cliente de forma gráfica.

- Proveedor de Datos. Servidor (***target***, en inglés) conocido como Servidor Z (Z39.50), es el encargado de proporcionar un servicio al cliente y entrega los resultados de una tarea específica, estableciendo procesos auxiliares que se encargan de recibir las solicitudes del cliente, verificar la protección, activar un proceso servidor para satisfacer el pedido, recibir su respuesta y enviarla al cliente.

Funcionamiento del protocolo Z39.50

El objetivo principal del protocolo Z39.50 consiste en permitirle al usuario realizar búsquedas en bases de datos que cuenten con un Servidor Z, sin tener que conocer para ello las sintaxis de búsqueda que utilizan dichos sistemas. Formalmente, facilita la interconexión entre los usuarios y las bases de datos donde se encuentra la información que necesitan a partir de una interfaz común y de fácil manejo, independientemente del lugar en que se encuentren las bases de datos así como la estructura y la forma de acceso de estas. Al tratarse de un protocolo orientado a conexión, este gestiona los diferentes mensajes que se transmiten en la sesión creada entre el cliente y el servidor, a través de procedimientos y formatos que permiten recuperar información, llamados registros bibliográficos o metadatos en el ambiente bibliotecario, desde bases de datos remotas, con el fin de acceder de manera uniforme y simultánea a los diferentes sistemas de información relacionados con los criterios de búsqueda.

Fuente: http://www.loc.gov/z3950/gateway.html

Figura 1.3 Funcionamiento del protocolo Z39.50

En la figura 1.3 se evidencia el funcionamiento del estándar Z39.50. Como ejemplo de ello, se desea realizar consultas de información sobre la Universidad *Adelphi*. Estas búsquedas se realizan desde el sitio *Web* de la Biblioteca del Congreso de los Estados Unidos disponible en la dirección *http://www.loc.gov/z3950/gateway.html*. Obligatoriamente, el usuario consultante debe definir los términos de búsqueda, los cuales deben coincidir con los filtros de búsqueda la como el autor, el título, la materia, el número de control etc…

Sistemas bibliotecarios que utilizan el protocolo Z39.50

Las instituciones que hacen uso de este estándar de interoperabilidad, se registran en el sitio *Web* oficial de la Biblioteca del Congreso de E.E.U.U. Para mayor información, estos listados se encuentran disponibles en la dirección *http://www.loc.gov/z3950/gateway.html*. Aquí se almacena un listado actualizado. La tabla 1.2 muestra algunos ejemplos de estos sistemas bibliotecarios digitales:

Tabla 1.2 Sistemas bibliotecarios que utilizan el protocolo Z39.50
Fuente: http://www.loc.gov/z3950/gateway.html

Sistema bibliotecarios	Disponible
Library of Congress	https://www.loc.gov
Abilene Library Consortium	https://www.alc.org
Murray State University	https://www.murraystate.edu
Albany State University	https://www.asurams.edu
Adelphi University	https://*www.adelphi.edu*

1.3.3 Protocolo *SWORD*

El Servicio *Web* simple que ofrece servicios de un depósito en un repositorio (*SWORD*, *Simple Web-service Offering Repository Deposit*, por sus siglas en inglés) es un protocolo ligero utilizado para efectuar envíos de contenidos desde un repositorio a otro (JISC, 2017).

El protocolo *SWORD* fue originalmente una iniciativa financiada por el Comité de Sistemas de Información Conjunta (*JISC*[5], *Joint Information Systems Committee*, por sus siglas en inglés) para definir y desarrollar un mecanismo estándar para depositar información en repositorios y otros sistemas.

[5] *https://www.jisc.ac.uk*

Historia de las versiones del protocolo *SWORD*

La primera versión del protocolo *SWORD* se desarrolló en el año 2007 para abordar la necesidad de una interfaz de depósito estandarizada para los repositorios digitales. Gracias al financiamiento del *JISC* (proyecto conformado por expertos en repositorio digitales del Reino Unido) desarrolló el este estándar y lo implementó para las plataformas de repositorio digitales como *DSpace*, *E-Prints*, *Fedora* e *Intralibrary*. La segunda versión fue diseñada e implementada en el año 2011 con fondos adicionales del *JISC*. Esto proporcionó mecanismos para no sólo depositar recursos, sino también para actualizarlos, reemplazarlos y eliminarlos.

Estructura del protocolo *SWORD*

El protocolo *SWORD* se compone por una herramienta informática (cliente) que permite el depósito de información bibliográfica en los repositorios. *Easy Deposit Client* es un ejemplo de esta herramienta.

Funcionamiento del protocolo *SWORD*

El funcionamiento del protocolo se basa en dos operaciones básicas:

- ***ServiceDocument***: operación encargada de seleccionar el repositorio destino para depositar la información bibliográfica.

- ***Deposit***: operación encargada de colocar la información bibliográfica en el repositorio destino.

Primeramente, se selecciona el repositorio destinado a recibir información mediante una dirección *URL*. En esta dirección se coloca al final la palabra reservada ***ServiceDocument*** o ***Service*** haciendo alusión a la operación antes mencionada, por ejemplo: *http://sword.intralibrary.com/IntraLibrary-Deposit/service.* Luego, se selecciona la colección de información bibliográfica que se desea depositar en el repositorio destino mediante la operación ***Deposit***. En este caso se eligen los materiales de investigación (***Research Materials****, en inglés*). Posteriormente, se describe la información bibliográfica a depositar mediante el título, los autores, el resumen, el tipo de documento y de manera opcional las citas bibliográficas y si existe una dirección *URL* del tipo de documento. En este caso el tipo de documento selecciona fue artículo de revista (***Journal Article***, en inglés). Acto seguido, se selecciona la ubicación del documento a depositar y finalmente se verifican los procedimientos realizados previamente para el depósito final. Para realizar estos pasos puede auxiliarse en la dirección https://repositorynews.wordpress.com/2010/06/04/easydeposit-the-sword-client-creation-toolkit/

Sistemas bibliotecarios que utilizan el protocolo *SWORD*

A continuación, la tabla 1.3 evidencia varios ejemplos de sistemas bibliotecarios que hacen uso del protocolo *SWORD*.

Tabla 1.3 Sistemas bibliotecarios que utilizan el protocolo *SWORD*

Fuente: http://swordapp.org/sword-v1/sword-v1-implementations

Sistemas bibliotecarios	Disponible

Science, Technology, Engineering and Mathematics (*STEM*)	https://www.stem.org.uk/
Department for Education	https://www.gov.uk
University of York	https://www.york.ac.uk
The City University of New York	http://www2.cuny.edu/libraries
Max Planck Digital Library's eSciDoc	http://colab.mpdl.mpg.de

1.3.4 Conclusiones del estudio de los protocolos

Con el estudio reciente acerca de los estándares de interoperabilidad, se obtuvo conocimiento sobre sus características estructurales y funcionales. De manera adicional, se observaron algunos sistemas bibliotecarios digitales que utilizan estos protocolos para intercambiar su información bibliográfica con otros sistemas. No obstante, es preciso seleccionar uno de ellos para que el Sistema ABCD interopere con otros sistemas, teniendo en cuenta dos criterios:

- *Software* libre como iniciativa para la Universidad de las Ciencias Informáticas
- Implementación a nivel mundial

***Software* libre como iniciativa para la Universidad de las Ciencias Informáticas**

Según la Fundación para *Software* Libre (*FSF*, *Free Software Foundation*, por sus siglas en inglés), del proyecto GNU, *software* libre es una cuestión de la libertad de los usuarios de ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el *software*[6].

Para que un *software* determinado sea libre, el usuario debe poseer las cuatro libertades esenciales (FSF, 2017):

- Libertad 0: La libertad de ejecutar el programa como se desea, con cualquier propósito.
- Libertad 1: La libertad de estudiar cómo funciona el programa, y cambiarlo para que haga lo que usted quiera. El acceso al código fuente es una condición necesaria para ello.
- Libertad 2: La libertad de redistribuir copias para ayudar a su prójimo.
- Libertad 3: La libertad de distribuir copias de sus versiones modificadas a terceros (libertad 3). Esto le permite ofrecer a toda la comunidad la oportunidad de beneficiarse de las modificaciones. El acceso al código fuente es una condición necesaria para ello.

Este es un criterio muy importante para aquellas entidades productivas que requieran la utilización de manera libre del *software*. En la Resolución 179/14 perteneciente a la Dirección de Servicios Jurídicos de la UCI se plantea en el acuerdo No. 084 del año 2004 que "el Comité Ejecutivo del Consejo de Ministros identificó la necesidad de ejecutar acciones que garantizaran la migración ordenada y progresiva hacia aplicaciones y plataformas de código

[6] *http://www.gnu.org/philosophy/free-sw.es.html*

abierto, en concordancia con el desarrollo del proceso de informatización de la sociedad como parte de la ejecución por el país de una política orientada a alcanzar la seguridad, invulnerabilidad e independencia tecnológica" (UCI, 2014).
La tabla 1.4 ejemplifica las 4 libertades de *software* libre con respecto a los estándares de interoperabilidad estudiados. Para ello se realizaron consultas en los sitios oficiales que los financian.

Tabla 1.4 Protocolos de búsqueda e intercambio de información bibliográfica bajo las 4 libertades de *software* libre

Libertades de *software* libre	Protocolos		
	OAI-PMH	**Z39.50**	***SWORD***
0. La libertad de ejecutar el programa como se desea, con cualquier propósito.	Sí	Sí	Sí
1. La libertad de estudiar cómo trabaja el programa, y cambiarlo para que haga lo que usted quiera. El acceso al código fuente es una condición necesaria para ello.	Sí	No	Sí
2. La libertad de redistribuir copias para ayudar a su prójimo.	Sí	Sí	Sí
3. La libertad de distribuir copias de sus versiones modificadas a terceros.	Sí	Sí	Sí

Estas características evidencian que el protocolo Z39.50 no cumple con la Libertad 2, pues la totalidad de sus implementaciones validadas por la Biblioteca del Congreso de los Estados Unidos son de código cerrado. En el caso particular del protocolo *OAI-PMH https://www.openarchives.org/pmh/tools/*. La empresa *JISC*, ofrece el código del protocolo *SWORD* en la dirección *http://sourceforge.net/projects/sword-app*.

Implementación a nivel mundial de los protocolos estudiados

Otro de los factores estadísticos que tienen en cuenta los actuales sistemas bibliotecarios para la adopción de protocolos de búsqueda e intercambio de información bibliográfica es el número o porcentaje de sus implementaciones a nivel mundial.
Para el caso de los protocolos estudiados, se puede estimar el número de implementaciones, ya que existen entidades encargadas de su desarrollo, mantenimiento e implementación. Estas instituciones recomiendan registrar dichas utilizaciones en sus sitios *Web* oficiales. Aunque el protocolo Z39.50 es un software privativo, varias entidades hacen de uso para intercambiar información; sus implementaciones el protocolo Z39.50, deben estar registradas en el sitio *Web* de la Biblioteca del Congreso de los Estados Unidos disponible en la dirección *http://www.loc.gov/z3950*. Las implementaciones del protocolo *OAI-PMH* se encuentran registradas en el sitio *Web* de la *OAI* disponible en la dirección *https://www.openarchives.org/Register/BrowseSites*, para el caso del protocolo *SWORD,* las implementaciones deben estar registradas en el sitio *http://swordapp.org/sword-v1/sword-v1-implementations*.

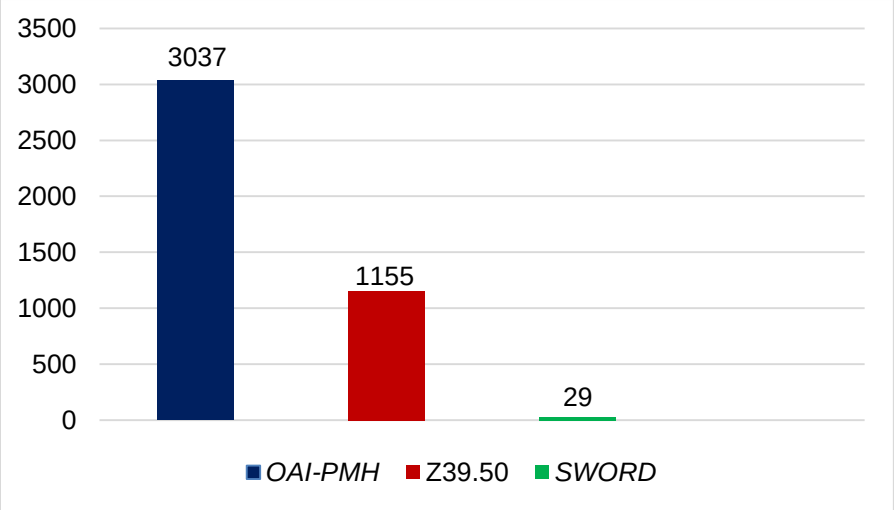


Figura 1.4 Número de implementaciones de los protocolos *OAI-PMH*, Z39.50 y *SWORD*

En la figura 1.4 se puede observar el número de implementaciones a nivel mundial de los protocolos mencionados previamente. El protocolo *OAI-PMH*, cuenta con 3037 implementaciones frente a las 1155 del protocolo Z39.50 y las modestas 29 del protocolo *SWORD*. Con estas estadísticas numéricas se evidencia la vasta aceptación que posee el estándar *OAI-PMH* a lo largo de todos estos años desarrollo desde su fecha de creación.

Concluido el análisis acerca de los criterios para la adopción del protocolo de búsqueda e intercambio de información bibliográfica, se selecciona el estándar *OAI-PMH.* Este cumple con las 4 libertades de *software* libre que propone la *FSF* y además es el más implementada a nivel mundial.

1.4 Sistema para la Automatización de Bibliotecas y Centros de Documentación

La evolución de los SIGBs va paralela a la rápida evolución de la tecnología y ha ido adaptándose a contextos de información cada vez más amplios, a usuarios más exigentes y a entornos de trabajo muy tecnificados. El Sistema ABCD es uno de los SIGBs más reconocidos para la automatización de bibliotecas.

En el centro CIGED de la UCI se interactúa con el Sistema ABCD para automatizar información. Su funcionamiento de manera general es mediante los módulos que lo componen:

- **Módulo de Catalogación**: Permite la construcción del instrumento que facilita el acceso a los documentos: *el catálogo*. Es esencial en cualquier SIGB porque sin un catálogo detrás estos sistemas no podrían funcionar. Aquí se crean registros bibliográficos, se normalizan entradas y se mantienen los índices.

- **Módulo de Circulación**: Establece relaciones temporales entre la información bibliográfica y los registros de usuarios y sirve fundamentalmente para desarrollar las operaciones préstamo a domicilio. Gracias a la parametrización de la política de circulación de la biblioteca es posible gestionar una gran cantidad de operaciones relacionadas con la circulación física de los documentos que posee la biblioteca: permite gestionar los usuarios, la devolución, las reservas, el envío de avisos, las sanciones por retrasos en devolución, hacer recuentos de uso en sala, etc.

- **Módulo de Adquisición**: Ayuda a gestionar la tenencia de nuevos documentos en la Biblioteca. Utiliza información bibliográfica existente en el catálogo para ejecutar los pedidos, o permite la incorporación en el catálogo de descripciones suficientes para realizar el pedido. Permite además, gestionar los proveedores y los fondos presupuestarios destinados a la adquisición, lanzar reclamaciones de pedidos no recibidos o cancelar peticiones, además de gestionar desideratas.

- **Módulo *OPAC***: E Catálogo Público de Acceso en Línea (*OPAC*, *Online Public Access Catalogue*, por sus siglas en idioma inglés) muestra públicamente el contenido del catálogo a través de Internet. Es decir, es la interfaz que permite a los usuarios acceder a los recursos bibliográficos del sistema e interactuar con ellos. Las últimas generaciones de *OPAC* son web, y suelen incorporar a las tradicionales operaciones de búsqueda y recuperación de la información, otros servicios de valor añadido, algunos de ellos personalizados.

- **Portal *Web***: Cuenta con un portal *Web* para la interacción del usuario con el sistema.

- **Sistema para el almacenamiento de información**: La información que gestiona el Sistema ABCD se encuentra almacenada en el sistema informático *J-ISIS*.

1.5 Sistema informático *J-ISIS*

El sistema informático Java - Conjunto Integrado para Servicios de Información (*J-ISIS*, *Java - Integrated Set for Information Services*, por sus siglas en inglés) es un Sistema Gestor de

Bases de Datos que permite la elaboración y administración de bases de datos. Fue creado por el francés Jean-Claude Dauphin, desarrollador de *software* de la *UNESCO HQ*[7] en octubre del año 2008 (Dauphin, 2015).
Con el estudio del presente sistema, se pretende obtener conocimiento acerca de sus funcionalidades. En la investigación, solo será de interés la función relacionada al acceso de los registros bibliográficos, ya que el contenido de información que almacenan es el que se necesita intercambiar con otros sistemas bibliotecarios.

1.5.1 Funciones de *J-ISIS*

El SGBD *J-ISIS* presenta las siguientes funciones (Dauphin, 2015):

- Crear bases de datos desde el inicio definiendo, de manera personal, su contenido (tablas, registros bibliográficos, etc.…).
- Acceder a las bases de datos almacenadas.
- Añadir de nuevos registros bibliográficos en una base de datos determinada.
- Acceder a los registros bibliográficos de las bases de datos almacenadas en el sistema.
- Modificar, corregir o suprimir registros bibliográficos existentes en una base de datos.
- Recuperar registros bibliográficos por sus contenidos a través de búsquedas avanzadas.
- Exportar los registros bibliográficos o porciones de ellos acorde a los requerimientos del usuario.
- Ordenar los registros bibliográficos en cualquier secuencia deseada.

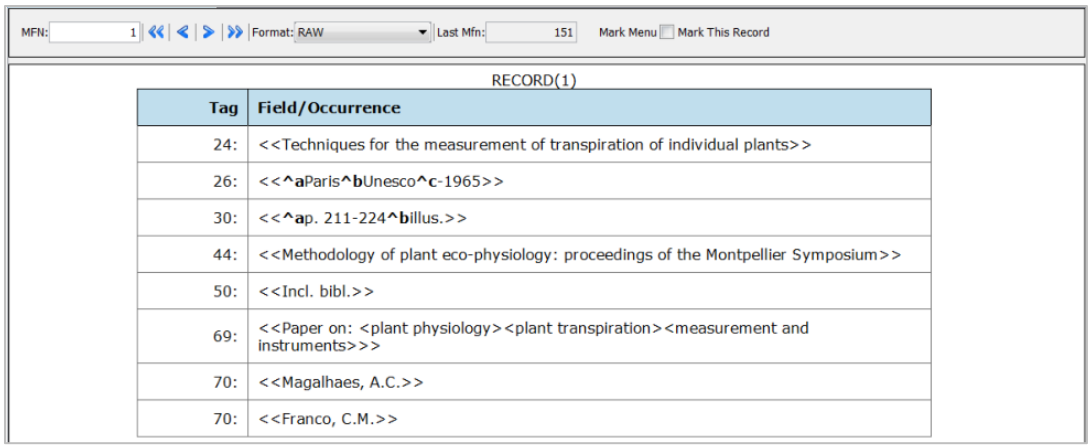
1.5.2 Acceso a los registros bibliográficos almacenados en las bases de datos de *J-ISIS*

El acceso a los registros bibliográficos almacenados en el SGBD *J-ISIS* es a través del Número de Archivo Maestro (*MFN*, *Master File Number*, por sus siglas en inglés). El *MFN* es un conjunto de archivos denominados *Berkeley DB*[8] en los cuales se almacenan todos los registros bibliográficos de una base de datos determinada albergada en el SGBD *J-ISIS*. Cada registro se identifica por un único *MFN* asignado automáticamente por el SGBD *J-ISIS* (Dauphin, 2015).

