Universidad de las Ciencias Informáticas

Facultad 2

Módulo Depósito de Documentos para el Sistema de Repositorios Digitales REPXOS 3.0.

Trabajo de Diploma para optar por el Título de

Ingeniero en Ciencias Informáticas

Autores: Adrián Porras Cabrera
Landy Torres Crego

Tutores: Gleidis Yuriannis Rosabal Espinosa
Yorjander Tan Guevara

La Habana, Junio de 2015

"NO SE VIVE CELEBRANDO VICTORIAS, SINO DERROTAS."

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ___ días del mes _____del año 2015.

Autores:

_____	_____
Landy Torres Crego	Adrián Porras Cabrera

Tutores:

_____	_____
Ing. Gleidis Yuriannis Rosabal	Ing. Yorjander Tan Guevara

DATOS DE CONTACTO

Tutores:

Ing. Gleidis Yuriannis Rosabal Espinosa: Graduada en el año 2011 como Ingeniera en Ciencias Informáticas, en la Universidad de las Ciencias Informáticas. Se desempeña como especialista B en ciencias Informáticas en el Departamento de Aplicaciones del Centro CIGED ocupando el rol de analista del proyecto Repositorio Institucional. Correo electrónico: gyrosabal@uci.cu.

Ing. Yorjander Tan Guevara: Graduado en el año 2013 como Ingeniero en Ciencias Informáticas, en la Universidad de las Ciencias Informáticas. Se desempeña ocupando el rol de desarrollador en el proyecto Repositorio Institucional del Centro de Informatización de la Gestión Documental. Correo electrónico: ytan@uci.cu

DEDICATORIA

Adrián Porras Cabrera

Dedico este trabajo de diploma a todas las personas que quiero, en especial mis padres, mi hermano, mi novia, abuelos, tíos, primos y demás familiares.

A todos los amigos que he conocido en el transcurso de mi carrera universitaria y a la Universidad de las Ciencias Informáticas por darme la oportunidad de cumplir muchos sueños de mi vida.

DEDICATORIA

Landy Torres Crego

Le dedico el trabajo de diploma a mi familia, a todos mis amigos, a mi novia y a la universidad por prepararme como profesional.

AGRADECIMIENTOS

Adrián Porras Cabrera

A mis padres Griselberto y Gladis, por darme siempre su apoyo y cariño.

A mi hermano Arián por ser mi compañero.

A mi novia Yanieska por quererme y escucharme en los momentos difíciles.

A mis compañeros de apartamento Erdin, Yendry y Landy por los buenos y malos momentos que hemos pasado juntos.

A toda la familia de Artemisa, en especial Alain y Osvaldo que en aquellos momentos fueron mis hermanos.

AGRADECIMIENTOS

Landy Torres Crego

Le agradezco a mi familia que tanto me han apoyado en mi transcurso por la Universidad, han sido 5 años de sacrificio pero que nunca se olvidarán, por los conocimientos obtenidos y la preparación para la vida. En especial a mis mamás Ana y Tere que siempre han estado ahí pase lo que pase, a mi papá Iván que aunque no viva conmigo no para de preocuparse por mis estudios. A mi hermano Andy que es lo más grande que tengo, a Carlitos que más que mi primo es mi otro hermano A la mamá de mi novia en el poco tiempo que nos conocemos, ya se ha convertido en otra madre para mí. En especial a mi novia por apoyarme tanto que sin ella creo que no me hubiera graduado, es la que ha aguantado mis nervios por la tesis, me ha ayudado a estudiar, es la persona más linda que conozco, no solo por fuera, solo espero que ahora vengan tiempos