1.5.3 Estructura de los registros bibliográficos almacenados en *J-ISIS*

La información que se almacenan las bases de datos del SGBD *J-ISIS* es tratada con el término de *registro bibliográfico* (***record***, en inglés). Se componen por conjuntos de etiquetas, campos u ocurrencias, tipo de formato de metadato y por un *MFN*.
Los campos u ocurrencias (***field/ocurrence***, en inglés) almacenan los datos en específico de un registro bibliográfico. Las etiquetas (***tag***, en inglés) son asignaciones numéricas a cada campo u ocurrencia de un registro bibliográfico respectivamente. Los tipos de formatos de metadatos (***format***, en inglés) son las distintas formas de catalogar un registro bibliográfico y el *MFN* es un identificador numérico único de cada registro bibliográfico que permite el acceso a estos para la extracción de sus datos.

[7] *http://www.unesco.org*
[8] *https://www.usenix.org/legacy/event/usenix99/full_papers/olson/olson.pdf*


MFN: 1 Format: RAW Last Mfn: 151 Mark Menu Mark This Record

RECORD(1)

Tag	Field/Occurrence
24:	<<Techniques for the measurement of transpiration of individual plants>>
26:	<<^aParis^bUnesco^c-1965>>
30:	<<^ap. 211-224^billus.>>
44:	<<Methodology of plant eco-physiology: proceedings of the Montpellier Symposium>>
50:	<<Incl. bibl.>>
69:	<<Paper on: <plant physiology><plant transpiration><measurement and instruments>>>
70:	<<Magalhaes, A.C.>>
70:	<<Franco, C.M.>>

Fuente: Extraído de un registro bibliográfico almacenado en el SGBD *J-ISIS*

Figura 1.5 Estructura de un registro bibliográfico

1.6 Estudio y análisis de los metadatos

Una de las características fundamentales de los sistemas bibliotecarios digitales actuales, es la manera en que se van a representar los materiales bibliográficos: **los metadatos**. Son muy utilizados en el ambiente bibliotecario para describir documentos, objetos de información o de aprendizaje (tesis, libros, videos, artículos u otro recurso que apoye la difusión del conocimiento) para ser visualizados sobre un ambiente *Web*. A continuación se ejemplifican los beneficios que aporta la utilización de metadatos, específicamente para los usuarios que realizan búsquedas bibliográficas (Argueta, 2012):

- Encontrar los datos buscados.
- Conocer información que es clave en los datos.
- Comprender en profundidad la información.
- Localizar datos (dentro y fuera de la organización).
- Transferir e interpretar los datos correctamente.

Según la tipología de metadatos observadas anteriormente, dentro de los sistemas bibliotecarios se hacen uso de los metadatos descriptivos. Se utilizan para almacenar aquellos atributos que están relacionados intrínsecamente con el documento y que se pueden diferenciar unos de otros.

A continuación, se analizarán dos formatos de metadatos descriptivos: *MARC* y *Dublin Core*. Esto se debe a que el contenido que almacenan los registros bibliográficos del SGBD *J-ISIS* está catalogado con la estructura del primer formato mencionado. El segundo formato de metadato es estudiado ya que la aplicación a desarrollar debe catalogar y mostrar los registros bibliográficos bajo las pautas estructurales que él establece.

1.6.1 Formato de metadato *MARC*

MARC (*Machine Readable Catalogue or Cataloging*, por sus siglas en inglés) es un modelo de metadatos creado en el año 1965. Desde su surgimiento y evolución ha sido el más utilizado por las bibliotecas para sus catálogos bibliográficos, ya que facilitó en gran parte la

organización de todos los documentos existentes con el objetivo de mejorar su capacidad de recuperación (Ramírez, 2007). Del presente formato de metadato se lleva a cabo un estudio referente a su estructura para la catalogación de información digital y luego se evidencian los tipos de normas que posee.

Estructura del formato de metadato *MARC*

Un registro bibliográfico con la estructura del formato de metadato *MARC* está compuesto principalmente por tres elementos: la estructura del registro, la designación del contenido y los datos contenidos en el registro (Gavilán, 2008):

1. **Estructura del registro**: constituye la realización de la *American National Standard for Information Interchange* (ANSI/NISO Z39.2) o de la norma equivalente ISO 2709. La estructura de un registro bibliográfico *MARC*, que constituye su semántica, está compuesta por tres componentes principales:

    - **Cabecera**: elementos de información que esencialmente proveen datos para el procesamiento del registro. Los elementos de información contienen números o valores codificados; y se identifican por la posición relativa del carácter. La Cabecera posee una longitud fija de 24 caracteres y constituye el primer campo de un registro *MARC*.

    - **Directorio**: una serie de entradas que corresponden a la etiqueta, la longitud y el punto de inicio de cada campo variable dentro de un registro bibliográfico. Cada entrada posee una longitud de 12 caracteres de posición. Las entradas del Directorio correspondientes a los campos variables de control aparecen, primero, en una secuencia que sigue el orden numérico de las etiquetas de campo; le siguen las entradas de los campos de datos variables en orden ascendente de acuerdo al primer carácter de la etiqueta. La secuencia almacenada de los campos de datos variables del registro no corresponde necesariamente al orden de las entradas correspondientes del Directorio. Las etiquetas duplicadas se distinguen únicamente por la localización de los campos respectivos dentro del registro. El Directorio finaliza con un carácter terminador de campo.

    - **Campos de variables**: dentro de un registro bibliográfico *MARC*, los datos se organizan como campos variables, y cada uno de estos se identifica mediante una etiqueta numérica de tres caracteres, que se almacena en la entrada correspondiente en el Directorio. Cada campo termina con un carácter terminador de campo. El último campo de un registro finaliza tanto con un terminador de campo como con un terminador de registro.

2. **Designación del contenido**: etiquetas, códigos y convenciones establecidas explícitamente para identificar y caracterizar posteriormente los elementos de datos contenidos en el registro y para facilitar su manipulación. La designación de contenido, que constituye la sintaxis de un registro bibliográfico *MARC*, es propia de cada formato y es

precisamente lo identifica y distingue un formato *MARC* del resto de formatos. Existen dos tipos de campos variables:

- **Campos de control variables**: los campos de control variables son estructuralmente diferentes a los campos de datos variables. Estos campos no contienen ni indicadores ni códigos de subcampo. Pueden contener un solo dato (001 el número de control) o una serie de elementos de datos identificables por su posición de carácter (008).

- **Campos de datos variables**: en los campos de datos variables se utiliza los siguientes elementos para designar el contenido: **Indicadores**: las dos primeras posiciones al inicio de cada campo de datos variable contienen valores que permiten interpretar o completar los datos que se encuentran grabados en el campo a partir del segundo byte del primer subcampo. Los valores de los indicadores se interpretan por separado, según la posición primera o segunda, y pueden consisten en dígitos, caracteres alfabéticos o espacios en blanco. Por ejemplo, en el campo 245 el segundo indicador indica al sistema que debe de ignorar determinado número de caracteres a efectos de ordenación del título propio; **Subcampos**: los códigos de subcampo preceden cada elemento de datos dentro del campo que precisan un tratamiento separado. Un código de subcampo consiste en un delimitador seguido de un identificador de elemento de datos. Los identificadores de elementos de datos pueden ser representados por caracteres alfabéticos en minúsculas o numéricos. Los códigos de subcampos están definidos de forma independiente para cada campo, aunque el formato intenta preservar el mismo código para significados similares allí donde es posible. El objetivo de los códigos de los subcampos es identificar elementos de datos. El formato no establece el orden de los subcampos, que viene determinado por la norma prevista para el contenido de datos, como por ejemplo unas reglas de catalogación.

3. **Contenido de datos**: los datos contenidos en un registro *MARC* suelen estar definidos por normas que no forman parte del formato como las reglas de catalogación, lista de encabezamientos o clasificación que utiliza la agencia que crear el registro.

La siguiente tabla muestra la estructura de los campos de variables de datos del formato de metadatos *MARC*. Este formato presenta más etiquetas, pero en la presente investigación se evidencian las más fundamentales. Cada una de las ***X*** representa un número y cada combinación de ellos hace referencia a una característica diferente del elemento a catalogar.

Tabla 1.5 Etiquetas fundamentales del formato de metadatos *MARC*
Fuente: Tesis de Magíster: "*Estudio, análisis, evaluación e implementación de un protocolo de intercambio de contenidos sobre internet para la interoperabilidad de repositorios de la Biblioteca Digital*", por Juan Siza Ramírez, 2007
URL: http://*www.bdigital.unal.edu.co/1758/1/299759.2010.pdf*

Etiqueta	Descripción
0XX	Información de control, números de identificación y clasificación, etc.
1XX	Asientos principales.
2XX	Títulos y párrafo del título (título, edición, pie de imprenta).
3XX	Descripción física, etc…

4XX	Menciones de serie.
5XX	Notas.
6XX	Campos de acceso temático.
7XX	Asientos secundarios diferentes a los de materias y series; campos ligados.
8XX	Asientos secundarios de series, existencias, etc…
9XX	Reservados para implementación local.

Normativas del formato de metadato *MARC*

El formato de metadato *MARC* posee varias normas de catalogación bibliográfica. Esto se debe a la autorización establecida por el estándar ISO 2709/73 "*Format for bibliographic information interchange on magnetic tape*" la cual plantea que solamente los gobiernos deben estructurar y dar forma a los registros (Bukuyemskyy, 2017). Es importante destacar que la diferencia que existe entre unas normas y otras es la combinación de las etiquetas de catalogación bibliográfica analizadas en la tabla anterior.

Ya desde los años 70, varios países introdujeron sus propias normas *MARC* ajustándolas a sus propios requerimientos lingüísticos y culturales. Estados Unidos introdujo la norma *USMARC* (*Library of Congress*) al que siguen *CANMARC* de Canadá, *IBERMARC* de España, *UKMARC* de Gran Bretaña, *AUSMARC* de Australia entre otras. En el año 1999 Estados Unidos y Canadá unieron sus normas *USMARC* y *CANMARC*, apareciendo como resultado la norma *MARC*21 (Gavilán, 2008).

La norma *MARC*21 es una de las más difundidas y aceptadas internacionalmente. Está elaborada y auspiciada por la *Network Development and MARC Standards Office* (*NDMSO*, por sus siglas en inglés) de la *Library of Congress*, actualmente en colaboración con la sección *Standards* de la *Library and Archives Canada* y la *Bibliographic Standards and Systems de la British Library*. La *Library of Congress* se encarga de su publicación en idioma inglés y español, mientras que la *Library and Archives Canada* se encarga de su publicación en idioma francés (Gavilán, 2008).

Esta norma se compone de una familia de cinco formatos coordinados: ***MARC*21 para datos de autoridad**, ***MARC*21 para datos de fondos**, ***MARC*21 para datos de clasificación**, ***MARC*21 para datos información de la comunidad** y ***MARC*21 para datos bibliográficos**. Esta última norma fue diseñada para contener elementos de datos de varias tipologías documentales: libros, ficheros de ordenador, mapas, música, recursos continuos, grabaciones sonoras y material mixto. Es decir, es un formato concebido como soporte de la información generada por la descripción de cualquier material bibliográfico, desde un manuscrito a una revista electrónica de acceso remoto (Gavilán, 2008).

1.6.2 Formato de metadato *Dublin Core*

Dublin Core (DC) es un modelo de metadatos elaborado y auspiciado por la Iniciativa de Metadatos *Dublin Core* (*DCMI*, *Dublin Core Metadata Initiative*, por sus siglas en inglés), 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 (Dublin Core, 2016).

Es uno de los modelos de metadatos más conocidos para catalogar documentos bajo un ambiente *Web* debido a la estructura representativa que posee. Este estándar se creó a partir de un taller de metadatos realizado precisamente en la ciudad de *Dublin*, *Ohio* (Estados Unidos) en el año 1995 (Ramírez, 2007). Del presente formato se estudia sus tipos de normas con sus respectivas estructuras para catalogar información bibliográfica.

Normativas de *Dublin Core*

Al igual que el formato de metadato *MARC*, *Dublin Core* presenta sus normativas. En este caso, el formato posee dos normas y se clasifican en ***Dublin Core Simple*** (**sin cualificar**) **y *Dublin Core* Cualificado** (Dublin Core, 2016). En la presente investigación solo se estudiará la normativa ***Dublin Core Simple***

- **Normativa *Dublin Core Simple*** (**sin cualificar**)
  A diferencia de la normativa *MARC*21, las etiquetas de *Dublin Core Simple* son más sugerentes. *Dublin Core Simple* presenta 15 etiquetas para catalogar los recursos bibliográficos: Cobertura, Colaborador, Creador, Derechos, Descripción, Editor, Fecha de publicación, Formato, Fuente, Identificador, Idioma, Materia, Tipo de recurso, Título y Relación con otros documentos

Estos pueden clasificarse en tres categorías (Dublin Core, 2016):

1. Sobre el contenido del recurso:

Tabla 1.6 Contenido del recurso *Dublin Core Simple*
Fuente: http://dublincore.org/documents/dces

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

2. Sobre la propiedad intelectual del recurso:

Tabla 1.7 Propiedad intelectual *Dublin Core* Simple
Fuente: http://dublincore.org/documents/dces

Etiquetas *DC*	Descripción
DC.Contributor	Colaborador. Responsable de hacer colaboraciones al contenido del recurso bibliográfico.
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 bibliográfico se encuentre disponible.
DC.Rights	Derechos. Información sobre los derechos de la propiedad intelectual del recurso bibliográfico, como por ejemplo el *copyright*.

3. Sobre la instancia del recurso:

Tabla 1.8 Instancia *Dublin Core* Simple
Fuente: http://dublincore.org/documents/dces

Etiquetas DC	Descripción
DC.Date	Fecha. Fecha asociada a la creación o modificación del recurso bibliográfico. Se suele seguir la notación AAAA-MM-DD (año, mes y día, respectivamente).
DC.Format	Formato. Descripción física del recurso bibliográfico, como su tamaño, duración, dimensiones, etc. si son aplicables. Se suelen usar tipos de Extensiones Multipropósito de Correo de Internet (*MIME*[9], *Multipurpose Internet Mail Extensions*, por sus siglas en inglés).
DC.Identifier	Identificación. Referencia unívoca para el contenido del recurso bibliográfico. Por ejemplo una *URL* o un *ISBN*.
DC.Type	El tipo o categoría del contenido. Palabras clave de un vocabulario que describen la naturaleza del recurso bibliográfico.

1.6.3 Equivalencia entre formatos de metadatos

Con el estudio de las características de los formatos de metadatos *MARC* y *Dublin Core*, se obtuvo conocimiento acerca de la catalogación de materiales bibliográficos digitales en los sistemas bibliotecarios. Para ello, se hace uso de etiquetas que los describen y ordenan.

El contenido que se almacena en los registros bibliográficos del SGBD *J-ISIS* se encuentra catalogado bajo la normativa *MARC*21. Uno de los elementos estructurales que hace esto posible son: **las etiquetas.** Cada una de ellas contiene un número único para hacer referencia a la información que se está preservando en el **campo u ocurrencia** del registro bibliográfico.

A continuación, se evidencia la equivalencia entre las etiquetas de *MARC*21 y *Dublin Core Simple*.

[9] *http://es.ccm.net/contents/118-mime-extensiones-multiproposito-de-correo-internet*

Tabla 1.9 Equivalencia entre las etiquetas de *MARC*21 y *Dublin Core Simple*

Etiquetas *Dublin Core* (norma *Simple*)	Etiquetas *MARC* (norma *MARC*21)
DC.Contributor	**720 ##$**
DC.Coverage	**500$a**
DC.Creator	**720 ##$a**
DC.Date	**260 ##$c**
DC.Description	**520 ##$a**
DC.Format	**856 ##$q**
DC.Identifier	**856 40 $u**
DC.Language	**546 ##$a**
DC.Publisher	**260 ##$b**
DC.Rights	**540 ##$a**
DC.Source	**786 0#$n**
DC.Subject	**653 ##$a**
DC.Title	**245 00$a**
DC.Type	**655 #7$a**

1.7 Herramientas y tecnologías de desarrollo de *software* utilizadas para el desarrollo de la aplicación

1.7.1 Herramientas de Entorno Integrado de Desarrollo

Las herramientas de Entorno Integrado de Desarrollo (*IDE*, *Integrated Development Environment*, por sus siglas en inglés) son aplicaciones compuestas por un conjunto de herramientas útiles para un desarrollador (Alegsa, 2010).

Es editor de código (con facilidades como resaltado de sintaxis, completamiento de código y navegación entre clases), un compilador y herramientas de automatización de la compilación, un depurador y en algunos casos un constructor de interfaz gráfica.

Entorno Integrado de Desarrollo *NetBeans*

NetBeans es un *IDE* de código abierto dedicado a proporcionar productos sólidos de desarrollo de *software* que satisfacen las necesidades de los desarrolladores, usuarios y las empresas que dependen de esta herramienta como base para sus productos; especialmente, para permitirles desarrollar estos productos de forma rápida, eficiente y sencilla aprovechando los puntos fuertes del lenguaje de programación *Java* y otros estándares relevantes de la industria (NetBeans, 2017). Esta herramienta se encuentra disponible en la dirección *https://netbeans.org/downloads/ y* se utiliza en la investigación para el desarrollo de la aplicación.

1.7.2 Herramienta de Ingeniería de *Software* Asistida por el Ordenador

Las herramientas de Ingeniería de *Software* Asistida por el Ordenador (*CASE*, *Computer Aided Software Engineering*, por sus siglas en inglés) comprende un amplio abanico de diferentes

tipos de programas que se utilizan para ayudar a las actividades del proceso del *software*, como el análisis de requerimientos, el modelado de sistemas, la depuración y las pruebas. En la actualidad, todos los métodos vienen con tecnología *CASE* asociada, como los editores para las notaciones utilizadas en el método, módulo de análisis que verifican el modelo del sistema según las reglas del método y generadores de informes que ayudan a crear la documentación del sistema. Las herramientas *CASE* también incluyen un generador de código fuente a partir del modelo del sistema y de algunas guías de procesos para los ingenieros de *software*.

Herramienta *CASE Visual Paradigm* 8.0

Visual Paradigm 8.0 es un *software* que permite analizar, diseñar, codificar, probar y desplegar de manera automática todo tipo de negocio que se presente en la vida real (CCM, 2008).

Soporta lenguajes gráficos para el diseño y codificación tales como el Lenguaje Unificado de Modelado (*UML*, *Unified Modeling Language*, por sus siglas en inglés), Lenguaje de Modelación de Sistemas (*SysML*, *Systems Modeling Language*, por sus siglas en inglés), Modelo y Notación de Procesos de Negocio (*BPMN*, *Business Process Model and Notation*, por sus siglas en inglés), entre otros.

Diseñado para una amplia gama de usuarios, incluidos los ingenieros de *software*, analistas de sistemas, analistas de negocios, sistema de arquitectos, al igual que para aquellas personas interesadas en la construcción de sistemas de forma fiable a través de la utilización del enfoque orientado a objetos. Esta herramienta se encuentra disponible en la dirección *https://www.visual-paradigm.com/download/*. Es usada en la investigación para el modelado de clases y diagramas que posibilitan una mejor comprensión del funcionamiento de la aplicación.

1.7.3 Lenguajes de programación

Un lenguaje de programación es un sistema estructurado y diseñado principalmente para que las máquinas y computadoras se entiendan entre sí y con los humanos. Contiene un conjunto de acciones consecutivas que el ordenador debe ejecutar (Tecnología, 2009). Permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador para que este pueda comunicarse con los dispositivos *hardware* y *software* existentes.

Lenguaje Gráfico Unificado de Modelado

El Lenguaje de Modelado Unificado (*UML*, *Unified Modeling Language*, por sus siglas en inglés) es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de *software*. *UML* entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de *software* reutilizables (Universidad de Chile, 2011).

El lenguaje *UML* se utiliza en la presente investigación ya que base para la herramienta de desarrollo *Visual Paradigm* 8.0 posibilitando de esta manera el modelado de clases y diagramas que posibilitan una mejor comprensión del componente informático a desarrollar.

Lenguaje de Programación *Java*

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por la empresa *Sun Microsystems* (Oracle, 2005). Se selecciona este lenguaje para desarrollar ya que es el lenguaje base del *IDE NetBeans*.

Lenguaje de Marcado Extensible

El Lenguaje de Marcado Extensible (*XML*, *Extensible Markup Language*, por sus siglas en inglés) es un lenguaje muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos (W3C, 2014). Es un lenguaje muy similar al Lenguaje de Marcas de Hipertexto (*HTML*, *HyperText Markup Language*, por sus siglas en idioma inglés) pero su función principal es describir datos y no mostrarlos como es el caso de *HTML*. *XML* es un formato que permite la lectura de datos a través de diferentes aplicaciones.

Este lenguaje permite estructurar y almacenar cualquier tipo de información. Para ello, hace uso de etiquetas, las cuales pueden ser definidas personalmente cuando se construye un archivo con este formato.

Para el uso correcto de este formato, las etiquetas deben tener un comienzo y un fin. La etiqueta “<película>” indica su apertura, definiendo de esta manera la cantidad de información que se puede almacenar dentro de ella. También poseen atributos distintivos que suelen ser definidos a gusto personal y toman valores. En el ejemplo anterior, los atributos de la presente etiqueta son: tipo, sistema y año. La etiqueta “</película>” señala su cierre. Cada vez que se creen etiquetas, obligatoriamente tiene que existir su culminación ya que permitirá delimitar unas de otras. Este lenguaje es utilizado para mostrar la información bibliográfica extraída del SGBD *J-ISIS* en el navegador *Web*.

1.7.4 Servicio *Web*

Conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web y que intercambian datos entre sí con el objetivo de ofrecer servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la *Web*. Estos servicios proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario. (W3C, 2017).

Servicio *Web REST*

El Estado de Transferencia Representacional (*REST*, *Representational State Transfer*, por sus siglas en inglés) es un estilo arquitectónico que especifica restricciones como la interfaz uniforme que, si se aplican a un servicio *Web*, inducen propiedades deseables, como rendimiento, escalabilidad y modificabilidad, permitiéndoles una mejor funcionalidad en la *Web* (Oracle, 2013). Se hace uso de esta tecnología en la investigación ya que el componente informático a desarrollar es un servicio *Web*, y se aprovecha la ventaja de que admite la utilización de protocolos de búsqueda e intercambio de información bibliográfica.

1.7.5 Servidor *Web*

Los servidores *Web* (también conocidos como servidores *HTTP*) son programas utilizados para la distribución (entrega) de contenido en redes internas o en Internet (“servidor” hace referencia al verbo “servir”). Como parte de una red de ordenadores, un servidor *Web* transfiere

documentos a los llamados “clientes” (***clients*** en inglés), por ejemplo, una página *Web* a un explorador (Digital, 2017).

Servidor *Web Oracle GlassFish*

Oracle GlassFish fue la primera implementación hecha por la *Java Platform, Enterprise Edition* (*Java EE*, por sus siglas en inglés). Este servidor ofrece servicios de aplicaciones de manera flexible y ligera (Oracle, 2017). Esta tecnología es utilizada en la investigación como intermediario comunicativo entre el servicio *Web* a desarrollar y el SGBD *J-ISIS*.

1.7.6 *Framework* de desarrollo *Web*

Un Marco de Trabajo (***Framework***, en inglés) de desarrollo *Web* es una estructura definida, reusable en el que sus componentes facilitan la creación de aplicaciones *Web* (Delgado, y otros, 2017).

Framework Spring Web MVC

Spring Web MVC (***Model-View-Controller***, por sus siglas en inglés) es un *framework* de código abierto encaminado al desarrollo de aplicaciones haciendo uso del lenguaje de programación Java. Cuenta con un conjunto de librerías en las que se pueden escoger aquellas que faciliten el desarrollo de una aplicación. Entre sus posibilidades más potentes está su contenedor de Inversión de Control, también llamado Inyección de Dependencias, el cual es una técnica alternativa a las clásicas búsquedas de recursos vía Interfaz de Nombrado y Directorio Java (*JNDI*[10], *Java Naming and Directory Interfaces*, por sus siglas en inglés). *Spring* permite configurar las clases en un archivo *XML* y definir en él las dependencias (Minter, 2008).

Se utiliza esta tecnología para hacer uso de las librerías de programación que ofrece. Estas permiten la proyección de cualquier contenido de información a través de la *Web*, en caso particular, posibilita mostrar el contenido bibliográfico del SGBD *J-ISIS* en una página *Web* con estructura *XML*.

Conclusiones

El estudio documental y sintetizado acerca de las características estructurales y funcionales de los protocolos de búsqueda e intercambio de información bibliográfica, permitió la adopción del estándar de interoperabilidad *OAI-PMH*. Esto fue posible ya que este estándar de interoperabilidad cumple con las 4 libertades de software libre, lo que permitirá en un futuro las modificaciones personales para adaptarlas a cualquier escenario de producción. Además, ee el protocolo que más se implementa a escala mundial.

Con el breve análisis de las funcionalidades del SGBD *J-ISIS* se obtuvo conocimiento acerca del acceso a los registros bibliográficos que almacena. La descripción de los formatos de metadatos, permitió establecer equivalencia entre las etiquetas de *MARC21* y *Dublin Core Simple*. Esto fue de gran ventaja ya que las etiquetas de *Dublin Core Simple* son más sugerentes que las de *MARC*21 y le permitirá al consultante entender la información que consultó. Finalmente, las herramientas y tecnologías de desarrollo de *software* seleccionadas cuentan con los requerimientos necesarios y actualizados para el desarrollo de la aplicación.

[10] Interfaz de Programación de Aplicaciones (API) de Java para servicios de directorio.

Capítulo 2 Propuesta de solución

2.1 Introducción

En el presente capítulo se describe la propuesta de solución del sistema. Para ello, se elabora el modelo de domino, pues ayuda a comprender mejor el funcionamiento de la aplicación. Luego, se definen las características estructurales de la aplicación mediante los estilos y patrones arquitectónicos y patrones de diseño.

2.2 Descripción de la propuesta de solución

Actualmente, el Sistema ABCD no es capaz de intercambiar su información bibliográfica con otros sistemas. Para solucionar este inconveniente, se propone desarrollar un servicio *Web* (componente informático), haciendo uso de las buenas prácticas que ofrece el protocolo *OAI-PMH*.

Según la estructura que define el estándar de interoperabilidad *OAI-PMH*, la aplicación se comportaría como Proveedor de Servicio. De esta manera, este debe gestionar todas las peticiones *OAI-PMH* efectuadas por el usuario a través de la *URL* del navegador *Web*. Luego, debe enviarlas al SGBD *J-ISIS* (Proveedor de Datos), encargado de devolver respuestas ante cualquier tipo de petición *OAI-PMH*.

El Proveedor de Datos posee propiedades informativas que también serán socializadas, pero no son de gran envergadura para el usuario consultante. La información representativa es la que está contenida en los registros bibliográficos. Esta se encuentra catalogada bajo la norma *MARC*21. Por eso, antes de que sea extraída y posteriormente intercambiada, el SGBD *J-ISIS* debe establecer equivalencia entre las etiquetas *MARC*21 y las de *Dublin Core Simple*. Todas las respuestas a las peticiones *OAI-PMH* que proporcione *J-ISIS* deben estar en formato *XML*.

Es importante desatacar que la aplicación a desarrollar es completamente externa de la arquitectura del Sistema ABCD. De esta manera se evita cualquier fallo técnico que pueda presentar el sistema impidiendo, por consiguiente, el funcionamiento de la aplicación.

2.3 Modelo de dominio

El modelo de dominio es una representación visual de las clases conceptuales u objetos del mundo real en el dominio de interés. Utilizando la notación *UML*, este se representa mediante el diagrama de clases. Estos pueden mostrar (Larman, 2003):

- *Clases conceptuales*: son representaciones reales o subjetivas de un objeto determinado del mundo real.
- *Asociaciones*: son relaciones que se establecen entre las clases conceptuales que indican conexión significativa e interesante.
- *Atributos*: son valores de datos lógicos que poseen las clases conceptuales.

En la presente investigación se identificaron como:

- clases conceptuales: Usuario, Petición, Proveedor de Servicio, Proveedor de Datos y *XML*.

- asociaciones: elabora, recibe, envía, muestra.
- atributos: para la clase conceptual Petición se identificaron los atributos “verbo” y “argumento”.

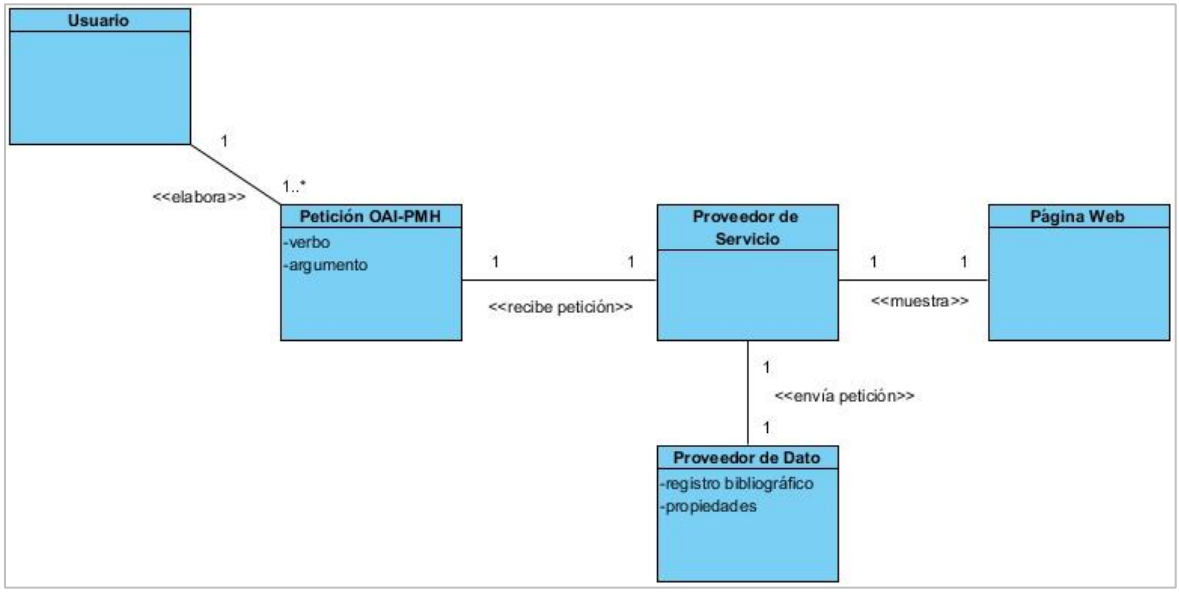


Figura 2.1 Diagrama del modelo de dominio

2.3.1 Descripción de las clases conceptuales del modelo de dominio

Nombre de la clase	Descripción
Usuario	Personal que consulta información bibliográfica. Comúnmente, debe acceder al sitio *Web* oficial del sistema bibliotecario, pero no requiere de autenticación para solicitar la información ya que es de acceso abierto. Luego, en la barra de navegación del navegador, al final de la dirección del sistema bibliotecario, elabora la sintaxis de cualquiera de las 6 peticiones *OAI-PMH*.
Petición *OAI-PMH*	Atributo que posee el protocolo *OAI-PMH* para consultar información bibliográfica. Posee parámetros que le permiten al usuario especificar el contenido bibliotecario que desea consultar. Con el parámetro “verbo”, el usuario define una de las 6 peticiones que soporta el protocolo *OAI-PMH* y con el parámetro “argumento”, formula rango o intervalos de búsqueda de búsqueda.
Proveedor de Servicio	Servicio virtual (mayormente servicios *Web*) que recibe y administra las peticiones *OAI-PMH* elaboradas por el usuario. Se encarga de enviarlas al Proveedor de Datos si y solo si fueron correctamente formuladas. Posee la tarea de crear las páginas *Web* con estructura *XML*, no solo cuando las peticiones se encuentren mal

	formuladas sino también ante cualquier respuesta que le envié el Proveedor de Datos una vez que este extraiga la información de sus sistemas de almacenamiento.
Proveedor de Dato	Entidad virtual (SGBD *J-ISIS*) que suministra la información bibliográfica. Recibe por parte del Proveedor de Servicios las especificaciones de las peticiones *OAI-PMH* si y solo si se encuentren bien elaboradas. En dependencia de estas especificaciones, el Proveedor de Datos se encarga de extraer el contenido almacenado en los registros bibliográficos. También puede ofrecer información acerca de las propiedades que lo caracterizan (petición ***Identify***). Finalmente, le envía al Proveedor de Servicio como respuesta, toda la información que se desea consultar.
Página Web	Respuesta final a una petición *OAI-PMH* realizada por el usuario consultante. Son generadas por el Proveedor de Servicios con formato *XML*. Estas se muestran en el navegador.

2.4 Arquitectura y diseño de la aplicación

Los estilos arquitectónicos son transformaciones impuestas al diseño de todo un sistema. El objetivo es establecer una estructura para todos los componentes del sistema (Pressman, 2010).

Aunque se han creado millones de sistemas de cómputo en los últimos 50 años, la gran mayoría puede clasificarse en un número relativamente pequeño de estilos arquitectónicos: Arquitectura centrada en datos, Arquitectura de flujos de datos, Arquitectura de llamada y retorno, Arquitectura orientada a objetos y Arquitectura estratificada (Pressman, 2010)**.**

En la presente investigación se utiliza el estilo arquitectónico Llamada y retorno. Este tipo de estilo arquitectónico permite que un diseñador de *software* (arquitecto del sistema) obtenga una estructura de programa que resulta relativamente fácil modificar cuando fuese necesario.

2.4.1 Patrones arquitectónicos

Un patrón arquitectónico, al igual que un estilo, impone una transformación en el diseño de la arquitectura. Sin embargo, difiere de un estilo en varios elementos fundamentales (Pressman, 2010):

- el alcance de un patrón es menor, ya que se concentra en un aspecto en lugar de hacerlo en toda la arquitectura;

- un patrón impone una regla sobre la arquitectura, pues describe la manera en que el *software* manejará algún aspecto de su funcionalidad al nivel de la infraestructura; los patrones tienden a abarcar específicos del comportamiento dentro del contexto de la arquitectura. Los patrones se utilizan en conjunto con un estilo arquitectónico para determinar la forma de la estructura general del sistema.

Dentro del estilo arquitectónico Llamada y retorno se encuentran los siguientes patrones arquitectónicos:

- Patrón Modelo-Vista-Controlador (MVC)
- Patrón N-Capas
- Patrón Orientado a Objetos
- Patrón Basado en Componentes

Para describir la arquitectura de la aplicación se hace uso del patrón Modelo-Vista-Controlador. Seguidamente se realiza una descripción de la arquitectura y se justifica su elección en la investigación:

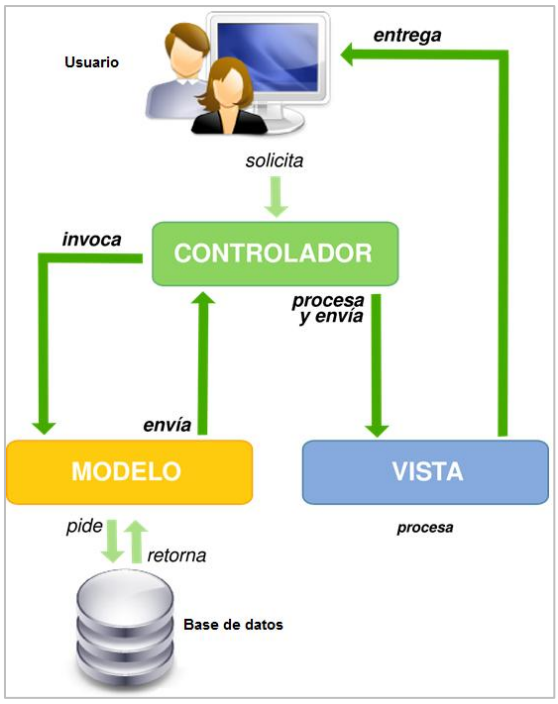


Figura 2.2 Funcionamiento del patrón arquitectónico Modelo-Vista-Controlador

Controlador: Responde a eventos (usualmente acciones del usuario) e invoca peticiones al componente "Modelo" cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos al componente "Vista" asociada si se solicita un cambio en la forma en que se presenta el componente "Modelo", por tanto, se podría decir que el "Controlador" hace de intermediario entre los componentes "Vista" y el "Modelo". Por lo tanto, el componente **Controlador** es el servicio *Web* ya que es el encargado de gestionar las todas las solicitudes *OAI-PMH* que sean elaboradas por el usuario mediante el método ***GET***.

Modelo: Es la representación de la información con la cual el sistema opera. Gestiona todos sus posibles accesos, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Por lo tanto, el componente **Modelo** son las peticiones del protocolo *OAI-PMH*. Es importante destacar que, exceptuando las peticiones "***Identify***", "***ListSets***" y "***ListMetadataFormats***", las restantes deben acceder al contenido almacenado en los registros bibliográficos. Esto conlleva a la conexión directa con el SGBD *J-ISIS*. Con cada petición que se realice a la base de datos, esta retorna la información necesaria al componente "Modelo"

Vista: Presenta al componente "Modelo" (información y lógica de negocio) en un formato adecuado para interactuar con el usuario (usualmente, las interfaces de usuario). Por lo tanto, el componente **Vista** son las páginas *Web* que se generan como respuesta ante las peticiones *OAI-PMH* que elabora el usuario. Estas respuestas la ofrece Proveedor de Datos (SGBD *J-ISIS*), pero el Proveedor de Servicio (servicio *Web*) es el encargado de estructurarla en formato *XML*.

2.4.2 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular (Larman, 2003).
Se propone utilizar los patrones *GRASP* (*General Responsibility Asignment Software Patterns*, Patrones Generales de *Software* para Asignar Responsabilidades, por sus siglas en inglés) ya que describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Constituyen el fundamento de cómo se diseñará el sistema, es decir, una serie de buenas prácticas de aplicación recomendable en el diseño de *software*.

Dentro de los patrones *GRASP* existen varios patrones para asignar responsabilidades pero los más importantes son los que se utilizarán en la presente investigación ya que se refieren a cuestiones y a aspectos fundamentales del diseño (Larman, 2003).

- **Patrón Controlador**: En términos de Programación Orientada a Objetos (POO), es la clase que tiene la responsabilidad de gestionar todos los eventos del sistema. Además, es el responsable directo de la interacción usuario-aplicación. Esta característica se evidencia en la clase *Controller.java* ya que se encarga de atender todas las peticiones *OAI-PMH* realizadas por el usuario mediante el método *GET.*

- **Patrón Creador**: En términos de POO, la clase "B" que tiene la responsabilidad de crear una instancia de la clase "A". La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos. En consecuencia conviene contar con un principio general para asignar las responsabilidades concernientes a ella. El diseño, bien asignado, poder soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reutilizabilidad. Esta característica se evidencia en la clase *OAI-PMH.java*. Esta clase se comunica con la clase controladora *Controller.java* y crea nuevas instancias de las peticiones *OAI-PMH* en correspondencia a las solicitudes que realiza el usuario.

- **Patrón Experto**: En términos de POO, es la clase que tiene la responsabilidad de conocer toda la información de las otras. Esta característica se evidencia en la clase *OAI-PMH.java*

- **Patrón Bajo Acoplamiento**: Tiene la responsabilidad de mantener bajo acoplamiento. En términos de POO, el **acoplamiento** es la medida con que un objeto (clase) está relacionado o conectado con otros. Una clase con bajo acoplamiento no depende de muchas otras. Una clase con alto acoplamiento recurre a muchas otras. Este tipo de clases no es conveniente ya que si ocurren cambios de codificación pueden afectarse una a otras. Esta característica se evidencia en las clases-peticiones *Identify.java*, *ListSets.java* y *ListMetadataFormats.java*

- **Patrón Alta Cohesión**: Asignar una responsabilidad de modo que la cohesión siga siendo alta. En términos de POO, la **cohesión** es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Una clase con baja cohesión no es conveniente ya que son difíciles de comprender, de reutilizar y de conservar. Estas características se evidencia en las clases-peticiones *GetRecord.java, ListIdentifiers.java* y *ListRecords.java*; *ListIdentifiers.java* pide los encabezados y *ListRecords.java* es listado de la clase *GetRecord.java*, respectivamente.

2.5 Descripción del funcionamiento del protocolo *OAI-PMH*

Tras estudiar el diseño y arquitectura de la aplicación a desarrollar, es momento de definir las funcionalidades del protocolo *OAI-PMH*. Este estándar, de manera simultánea, funcionará con la aplicación a desarrollar.

2.5.1 Listado de peticiones que ofrece el protocolo *OAI-PMH*

Precisamente, lo que permite el funcionamiento del protocolo *OAI-PMH* es la formulación de sus peticiones para consultar cualquier tipo de información bibliográfica. El protocolo *OAI-PMH* soporta 6 tipos de peticiones en las que el Proveedor de Servicios puede conocer las características del servidor y el sistema de información, y de igual forma conocer la estructura para la recolección de los metadatos (Barrueco, 2009).

A continuación, se evidencian las peticiones que ofrece el estándar, cada una de ellas con sus correspondientes descripciones y argumentos. El orden de representación que tienen en la tabla es necesario ya que si no se conoce la información que brinde una no se tendrá claridad para poder ejecutar otra petición.

Tabla 2.1 Listado de peticiones que ofrece el protocolo *OAI-PMH*
Fuente: http://www.openarchives.org/OAI/openarchivesprotocol.html

Tipo de petición	Descripción de la petición	Descripción del argumento
ListSets	*Listado de conjuntos o agrupaciones de materiales bibliográficos*. Petición que se utiliza para conocer cuáles son las agrupaciones o conjuntos (***sets***, en inglés) de materiales bibliográficos (libros, revistas, artículos, etc…) que posee un sistema bibliotecario.	No posee argumentos.
ListMetadataFormats	*Listado de formatos de metadatos*. Petición que se utiliza para conocer los formatos de metadatos que catalogan los registros bibliográficos. Los devuelve en prefijo (ejemplo: *oai_dc*). De manera opcional se utiliza el argumento *identifier*.	• ***identifier***: es un argumento opcional que se emplea para especificar el identificador único de los registros bibliográficos. Para utilizarlo es necesario conocer la repuesta que ofrece la petición ***ListIdentifiers***. No obstante, cuando no se utiliza este argumento, se devuelve de manera general el listado de formatos de metadatos que se utilizan para catalogar los registros bibliográficos almacenados en el sistema bibliotecarios.

ListIdentifiers	*Lista de identificadores*. Petición que se utiliza para conocer los identificadores únicos de los registros bibliográficos.	• ***from***, ***until***: son argumentos para establecer un rango de fecha de búsqueda de los registros bibliográficos; ***from*** (desde, en español) especifica el punto de partida o la fecha inicial y ***until*** (hasta, en español) señala el punto final o la fecha terminal). El formato de fechas es AAAA-MM-DD (año, mes y día, respectivamente). El argumento ***set*** se utiliza para definir el conjunto o agrupación de materiales bibliográficos que se desean consultar. Estos 3 son argumentos opcionales • ***metadataPrefix*** es un argumento requerido. Se aplica una vez conocido los formatos de metadatos después de ejecutar la petición ***ListMetadataFormats***.
GetRecord	*Obtener un registro bibliográfico*. Petición que solicita un registro bibliográfico en específico. Debe incluir, obligatoriamente, los argumentos *identifier* y *metadataPrefix*.	• ***identifier***, ***metadataPrefix***: son argumentos obligatorios.
ListRecords	*Listado de registros bibliográficos*. Petición que solicita un listado de los registros bibliográficos almacenados en un sistema bibliotecario. Debe incluir, obligatoriamente, el argumento *metadataPrefix*. De manera opcional y exclusiva, posee los argumentos *from*, *until* y *set* y *resumptionToken* respectivamente.	• ***from***, ***until*** y ***set*** (opcionales) • ***metadataPrefix*** (requerido)
Identify	*Identificación*. Petición que se utiliza para conocer la información tanto técnica como administrativa del Proveedor de Datos (sistema bibliotecario).	No posee argumentos

2.5.2 Sintaxis de las peticiones *OAI-PMH*

Todas las solicitudes o peticiones realizadas a un sistema de información que haya implementado el protocolo *OAI-PMH*, se efectúan a través de una dirección de internet o de un Localizador Uniforme de Recursos (*URL*, *Uniform Resource Locator*, por sus siglas en inglés). Seguidamente, se muestra una figura referente a la sintaxis de una petición del protocolo *OAI-PMH*:

- *http://direccion_sistema_bibliotecario/oai/request?verb=tipo_peticion&argumento*

El parámetro "***direccion_sistema_bibliotecario***" hace referencia a la dirección *URL* de la institución a la que va a extraer el contenido bibliográfico. El parámetro "***oai***" se refiere a la versión del protocolo *OAI-PMH* que se está utilizando. El parámetro "***request?verb=***" es una cadena de caracteres reservada que indica el comienzo la petición. El parámetro "***tipo_peticion***" señala el tipo de solicitud que se desea realizar. La simbología "***&***" indica que se agregarán argumentos a la petición si y solo si esta los posee como propiedad. Finalmente, el parámetro "***argumento***" se utiliza para establecer un rango de búsqueda bibliográfica y para agregar todas las propiedades que soporta una petición *OAI-PMH*.

A continuación, se observa la declaración de una petición *OAI-PMH* dirigida al Repositorio Institucional de la Universidad "Hermanos Saíz Montes de Oca", perteneciente a la provincia de Pinar del Río. En este sencillo ejemplo, la solicitud fue elaborada en el navegador *Web Mozilla Firefox*. En este caso articular, a dirección del sistema bibliotecario hace referencia a *rc.upr.edu.cu*. Se efectúa la búsqueda con la primera versión del protocolo *OAI-PMH* (si fuese la segunda versión sería *oai2*) y posteriormente se especifica el tipo de petición con el parámetro *driver?verb=*. En este último caso, algunas instituciones definen de manera personalizada el argumento *request,* pero esto no es de importancia siempre y cuando el servicio haga su función correctamente. Se desea del consultar un listado de documentos alojados en el repositorio que estén catalogados con la norma *Dublin Core* Simple (prefijo *oai_dc*):

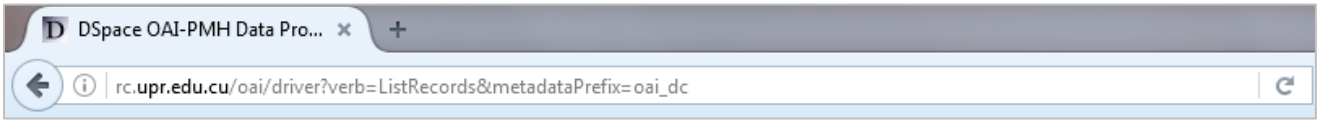


Figura 2.3 Petición *OAI-PMH* en tiempo de ejecución

2.5.3 Estructura de las respuestas a las peticiones *OAI-PMH*

Es muy importante comprender lo que proyectan las respuestas a las peticiones *OAI-PMH* en formato *XML*. Cada etiqueta hace referencia a una información en específico. Seguidamente, se muestra una tabla con las etiquetas globales, o sea, siempre van a ser visibles ante cualquier petición.

Tabla 2.2 Estructura de las etiquetas globales en formato *XML*

Etiquetas *XML* globales	Descripción de las etiquetas
<oai-pmh>	Corresponde al inicio del funcionamiento del protocolo *OAI-PMH*.
<responseDate>	Corresponde a la fecha y hora en formato de

	Tiempo Universal Coordinado (*UTC*, *Universal Time Coordinated*, por sus siglas inglés), en la cual el Proveedor de Datos atendió la transacción *HTTP*.
<request>	Corresponde a la referencia que indica el "*verbo*" junto con todos los parámetros solicitados.

Conclusiones

La modelación del diagrama de clases permitió adquirir un mayor conocimiento acerca del funcionamiento de la aplicación. Con el uso de los estilos y patrones arquitectónicos, se definió la arquitectura del sistema, lo cual posibilitó obtener mejor visón acerca de los componentes que permiten el funcionamiento de la aplicación. El uso de patrones de diseño trajo como ventaja la asignación de responsabilidades a los componentes que intervienen en la aplicación. Esto fue muy ventajoso a la hora de mejorar la codificación del servicio *Web*.

Capítulo 3 Implementación y estrategias de pruebas

3.1 Introducción

En el presente capítulo se hace referencia al trabajo de implementación y estrategias de prueba al componente informático, los cuales son determinantes en el proceso de desarrollo. De esta manera, se muestran los diferentes artefactos que se generan como el diagrama de componentes y de despliegue. Finalmente, se definen las estrategias de pruebas que se le aplican al sistema para comprobar su comportamiento y rendimiento en tiempo de ejecución.

3.2 Diagrama de componentes

Un componente es un bloque de construcción de *software* de cómputo, es una parte modular, desplegable y sustituible de un sistema, que incluye la implantación y expone un conjunto de interfaces. Describen los elementos físicos del sistema y sus relaciones, representan todos los elementos que entran en la fabricación de aplicaciones informáticas; son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar la relación entre ellos. Es un grafo de componentes unidos a través de relaciones que pueden ser de ejecución o compilación (Pressman, 2010). Se modela a través de un diagrama teniendo en cuente el patrón arquitectónico seleccionado.

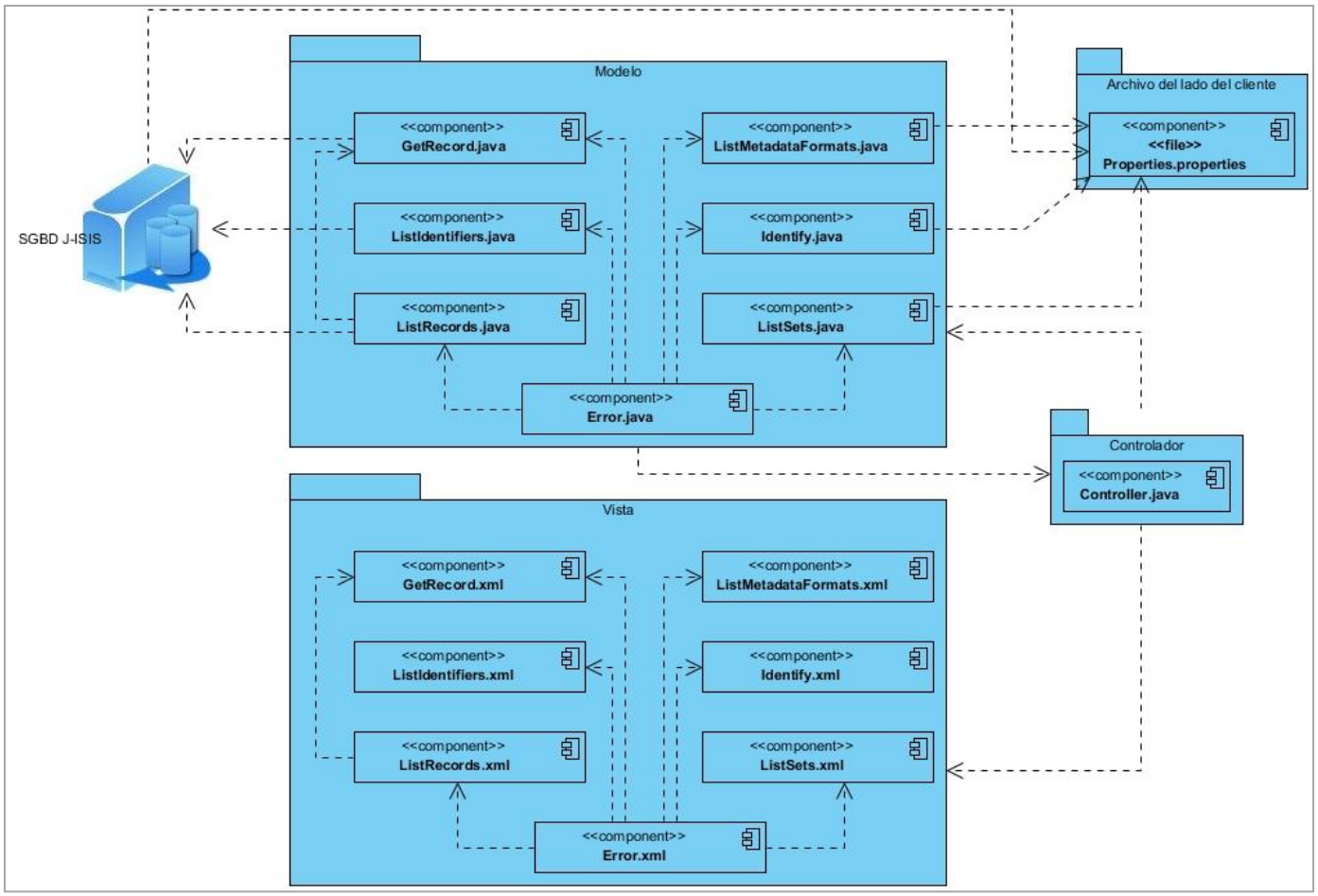


Figura 3.1 Diagrama de componentes

3.2.1 Descripción de los elementos del diagrama de componentes

Elementos del diagrama de componentes	Descripción de los elementos del diagrama de componentes
Controller.java	La clase ***Controller.java*** es la encargada de controlar todos los eventos del sistema. Cada petición *OAI-PMH* que se realice,
Properties.properties	Es un archivo que tiene almacenado los parámetros de conexión al SGBD *J-ISIS*. Contiene las propiedades de agrupaciones de materiales bibliográficos del sistema bibliotecario necesario para la clase ***ListSets.java***. Almacena las diferentes normas de catalogación de información necesarias para la clase ***ListMetadataFormats.java***

	Por último, posee los parámetros que identifican al sistema bibliotecario que se le consulta información, necesario para la clase ***Identify.java***
GetRecord.java	Clase que hace referencia a la petición ***GetRecord*** del protocolo *OAI-PMH.* Establece conexión con el SGBD *J-ISIS* para extraer toda la información del registro bibliográfico que se solicitó.
ListIdentifiers.java	Clase que hace referencia a la petición ***ListIdentifiers*** del protocolo *OAI-PMH*. La información que proporciona depende de la clase ***GetRecord.java***, ya que debe almacenar los encabezados de los registros bibliográficos.
ListRecords.java	Clase que hace referencia a la petición ***ListRecords*** del protocolo ***OAI-PMH***. La información que proporciona depende de la clase ***GetRecord.java***, ya que es un listado de esa clase.
ListMetadataFormats.java	Clase que hace referencia a la petición ***ListMetadataFormats*** del protocolo ***OAI-PMH***. La información que proporciona depende del archivo de configuración ***Properties.properties***, ya que aquí se almacena el contenido que debe mostrar.
Identify.java	Clase que hace referencia a la petición ***Identify*** del protocolo ***OAI-PMH***. La información que proporciona depende del archivo de configuración ***Properties.properties***, ya que aquí se almacena el contenido que debe mostrar.
ListSets.java	Clase que hace referencia a la petición ***ListSets*** del protocolo ***OAI-PMH***. La información que proporciona depende del archivo de configuración ***Properties.properties***, ya que aquí se almacena el contenido que debe mostrar.
Error.java	Es una clase que depende de todas. Se encarga de controlar los errores que

	puedan existir cuando el usuario elabore de manera incorrecta una petición *OAI-PMH*. Para cada petición, esta clase contiene el contenido de su error.
SGBD *J-ISIS*	Contiene toda la información de los registros bibliográficos que se desean consultar.
GetRecord.xml	Respuesta a la petición ***GetRecord*** como página *Web* con estructura *XML.*
ListIdentifiers.xml	Respuesta a la petición ***ListIdentifiers*** como página *Web* con estructura *XML.*
ListRecords.xml	Respuesta a la petición ***ListRecords*** como página *Web* con estructura *XML.*
ListMetadataFormats.xml	Respuesta a la petición ***ListMetadataFormats*** como página *Web* con estructura *XML.*
Identify.xml	Respuesta a la petición ***Identify*** como página *Web* con estructura *XML.*
Listsets.xml	Respuesta a la petición ***Listsets*** como página *Web* con estructura *XML.*
Error.xml	Respuesta a todos los errores de petición como página *Web* con estructura *XML.*

3.3 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 por enlaces de comunicación.

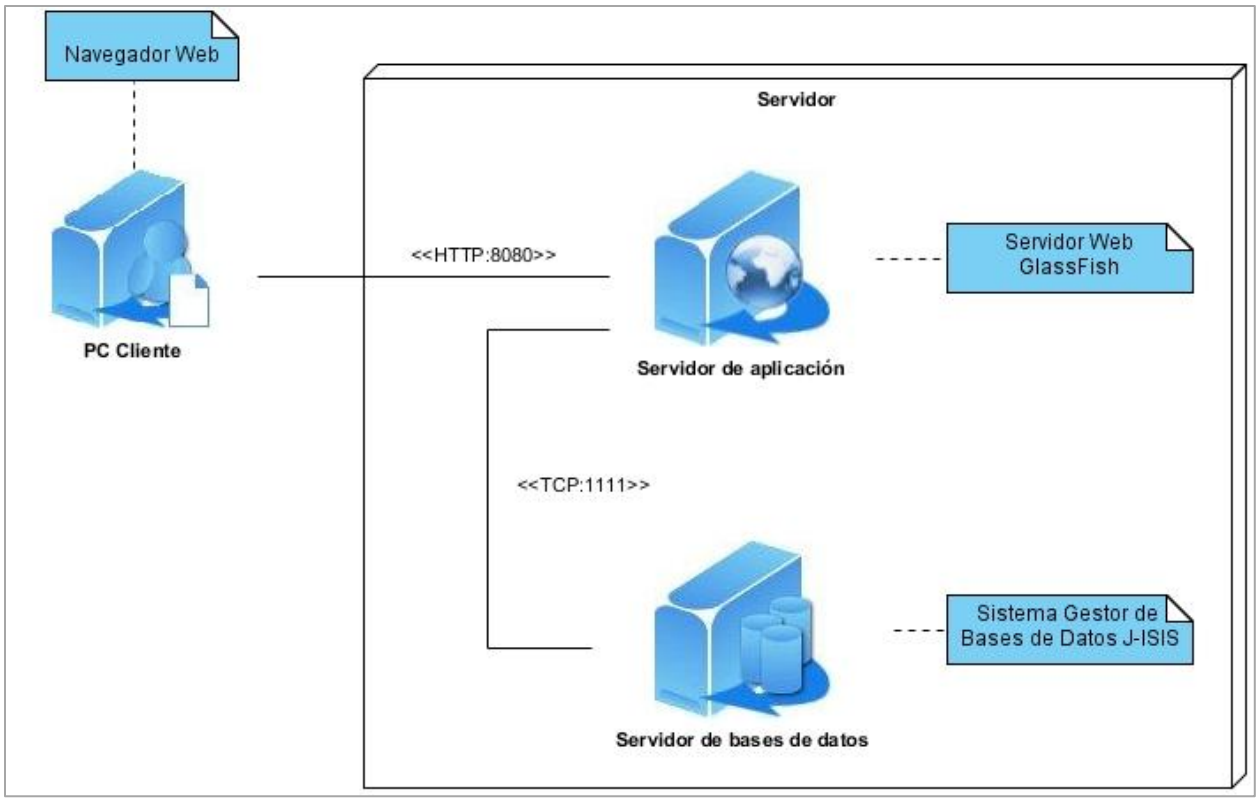


Figura 3.2 Diagrama de despliegue

3.3.1 Descripción de los elementos del diagrama de despliegue

Elementos del diagrama de despliegue	Descripción de los elementos del diagrama de despliegue
PC Cliente	Representación de la computadora personal (*PC*, *Personal Computer*, en inglés) del cliente. Con el uso de esta herramienta, el cliente comprobará; por medio de navegadores *Web*, el funcionamiento del servicio *Web*, haciendo uso del protocolo *OAI-PMH*. Se comunica con el servidor de aplicación a través del protocolo *HTTP* por el puerto 8080.
Servidor de Aplicación *Web*	Representación del servidor de aplicación *GlassFish* v4.1.1. Es la entidad virtual intermediaria que se encarga del flujo de información entre el servicio *Web* y el SGBD *J-ISIS*. La conectividad con el servidor de bases de datos se realiza a través del Protocolo

	de Control de transferencia (*TCP*, *Transfer Control Protocol*, por sus siglas en inglés) por el puerto 1111.
Servidor de Bases de Datos	Representación del SGBD *J-ISIS*. Aquí se almacena toda la información bibliográfica que se gestiona en el Sistema ABCD.

3.4 Desarrollo del servicio *Web*

Los autores de la presente investigación consideran importante realizar un desglose, paso a paso, de cómo se desarrolló la aplicación. En el presente epígrafe se describen estos pasos.

Paso #1: Creación del proyecto *Web*

Para desarrollar el servicio *Web* es hizo necesario utilizar el *IDE NetBeans*. Esta herramienta informática ofrece por defecto las librerías actualizadas del *Framework* de desarrollo *SpringMVC* (***Model-View-Controller***, en inglés respectivamente).

Se selecciona la opción "***File***" y se crea un nuevo proyecto en la opción "***New Project***". Acto seguido, se muestra un panel de selección de proyectos a crear. Se elige la categoría "***Java Web***" en el recuadro izquierdo y se señala la opción "***Web Application***" en el recuadro derecho. y selecciona la opción "***Next***".

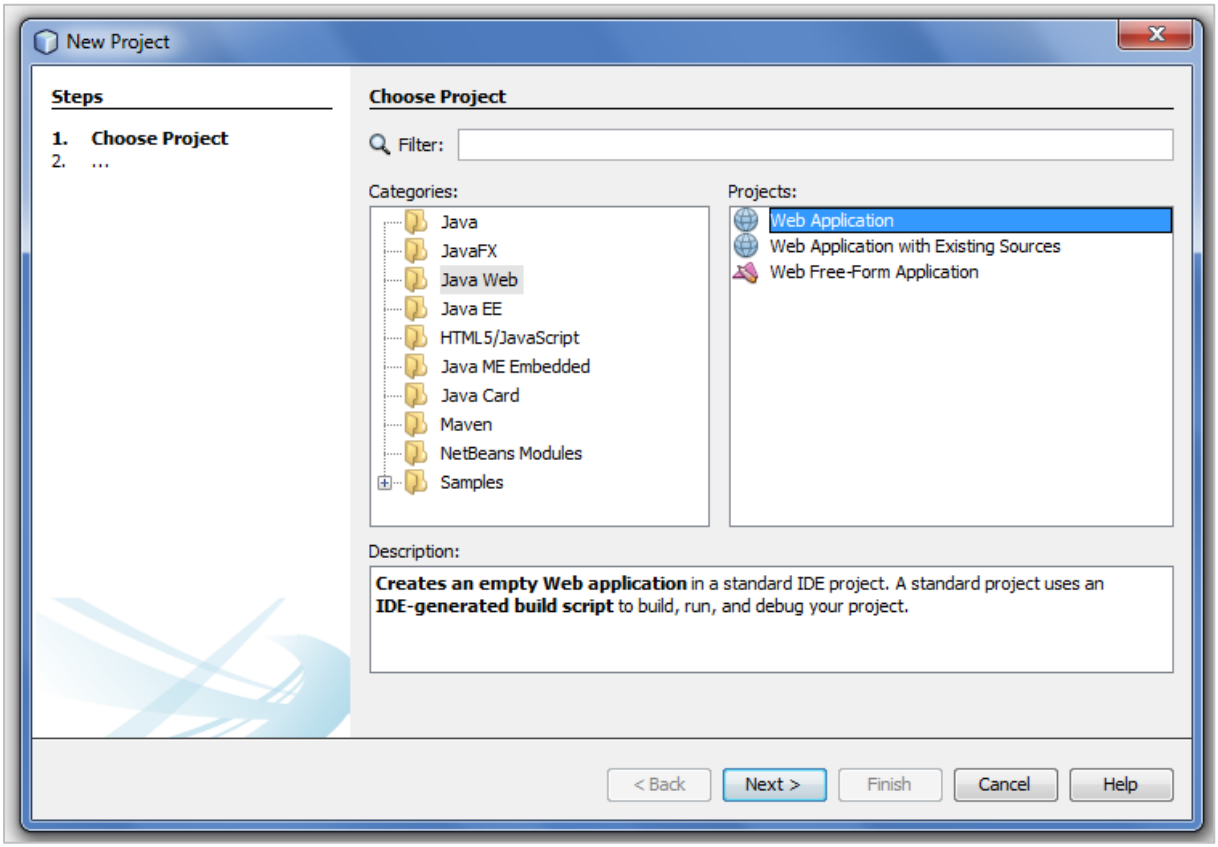


Figura 3.3 Creación del proyecto *Web*

Luego, se muestra un panel que permite nombrar y seleccionar el directorio raíz donde se almacenarán todos los archivos de configuración del servicio *Web*. Se le nombra "***oai2***" para hacer referencia a la versión del protocolo *OAI-PMH* que va a hacer utilizado. Una vez definido estos aspectos, se continúa con la configuración del servicio *Web* con la opción "***Next***". Después, se selecciona el servidor que posibilitará el arranque del servicio *Web*. Este cumple con la función de intermediario comunicativo entre el servicio *Web* y el SGBD *J-ISIS*. En la investigación se utiliza el servidor *GlassFish* v4.1.1. Se continúa con la configuración del servicio *Web* con la opción "***Next***".

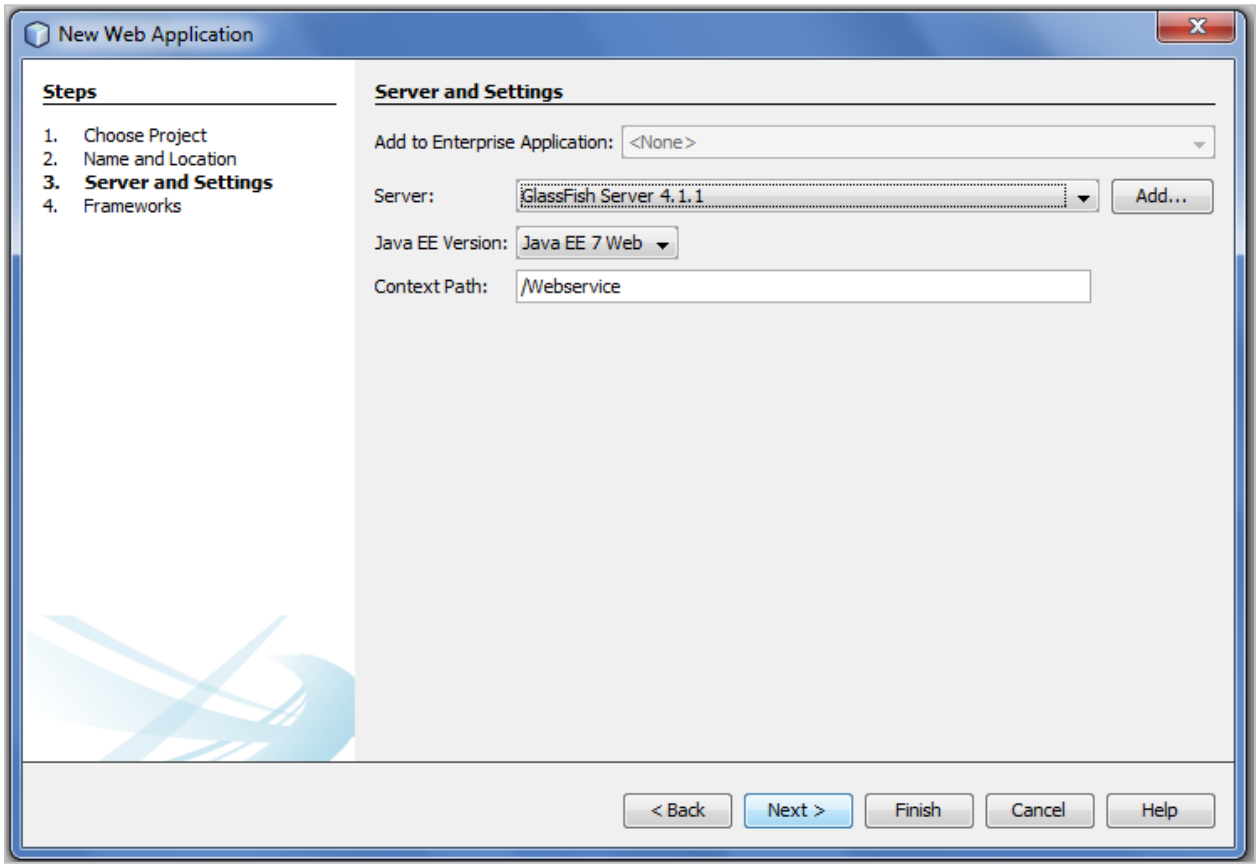


Figura 3.4 Selección del servidor *Web*

Por último, se muestra un panel que posibilita la selección de tecnologías que funcionarán de manera simultánea con el servicio *Web*. Para este caso se elige la tecnología "***Spring Web MVC***" que automáticamente suministrará todas sus librerías posibles. Luego se finaliza la configuración del servicio en la opción "***Finish***".

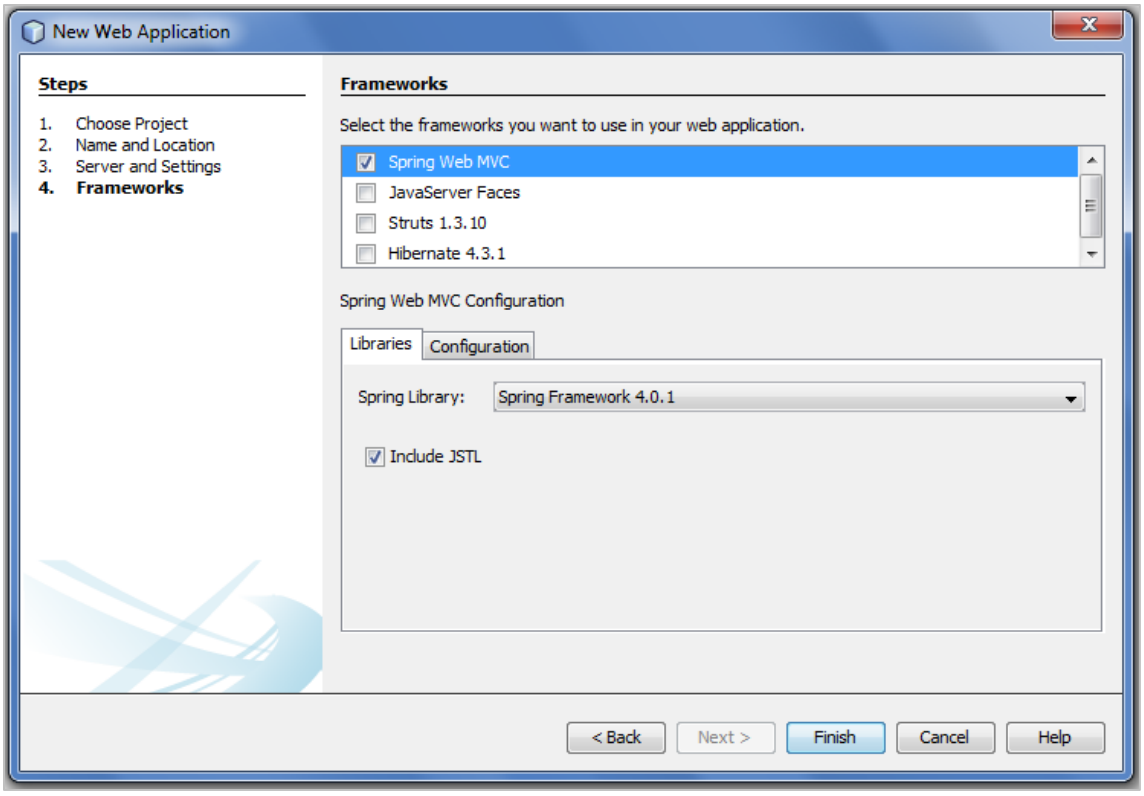


Figura 3.5 Selección del *Framework Spring MVC*

Una vez terminada la creación del servicio *Web*, es momento de proseguir con su configuración interna.

Paso #2: Configuración interna del proyecto *Web*

Ya creado el servicio *Web*, el *IDE NetBeans* prepara el espacio de trabajo (***workspace***, en inglés) para realizar las configuraciones internas deseadas. En el panel izquierdo se encuentran todas las librerías que ofrece el *Framework Spring MVC* y varias carpetas que contienen archivos de configuración del servicio.

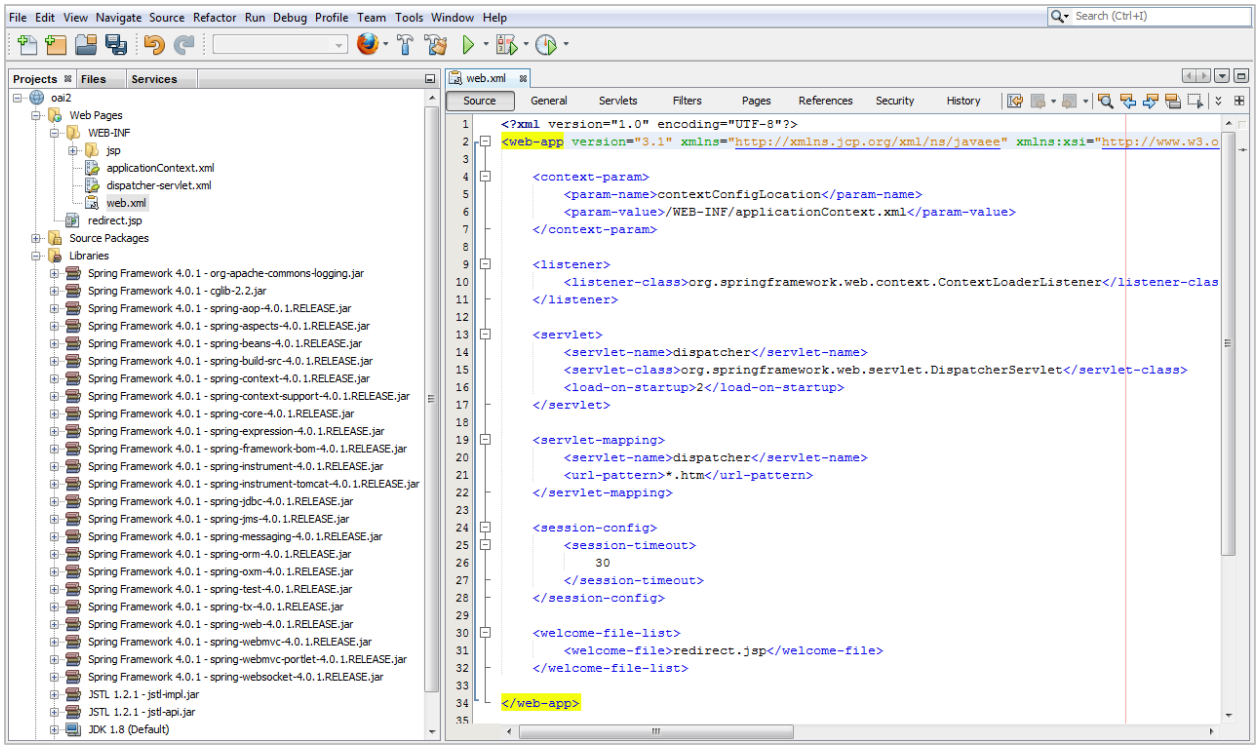


Figura 3.6 Espacio de trabajo inicial

Configuración del archivo *web.xml*

Dentro de la carpeta "***WEB-INF***", existe una carpeta y un archivo de configuración nombrado "***jsp***" y "***applicationContext***", respectivamente. En este caso, no serán necesarios ya que son utilizados para otros aspectos de configuración de servicio *Web*, por lo tanto, pueden ser eliminados sin problemas.

Es de vital importancia la configuración y utilización del archivo "***web.xml***" ya que contiene información sobre la estructura y las dependencias del servicio *Web*. Este, para que pueda ejecutarse, deben declararse varias etiquetas de configuración en el archivo "***web.xml***" como se muestra en la siguiente figura 3.7.

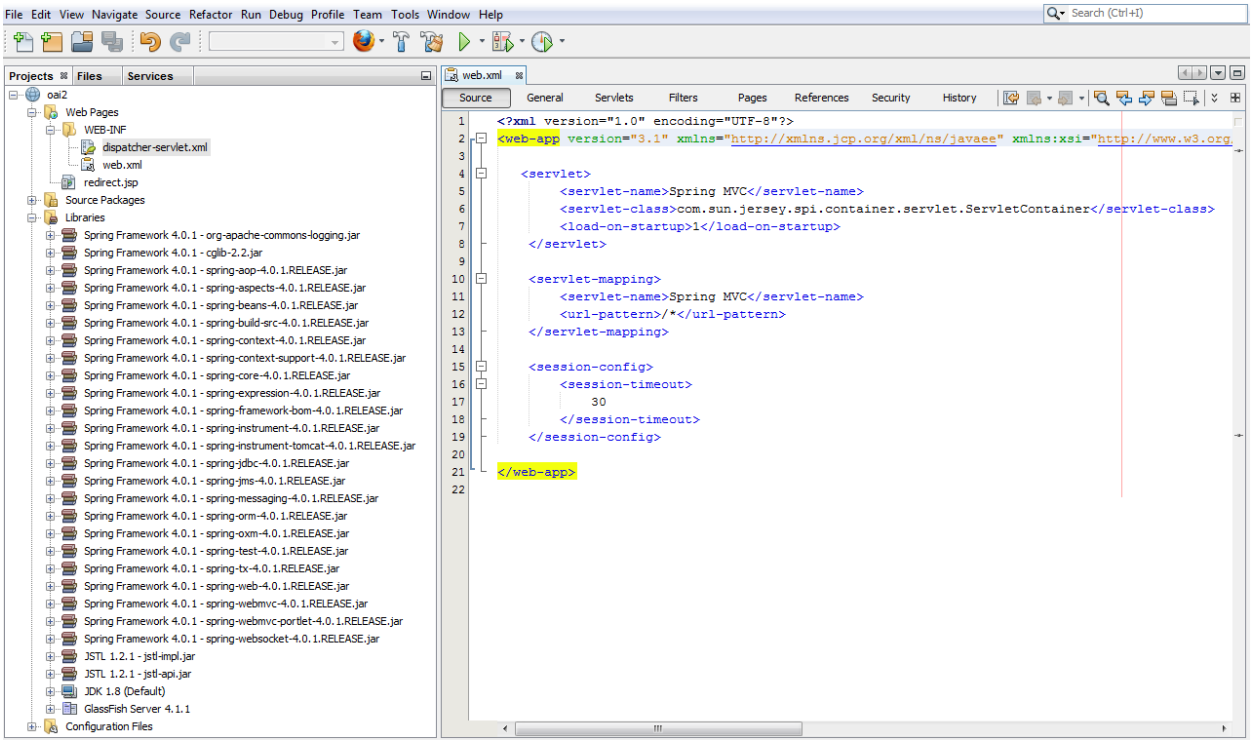


Figura 3.7 Configuración del archivo *web.xml*

La etiqueta "*<servlet>"* se utiliza para indicar el comienzo de la configuración del servidor. Dentro de ella se encuentra las etiquetas:

- *"<servlet-name>"*: Define el nombre canónico del servicio *Web*.

- "*<servlet-class>*": Llama a los métodos de una de las clases ya compiladas de la librería ***jersey-bundle.1-19.jar***. En este caso la clase se denomina "***ServletContainer.class***".

- **"***<load-on-startup>*": Indica el orden en que deben cargarse los servicios *Web.* En este caso toma valor "1", ya que solo desarrolla un solo servicio.

La etiqueta "*<servlet-mapping>*" se utiliza para definir los patrones *URL*. Dentro de ella se encuentran las etiquetas:

- "*<servlet-name>*": (Etiqueta requerida) Define El nombre del servicio al que está asignando un patrón de *URL*. Este nombre corresponde al nombre asignado en la configuración del *<servlet>*.

- "*<url-pattern>*": (Etiqueta requerida) Describe un patrón utilizado para resolver *URL*. La parte de la URL después de *http://host:puerto + Nombre_servicio_Web* se compara con la *<url-pattern>*. Si los patrones coinciden, se llamará al servicio *Web* mapeado en este elemento.

La etiqueta <*session-config*> Define los atributos de sesión del servicio *Web*. Dentro de ella se encuentran las etiquetas:

- <*session-timeout*>: (Etiqueta opcional) Define, en minutos, el tiempo de caducidad del servicio *Web*

Para mayor información referente a las etiquetas de configuración del archivo *web.*xml, diríjase a la dirección:

https://docs.oracle.com/cd/E21764_01/web.1111/e13712/web_xml.htm#WBAPP502

Incorporación de la librería *jersey-bundle.1-19.jar*

Para que esta configuración del archivo "***web.xml***" funcione correctamente, se debe añadir la librería ***jersey-bundle.1-19.jar*** disponible en la dirección *https://jersey.java.net/download.html*. Es recomendable crear un directorio (carpeta) con nombre personalizado dentro del proyecto *Web* creado para así evitar desorden en las dependencias de librerías que se necesiten. En este caso se crea la carpeta "***libs***". Esta librería permite, además, la elaboración de páginas *Web* con formato *XML* debido a las anotaciones que brinda.

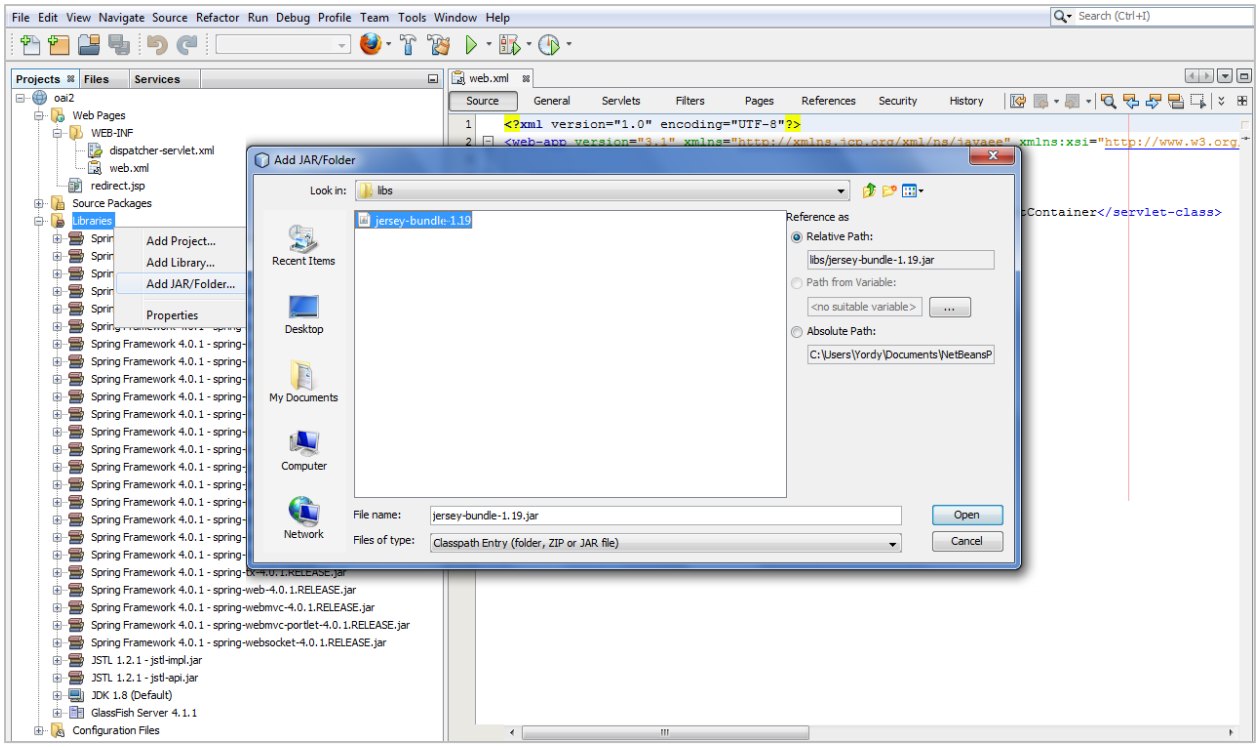


Figura 3.8 Incorporación de la librería *jersey-bundle.1-19.jar*

Finalmente, con la configuración del archivo "***web.xml***" y la incorporación de la librería ***jersey-bundle.1-19.jar***, se ha configurado internamente el servicio *Web*.

Paso #3 Creación del servicio *Web REST*

La Transferencia de Estado Representacional (*REST*, *Representational State Transfer*, por sus siglas en inglés) es un tipo de servicio *Web* que se utiliza para capturar consultas que se

realicen en el navegador. Con esta característica, se controlan las peticiones que hagan los usuarios mediante el método *GET*.

Configurado los archivos internos del servicio *Web* "***oai2***", es momento de añadir el componente *REST*. Para ello, en el panel izquierdo del espacio de trabajo, se selecciona la opción "***Source Package***", y se crea un paquete con nombre personalizado. En este caso se crea con el nombre "***webservice.controller***".

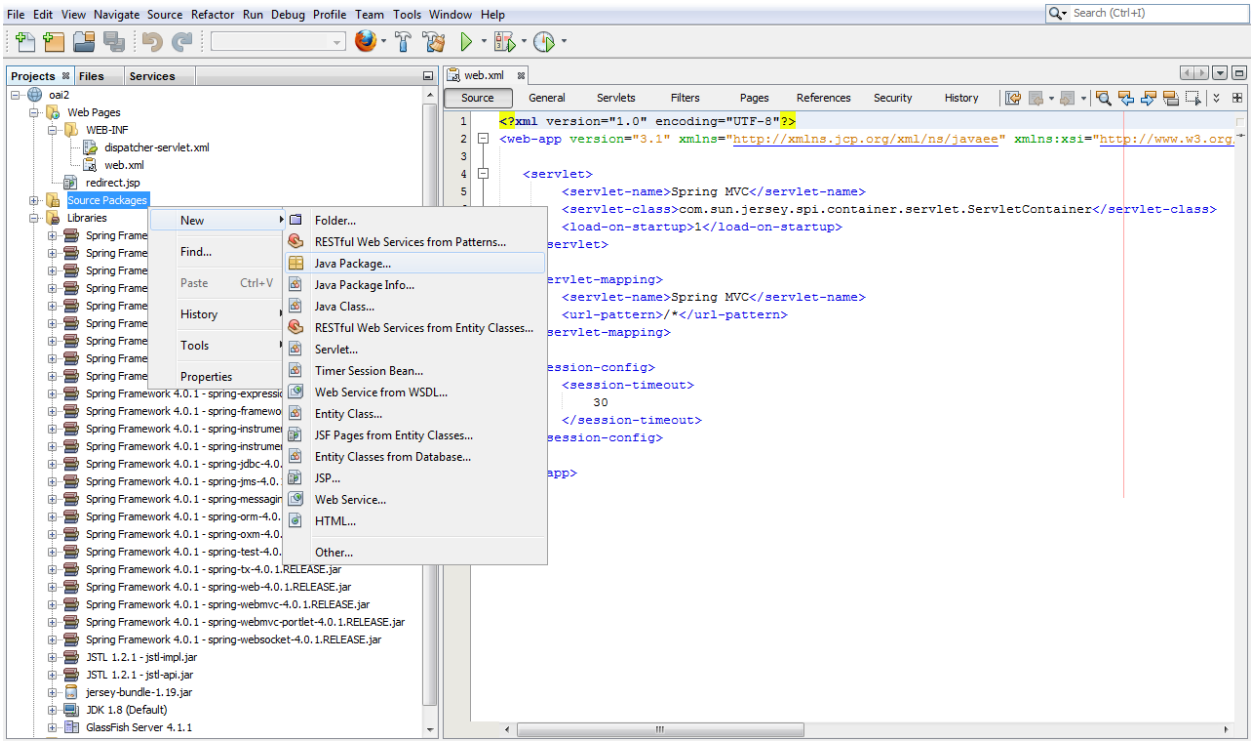


Figura 3.9 Creación del paquete *REST*

Sobre el paquete creado previamente, se crea el componente en la opción "***RESTful Web from Patterns***".

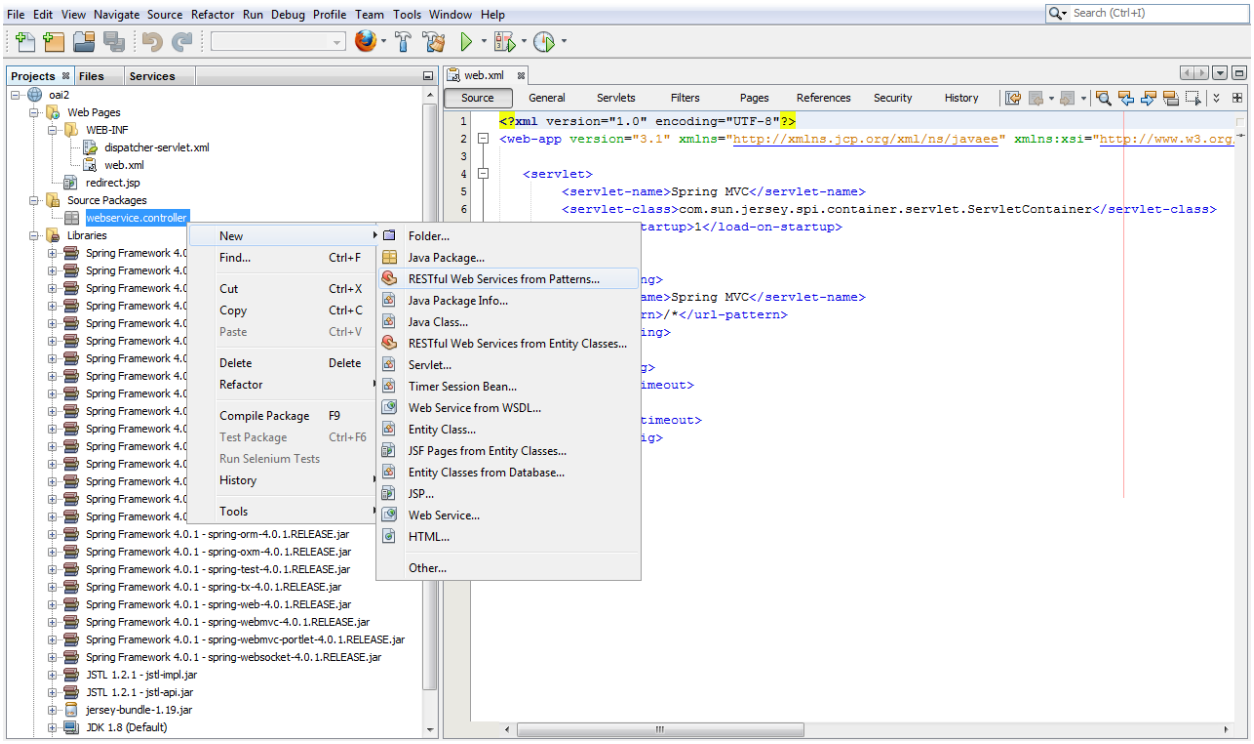


Figura 3.10 Creación del componente *REST*

Seguidamente, se selecciona la opción "***Conatiner-Item***" para la creación de la clase controladora. Esta es la encargada de controlar todos los eventos que se realicen por la *URL* y devolver una página *Web* con estructura *XML*. Se continúa con la configuración del servicio *Web* con la opción "***Next***". Acto seguido, aparece un panel para definir el nombre de la clase que será convertida a *XML* y de la clase controladora. A estas se le llama "***Identify***" "***Controller***". Esta configuración se realiza en los campos "***Class Name***" y "***Container Class Name***", respectivamente. Finalmente, se termina la configuración del servicio con la opción "***Finish***".

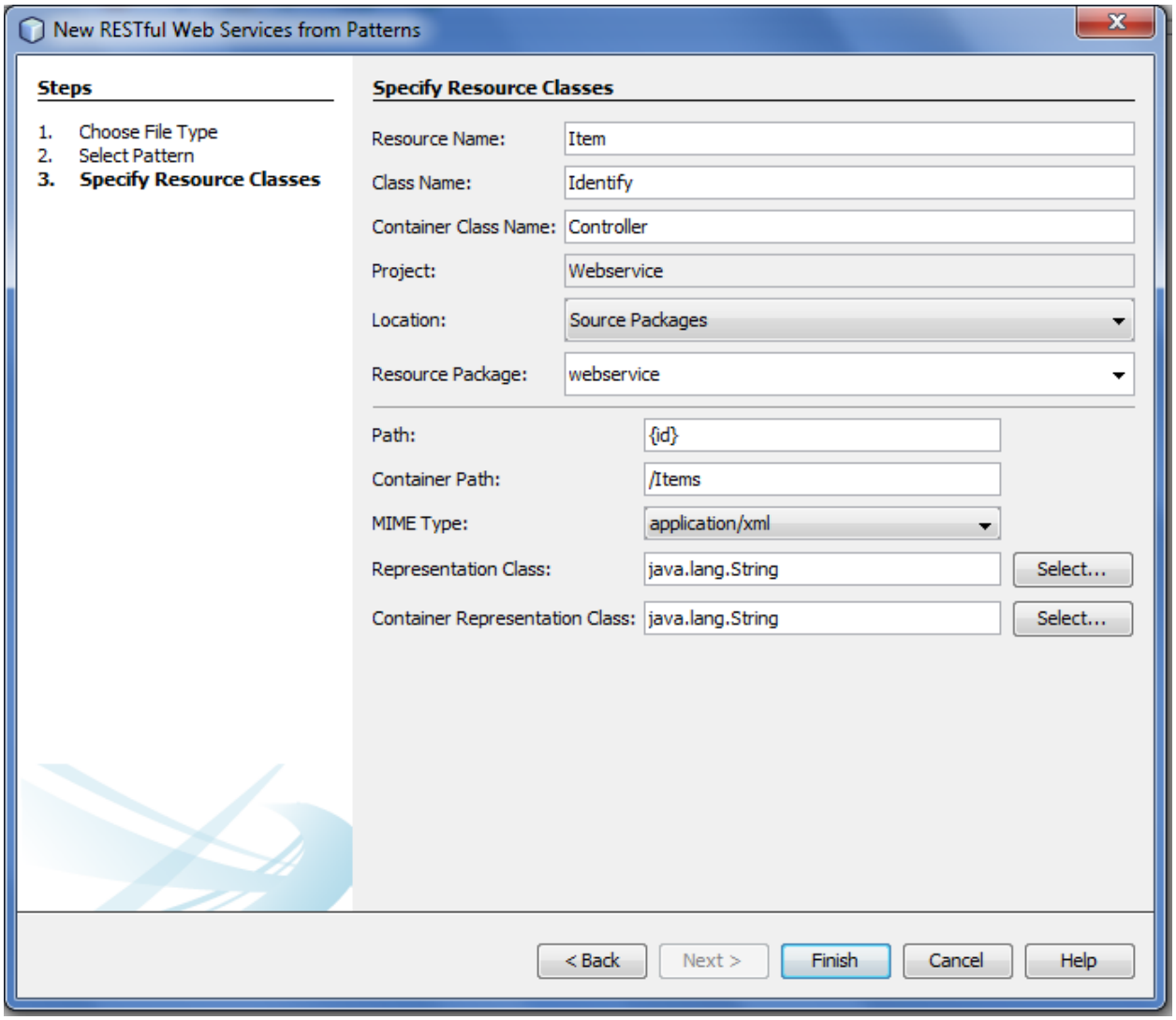


Figura 3.11 Definición de las clases del componente *REST*

Paso #4 Configuración de la clase *Identify.java*

Para elaborar las páginas con estructura *XML* es necesario crear clases de *Java*. En este caso, estas instancias coinciden con las 6 peticiones que admite el protocolo *OAI-PMH*. Como ejemplo, se toma a la solicitud ***Identify***, encargada de notificar las propiedades que presenta el SGBD *J-ISIS*. Deben ser configuradas con las siguientes anotaciones *JAXB* (*Java Architecture for XML Binding, en inglés*).

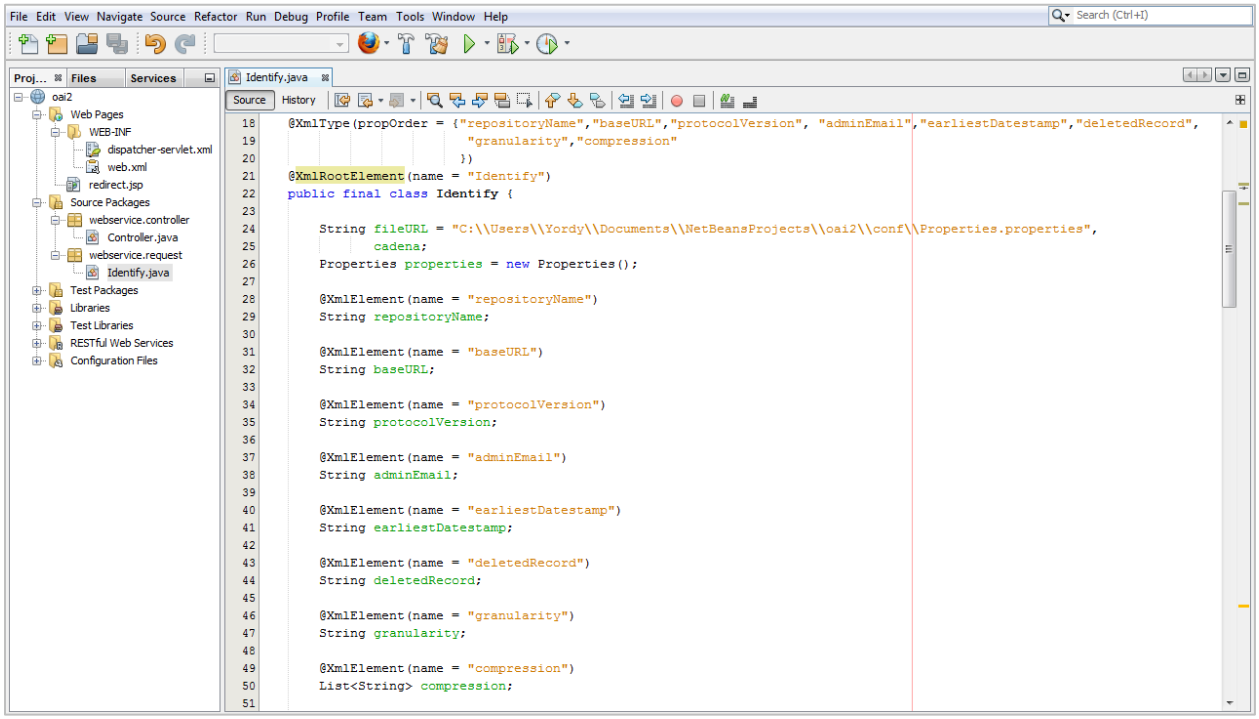


Figura 3.12 Configuración de la clase *Identify*

Para tener una mejor organización, se subdivide en paquetes las clases que hacen analogía a las peticiones *OAI-PMH*. En el paquete "w***ebservice.controller***" se mantiene la clase controladora "***Controller***"

- ***@XmlRootElement* (atributos)**: Esta anotación es importante y obligatoria ya que es la encargada de convertir la clase creada en una página *Web* con estructura *XML* .Como todo formato en *XML* posee etiquetas de apertura y cierre (ver XML arriba), se puede definir un nombre de manera personalizada la etiqueta de apertura con el atributo "***name***".

- ***@XmlType* (atributos)**: Esta anotación define el orden que van a tomar los elementos.

- ***@XmlElement* (atributos)**: Esta anotación es importante pues convierte los atributos de la clase creada en elementos de la página *Web* con formato *XML*.). Posee de igual forma, la propiedad "***name***" para personalizar su nombre.

- **@*XmlAtribute* (atributos)**: esta anotación se utiliza para colocar atributos en las etiquetas de una página *Web* con estructura *XML*. Posee de igual manera, la propiedad "***name***" para personalizar su nombre. La clase ***Identify*** no posee atributos dentro de sus etiquetas.

Esta sencilla configuración se realiza para las restantes peticiones de *OAI-PMH*, o sea, por cada solicitud se deben crear una clase: *ListSets.java*, *ListIdentifiers.java*, *ListMetadataFormats.java*, *GetRecod.java* y *ListRecords.java*

Paso #5 Configuración de la clase *Controller.java*

Con la configuración de la clase controladora, se administran todos los eventos que se hagan en el navegador *Web* por parte del usuario a través de la *URL*. Para ello, se hace uso de las anotaciones de *JAX-RS* (***Java API for RESTful Web Services***) predefinidas por la herramienta *NetBeans*. Ya aquí se evidencia la participación del protocolo *OAI-PMH*.

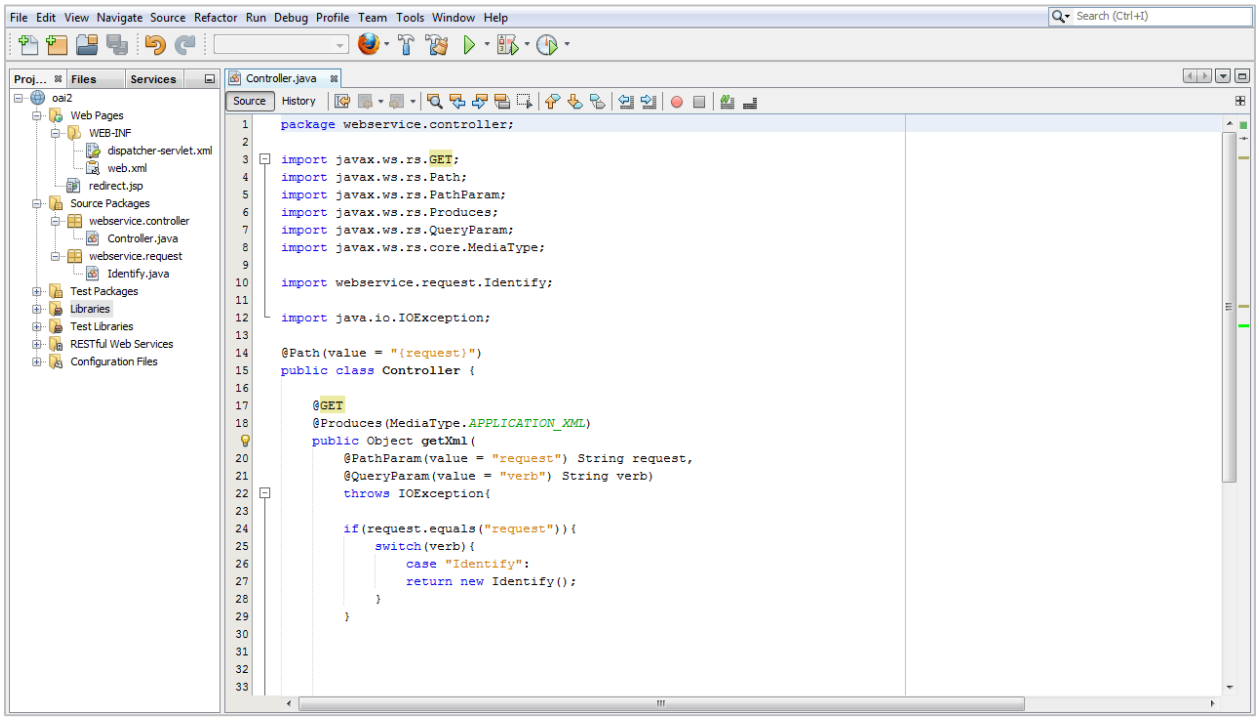


Figura 3.13 Configuración de la clase controladora

- **@*Path* (atributos)**: esta anotación es utilizada para capturar los parámetros que se escriben en la barra *URL* del navegador *Web*. Se coloca por encima del nombre de la clase "***Controller***" creada previamente. El atributo "***value***" es la encargado de almacenar estos parámetros usando la simbología "**{}**".

- **@*GET***: esta anotación es clave, pues indica que los parámetros que escriba el usuario siempre serán *URL* del navegador. No posee atributos. Se coloca por encima del

método encargado de gestionar los eventos que se realicen por la *URL*. En este caso el método es "***getXml()***"

- **@*Produces* (atributos)**: esta anotación es la que permite no solo construir páginas en formato *XML* sino con otros tipos de formatos (*JSON*, texto plano, etc…). Con el atributo "***MediaType.APPLICATION_XML***" se obtiene el resultado esperado. Se coloca por encima del método encargado de gestionar los eventos que se realicen por la *URL*. En este caso "***getXml()***".

- **@*PathParam* (atributos)**: esta notación se encarga de capturar el valor que adquiere la anotación @*Path* con el atributo "***value***". Para ello es necesario crear un tipo de dato "***String***" que permita almacenar ese valor de la anotación @*Path*. En este caso es "***String request***". Nótese que ambos atributos de las anotaciones son de igual nombre. No acepta la simbología "***?***": necesaria para poder elaborar una consulta *OAI-PMH*. Se coloca como parámetro dentro del método "***getXml()***".

- **@*QueryParam* (atributos)**: esta notación utiliza la simbología "***?***": necesaria para elaborar una petición *OAI-PMH*. Se encarga de capturar todos los parámetros después de esta simbología con el atributo "***value***". Para ello es necesario crear un tipo de dato "***String***" que permita almacenar ese valor de la anotación. Se coloca con parámetro dentro del método "***getXml()***".

Para comprobar el funcionamiento de la clase controladora, se debe ejecutar el servicio *Web*, en la opción "***Run***".

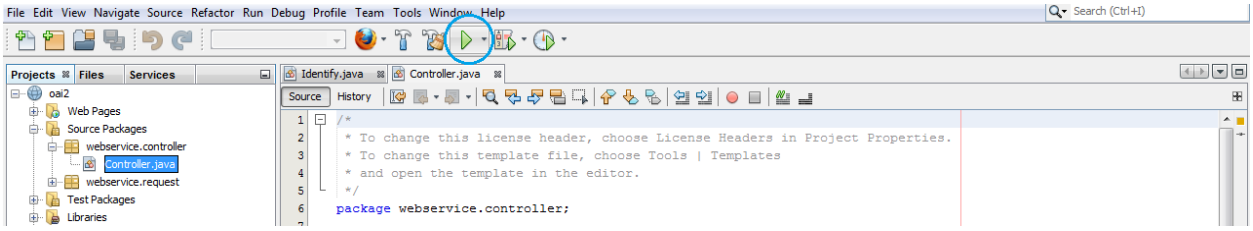


Figura 3.14 Ejecución del servicio *Web*

Automáticamente, debe ejecutarse el navegador *Web* que se encuentre instalado en el ordenador. Nótese la *URL* de la barra de navegación. Se tiene el nombre del servicio *Web* creado y al final de la *URL* se espera una petición a realizar. Es aquí donde se formulan las peticiones que brinda el protocolo *OAI-PMH* para la consultoría de información.

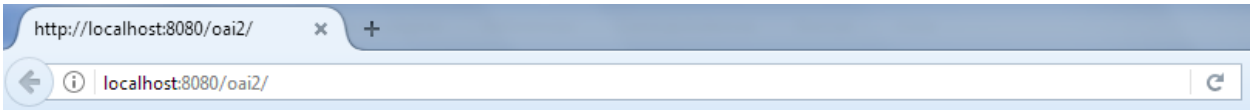


Figura 3.15 Servicio *Web* en tiempo de ejecución en espera de una petición *OAI-PMH*

Después de "***oai2/***" se crean las consultas de *OAI-PMH*

- petición ***Identify*** (devuelve las propiedades que posee el sistema bibliotecario que se le consulta información bibliográfica):

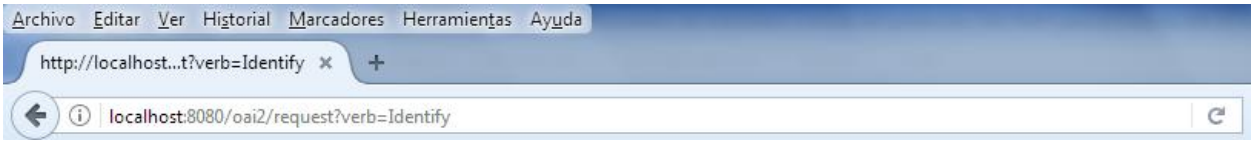


Figura 3.16 Petición *Identify* en tiempo de ejecución

- petición ***ListSets*** (devuelve el listado de agrupaciones de materiales bibliográficos que se encuentran almacenados en el sistema bibliotecario):

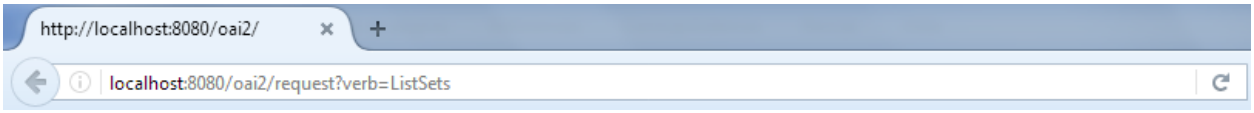


Figura 3.17 Petición *ListSets* en tiempo de ejecución

- petición ***ListMetadataFormtas*** (devuelve el listado de formato de metadatos con que el sistema bibliotecario cataloga sus materiales):

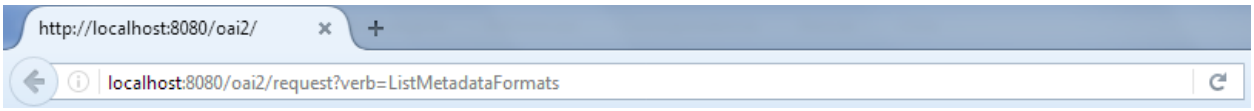


Figura 3.18 Petición *ListMetadataFormats* en tiempo de ejecución

- petición ***ListIdentifiers*** utilizando el argumento ***metadataPrefix*** con valor "***oai_dc***" (devuelve todos los identificadores de los materiales que se encuentran catalogados bajo el formato de metadato *Dublin Core* Simple):

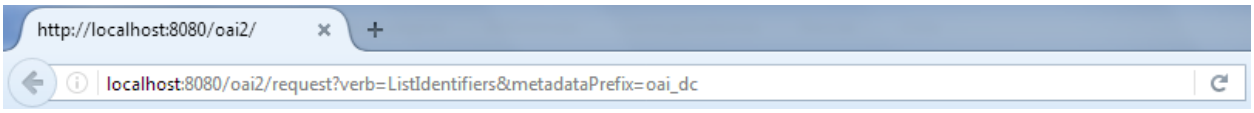


Figura 3.19 Petición *ListIdentifiers* en tiempo de ejecución

- petición ***GetRecord***: utilizando los argumentos ***identifier*** y ***metadataPrefix*** con valor "***oai_dc***" (devuelve un material bibliográfico en específico):

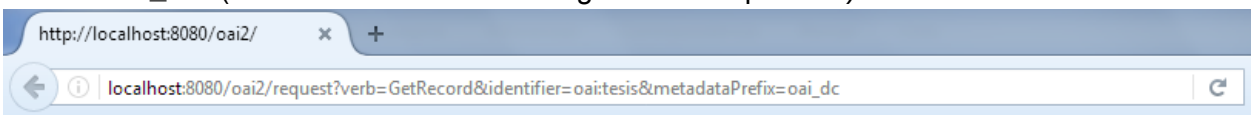


Figura 3.20 Petición *GetRecord* en tiempo de ejecución

- petición ***ListRecords*** utilizando el argumento ***metadataPrefix*** con valor "***oai_dc***" (devuelve el listado de los materiales bibliográficos almacenados en el sistema bibliotecario):

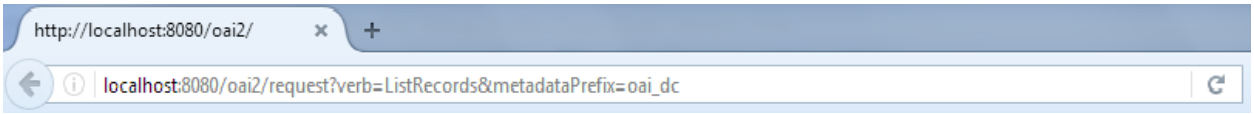


Figura 3.21 Petición *ListRecords* en tiempo de ejecución

3.5 Estrategias de pruebas

Las estrategias de prueba de *software* proporcionan una guía que describe los pasos que deben realizarse como parte de la prueba, cuándo se planean y se llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recursos se requerirán. Por tanto, cualquier estrategia de prueba debe incorporar la planificación, diseño, ejecución, recolección y evaluación de los casos de prueba (Pressman, 2010).

3.3.1 Tipos de pruebas

Prueba de Carga

La prueba de carga se diseña para comprobar el rendimiento del *software* en tiempo de corrida, dentro del contexto de un sistema integrado. Esta ocurre a lo largo de todos los pasos del proceso de prueba. Incluso en el nivel de unidad, puede accederse al rendimiento de un módulo individual conforme se realizan las pruebas. Sin embargo, no es sino hasta que todos los elementos del sistema están plenamente integrados cuando puede determinarse el verdadero rendimiento de un sistema (Pressman, 2010). Se hace uso de este tipo de prueba para valorar el número esperado de usuarios concurrentes utilizando la aplicación en tiempo de ejecución.

Prueba de *Stress*

La prueba de *stress* va enfocada a evaluar cómo el sistema responde bajo condiciones anormales. Mayormente, mide la fiabilidad del sistema estableciendo términos estadísticos

como la probabilidad de operación libre de fallos. Se le aplica este tipo de prueba a la aplicación para que sea capaz de responder de manera adecuada ante cualquier fallo técnico que pueda presentar el SGBD *J-ISIS* en un momento determinado.

Herramienta informática utilizada para aplicar las pruebas de Carga y *Stress*

La aplicación *Apache JMeter* ™ es un herramienta de código abierto diseñada para realizar pruebas de comportamiento y rendimiento sobre aplicaciones *Web*. También se puede utilizar para simular la carga de un servidor, grupo de servidores, red u objeto para probar su resistencia o para analizar el rendimiento general bajo diferentes tipos de carga (Fundation, 2017). Esta herramienta se encuentra disponible en su sitio *Web* oficial en la dirección *http://jmeter.apache.org/download_jmeter.cgi*

3.3.2 Métodos de pruebas

Método de prueba Caja Blanca

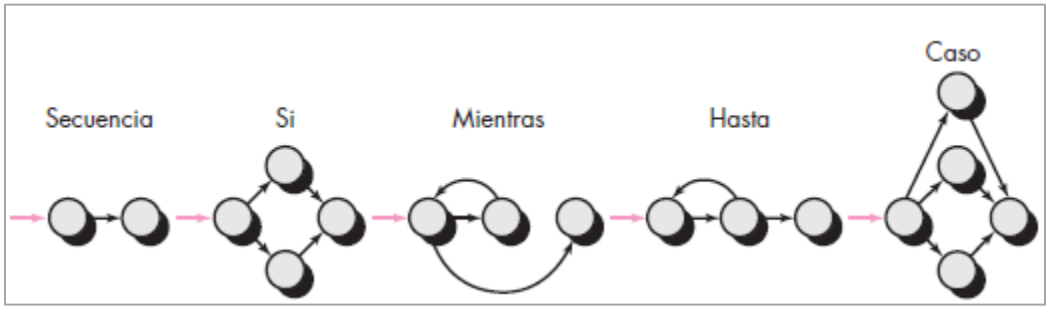
La *prueba de Caja Blanca*, en ocasiones llamada *prueba de Caja de Vidrio*, es una filosofía de diseño de casos de prueba que usa la estructura de control descrita como parte del diseño a nivel de componentes para derivar casos de prueba. Al usar los métodos de prueba de Caja Blanca, puede derivar casos de prueba que (Pressman, 2010):

- Garanticen que todas las rutas independientes dentro de un módulo se revisaron al menos una vez.
- Revisen todas las decisiones lógicas en sus lados verdadero y falso.
- Ejecuten todos los ciclos en sus fronteras y dentro de sus fronteras operativas.
- Revisen estructuras de datos internas para garantizar su validez.

Para utilizar el método de prueba de Caja Blanca, se necesitan aplicar varias técnicas. En la presente investigación se hace uso de la técnica de Ruta Básica.

1. Técnicas de Ruta Básica

La *Ruta* o *Trayectoria básica* es una técnica de prueba de Caja Blanca propuesta por primera vez por el ingeniero Thomas J. McCabe. El método de ruta básica permite al diseñador de casos de prueba derivar una medida de complejidad lógica de un diseño de procedimiento y usar esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de prueba derivados para revisar el conjunto básico tienen garantía para ejecutar todo enunciado en el programa, al menos una vez durante la prueba (Pressman, 2010).



Fuente: Libro "*Ingeniería del software, un enfoque práctico*", Rogger S. Pressman

Figura 3.22 Esquema para representar el código fuente de una aplicación

La figura anterior muestra un modelo que debe tenerse en cuenta para comprender la codificación de una aplicación. Cada círculo representa una o más son declaraciones de código fuente. Para el caso del bloque "Secuencia" (***Sequence***, en inglés), hace referencia a declaraciones de variables o instancias simples. Los bloques "Si y Caso" (estructuras condicionales ***If*** y ***Case*** para el lenguaje de programación *Java*, respectivamente) indica la presencia de bifurcaciones que puede presentar el código fuente. Finalmente, para el bloque "Mientras y Hasta" (estructuras repetitivas ***While*** y ***For*** para el lenguaje de programación *Java*, respectivamente) señala los ciclos (***loop***, en inglés) presentes en la codificación.

- Variante ***Notación de gráfico o grafo de flujo***
  Antes de considerar el método de ruta básica, debe introducirse una notación simple para la representación del flujo de control, llamado *gráfico de flujo* (o *gráfico de programa*). El gráfico de flujo muestra el flujo de control lógico que usa la notación ilustrada en la figura anterior.

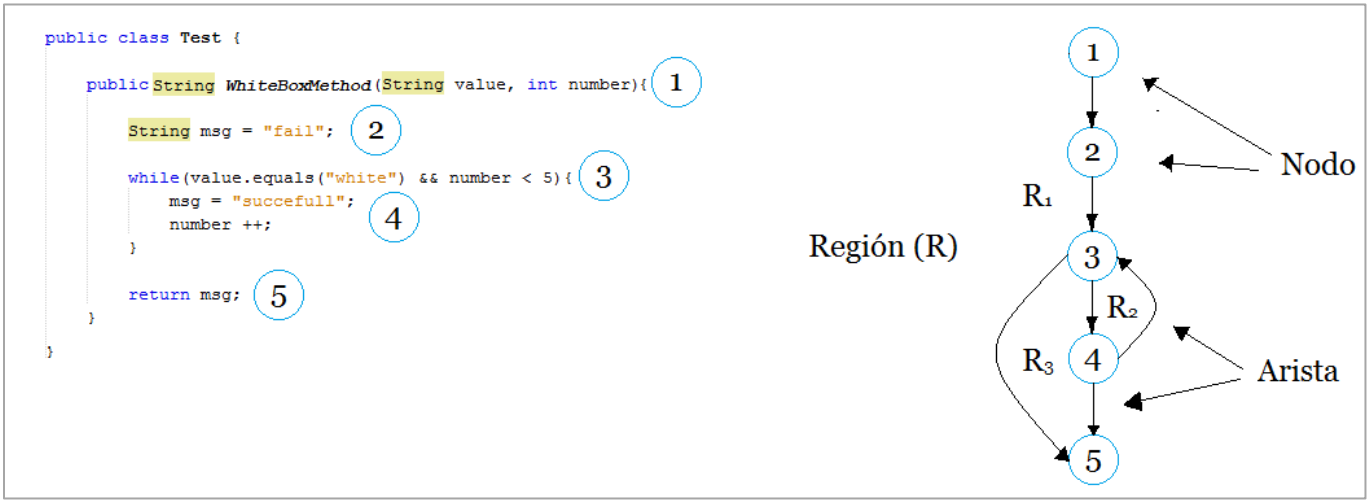


Figura 3.23 Grafo de flujo

Esta variante es necesaria ya que proporcionada una visión estructural del código fuente y sirve como base para la variante que se verá a continuación. Cada círculo, llamado *nodo de gráfico de flujo*, representa uno o más enunciados de procedimiento. Como se puede observar, el nodo 1 y 2 hace referencia al bloque "Secuencia". Las flechas en el gráfico de flujo, llamadas *aristas* o *enlaces*, representan flujo de control. Una arista debe terminar en un nodo, incluso si el nodo no representa algún enunciado de procedimiento. Las áreas acotadas por aristas y nodos se llaman *regiones*. Cuando se cuentan las regiones, el área afuera del gráfico se incluye como región (Pressman, 2010).

- Variante ***Rutas independientes***

Una *ruta independiente* es cualquiera que introduce al menos un nuevo conjunto de enunciados de procesamiento o una nueva condición en el programa. Cuando se establece como un gráfico de flujo, una ruta independiente debe moverse a lo largo de al menos una arista que no se haya recorrido antes de definir la ruta (Pressman, 2010). Por ejemplo, un conjunto de rutas independientes para el gráfico de flujo que se ilustra en la figura anterior es:

- ruta 1: 1-2-3-5
- ruta 2: 1-2-3-4-5
- ruta 3: 1-2-((3-4-3), puede repetirse *n*-veces hasta que se cumpla la condición)-5

Como se puede observar, existen 3 rutas independientes en el código y todas concluyen en el nodo 5. Este es el principal objetivo de este método de prueba, que todas las rutas terminen en un nodo único.

Pero, detectar las rutas independientes se puede complejizar cuando el código fuente es demasiado engorroso. Es por ello que se utiliza la variante "complejidad ciclomática"

- Variante ***Complejidad ciclomática***

  La *complejidad ciclomática* V(G) es una medición de *software* que proporciona una evaluación cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto del método de prueba de la ruta básica, el valor calculado por la complejidad ciclomática define el número de rutas independientes del conjunto básico de un programa y le brinda una cota superior para el número de pruebas que debe realizar a fin de asegurar que todos los enunciados se ejecutaron al menos una vez (Pressman, 2010).

  La complejidad se calcula de varias formas (Pressman, 2010):

  - El número de regiones (R) del gráfico de flujo del código fuente de la aplicación.
  - Por la formula V(G) = *E* – *N* + 2, donde *E* es el la cantidad de aristas y *N* el número de nodos del grafo.

  Para el ejemplo anterior, en todos los casos los resultados son 3.

3.6 Resultados de la pruebas realizadas a la aplicación

Luego de analizar los fundamentos teóricos acerca de las estrategias de pruebas y sus diferentes tipologías y sobre las herramientas que se utilizan para aplicarlas, llega el momento de arrojar sus resultados.

3.6.1 Resultados de las pruebas de Caja Blanca

Como se estudió previamente, una de las técnicas del método de prueba de Caja Blanca es la de la ruta independiente. Se determinó aplicarla en el método "***fileReader***" de la clase *Identify.java.*

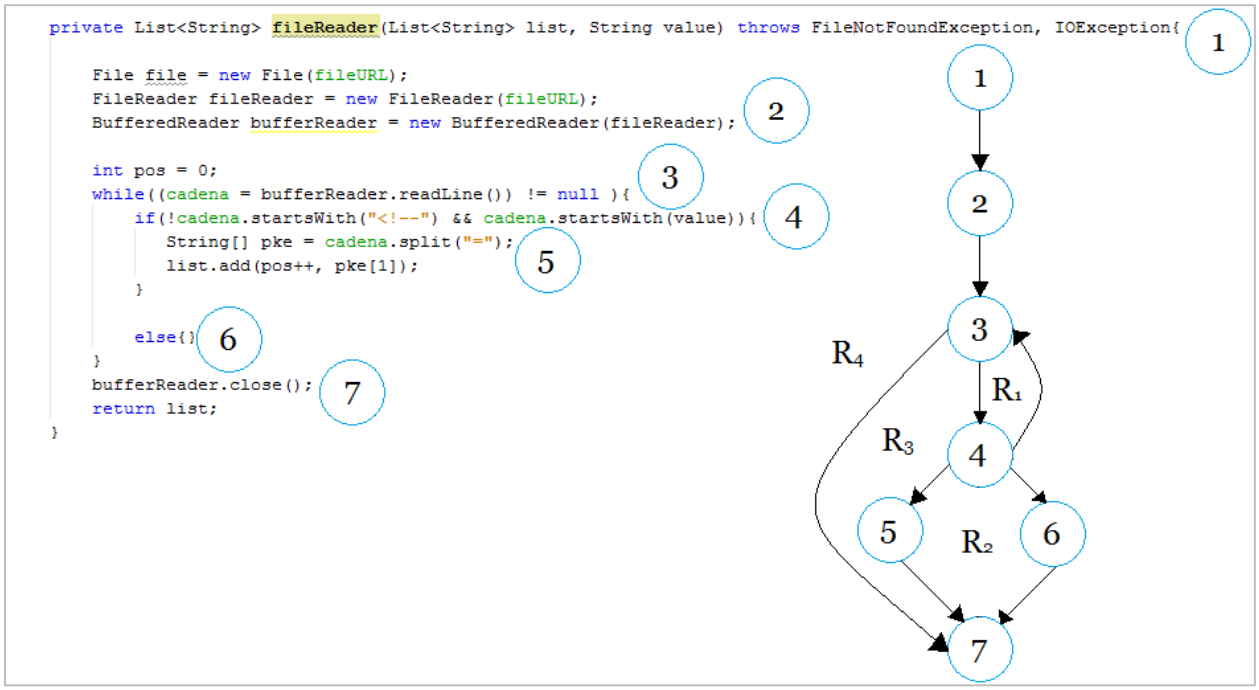


Figura 3.24 Técnica de la ruta independiente al método "*fileReader*" de la clase *Identify.java*

Con la ruta independiente determinada, se aplica una de las tres formas para calcular la complejidad dicromática, se utilizó la fórmula V(G)= E - N + 2, para la cual se obtuvo 9 aristas y 7 nodos, por lo tanto: V (G)= 9 - 7 + 2, quedando V(G) = 4.

3.6.2 Resultados de las pruebas de Carga y *Stress*

Los resultados obtenidos en las pruebas de Carga y *Stress* se muestran a continuación. El eje vertical señala los segundos que demora la respuesta de cada petición *OAI-PMH* y el eje horizontal indica la cantidad de usuarios que pueden estar realizándola en ese instante de tiempo en segundo.

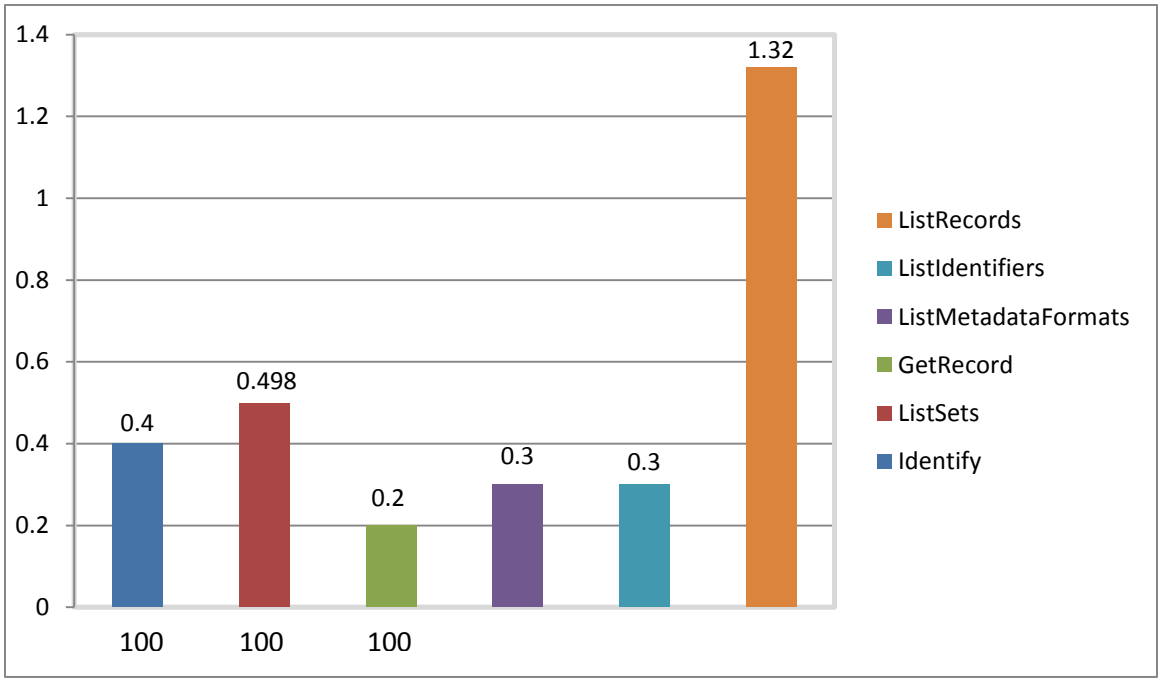


Figura 3.25 Resultados de las pruebas de Carga y *Stress*

Los resultados arrojan que la petición ***ListRecords*** es la que más se demora en dar respuesta, pero su valor no excede de los 2 segundos, un aspecto muy importante en panoramas de tiempo de respuesta de los sistemas *Web*.

Conclusiones

En el capítulo que concluye se modeló el diagrama de componentes, con el cual se obtuvo una panorámica de elementos que intervienen en el funcionamiento de la aplicación. Se elaboró el diagrama de despliegue el cual posibilitó la identificación de los artefactos físicos que necesita la aplicación para funcionar (ordenador, servidor *Web* y el SGBD *J-ISIS*). Luego, se realizó un resumen referente al desarrollo de la aplicación con el objetivo de que cualquier personal que haga uso de la presente investigación obtenga conocimiento básico de cómo implementarlo. Finalmente se le realizan estrategias de pruebas para comprobar el comportamiento y rendimiento de la aplicación en tiempo de ejecución. Los resultados obtenidos fueron satisfactorios.

Conclusiones generales

Utilizando como base el marco teórico conceptual y a partir del estudio realizado sobre las características estructurales y funcionalidades de los protocolo de búsqueda e intercambio de información bibliográfica y del SGBD *J-ISIS*, se desarrolló un componente informático que permitió intercambiar la información bibliográfica del Sistema ABCD con otros sistemas. Esto trajo como ventaja, la promoción de los resultados académicos de la comunidad universitaria (tesis, artículos científicos, etc…) y la divulgación de su información para que otros sistemas, tanto internos como externos, pudiesen actualizarse cada vez que fuese necesario. Para el correcto intercambio de información se utilizó el formato de metadatos *Dublin Core Simple*, el cual posibilitó la catalogación (organización y ordenamiento) y proyección, a través del ambiente *Web*, de dicha información.

Recomendaciones

- Desarrollar una interfaz *Web* para una mejor interacción entre el usuario y la aplicación desarrollada.

- Vincular el componente informático con Solar, para que los tiempos de respuestas a las búsquedas bibliográficas sean mucho más rápidos.

Referencias bibliográficas

Alegsa, Leandro. 2010. ALEGSA.com.ar. *Diccionario de Informática y Tecnlogía.* [En línea] 5 de Diciembre de 2010. http://www.alegsa.com.ar/Dic/ide.php.

ANSI/NISO. 2003. *Information Retrieval (Z39.50): Application Service Definition and Protocol Specification.* 4733 Bethesda Avenue, Suite 300, Bethesda, MD 20814. : NISO Press, 2003. pág. 276. 1041-5653.

Argueta, Miguel Ángel G Mejía. 2012. *La Interoperabilidad y el intercambio de metadatos en la Red.* - : Revista Digital Universitaria, 2012. pág. 11. 1067-6079.

Barrueco, José Manuel. 2009. *OAI-PMH: Protocolo para la transmisión de contenidos en Internet.* Valencia : Bibliotecas de Ciencias Sociales, 2009. 46010.

Bárzaga, Rubén Barreto y Ariste, Osmín Alejandro Velázquez. 2014. *Gestión semiautomática de metadatos bibliográficos desde corpus de texto en formato PDF.* Universidad de las Ciencias Informáticas. La Habana : Universidad de las Ciencias Informáticas, 2014. pág. 91, Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas. TD_07351_14.

Bukuyemskyy, Sergiy. 2017. Taller Digital. *Taller Digital.* [En línea] Taller Digital, 15 de Febrero de 2017. http://blog.eltallerdigital.com/2011/02/catalogos-bibliograficos-en-el-entorno-moderno-de-la-informacion-capitulo-2-formato-marc/.

CCM. 2008. CCM. *CCM.* [En línea] CCM Benchmark Group, 2008. http://es.ccm.net/download/descargar-28127-visual-paradigm-for-uml-enterprise-edition.

Dauphin, Jean-Claude. 2015. *J-ISIS Quick Tutorial.* s.l. : JCD, 2015.

Definicion.de. 2017. Definicion.de. *Definicion.de.* [En línea] 2017. http://definicion.de/informacion/.

DefinicionABC. 2010. DefinicionABC. *DefinicionABC.* [En línea] 2010. www.definicionabc.com/tecnologia/proveedor.php.

Delgado, Carlos Fco Marcos y Garrido, Alejandro Poyo. 2017. Scridb. *Scridb.* [En línea] Scridb, 2017. https://es.scribd.com/doc/189203494/Comparativa-Framework.

Digital, Guía. 2017. Guía Digital. *Guía Digital.* [En línea] Guía Digital de España, 2017. https://www.1and1.es/digitalguide/servidores/know-how/servidor-web-definicion-historia-y-programas/.

Dublin Core, Metadata Initiative. 2016. Dublin Core. *Dublin Core.* [En línea] Dublin Core Metadata Initiative, 2016. http://www.dublincore.org/.

Estrada Corona, Adrián. 2004. *Protocolo TCP/IP de Internet.* Vinculación de la Coordinación de Publicaciones Digitales de la DGSCA-UNAM. Mexico : Revista Digital Universitaria, 2004. pág. 7, Informe. 1067-6079.

FSF. 2017. Free Software Foundation,. *Free Software Foundation,.* [En línea] Free Software Foundation,, 2017. http://www.gnu.org/philosophy/free-sw.es.html.

Fuentes, Dra. María del Carmen Gómez. 2013. *Bases de Datos.* [ed.] Unuiversidad Autónoma Metropolitana. Mexico, D.F 2013 : Casa abierta al tiempo, 2013. pág. 201. 978-607-477-880-9.

Fundation, The Apache Software. 2017. The Apache JMeter™. *The Apache JMeter™.* [En línea] Apache Software Foundation, 2017. http://jmeter.apache.org/.

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

—. 2008. SIGB. Catálogos y gestión de autoridades. Diseño y prestaciones de OPACs. [En línea] 2008. eprints.rclis.org/13188/1/sigb.pdf.

Gómez, Laureano Felipe. 2007. *Interoperabilidad en los Sistemas de Información Documental (SID): la información debe fluir.* Bogotá : Códice, 2007. Vol. III. 1794-9815.

JISC. 2017. SWORD. *SWORD.* [En línea] JISC, 2017. http://swordapp.org/about/.

Larman, Craig. 2003. *UML y Patrones.* Vancuover : s.n., 2003. pág. 520. 84-205-3438–2.

Minter, Dave. 2008. *Beginning Spring 2.* New York : Apress, 2008. 978-1-4302-0493-1.

Nacional de Colombia, Universidad. 2017. Sistema de Informácion de la Amazonia Colombiana. *Sistema de Informácion de la Amazonia Colombiana.* [En línea] IMANI, 19 de Enero de 2017. http://www.unal.edu.co/siamac/sig/metadatos1.html.

NetBeans. 2017. NetBenas IDE Fits the Piece Together. *NetBenas IDE Fits the Piece Together.* [En línea] Oracle, 2017. https://netbeans.org/.

Oracle. 2005. Java. *Java.* [En línea] Oracle Corporation, 2005. https://www.java.com/es/download/faq/whatis_java.xml.

—. 2017. Oracle. *Oracle.* [En línea] Oracle, 2017. http://www.oracle.com/technetwork/middleware/glassfish/overview/index.html.

—. 2013. The Java EE 6 Tutorial . *The Java EE 6 Tutorial .* [En línea] Oracle, 2013. http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html.

Powell, Andy y Johnston, Pete. 2003. Metadata Innovation. *Metadata Innovation.* [En línea] The Metada Community, 2 de Abril de 2003. http://dublincore.org/documents/dc-xml-guidelines/.

Pressman, Roger S. 2010. *Ingeniería del Software. Un enfoque práctico.* Séptima. Estados Unidos : Mac Grau Hill, 2010. 9789701054734.

Prieto, José Ángel Parrado. 2012. *Sistemas Integrados de Gestión Bibliotecaria libres y de código abierto.* Facultad de Filosofía y Letras, Universidad de León. 2012.

Ramírez, Juan Pablo Siza. 2007. *Estudio, análisis, evaluación e implementación contenidos sobre internet para la interoperabilidad de repositorios de la Biblioteca Digital en la Universidad Nacional de Colombia con un proveedor de servicios de contenidos.* Departamento de Ingeniería de Sistemas e Industrial, Unversidad Nacional de Colombia. Bogotá : Unversidad Nacional de Colombia, 2007. pág. 124, Tesis para optar al grado de Magíster en Ingeniería de Teleocmunicaciones.

Regalado Martínez, Jorge. 2012. *Diseño e implementación de la Base de Datos del proyecto Sistema de Informatización del Registro de .* Universidad de las Ciencias Informáticas. Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2012. pág. 91, Tesis para optar por el título de Ingeniero en Ciencias Informáticas. TD_05494_12.

Tecnología. 2009. Tecnología. *Tecnología.* [En línea] Tecnología, 2009. http://www.areatecnologia.com/informatica/lenguajes-de-programacion.html.

UCI. 2017. Universidad de las Ciencias Informáticas. *Universidad de las Ciencias Informáticas.* [En línea] Universidad de las Ciencias Informáticas, 2017. http://www.uci.cu/universidad/mision.

UCI, Dirección de Servicis Jurídicos de la. 2014. Resolución MIgración de Software Libre. [En línea] 2014. http://migracion.uci.cu/wp-content/uploads/2014/04/Resoluci%C3%B3n-Migraci%C3%B3n-SWL-22_04_2014.pdf.

Universidad de Chile. 2011. Tutorial de UML. *Tutorial de UML.* [En línea] Universidad de Chile, 2011. http://users.dcc.uchile.cl/~psalinas/uml/introduccion.html.

Venemedia. 2014. conceptodefinicion.de. *conceptodefinicion.de.* [En línea] Venemedia, 2014. http://conceptodefinicion.de.

W3C. 2014. Guía Breve de Tecnologías XML. *World Wide Web Consortium (W3C).* [En línea] Fundación CTIC, 2014. http://www.w3c.es/Divulgacion/GuiasBreves/TecnologiasXML.

—. 2017. W3C . *W3C .* [En línea] 2017. http://www.w3c.es/Divulgacion/a-z/.

Zubiri, Federico y Mereles, Andrés. 2012. *Repositorios Institucionales.* 2012.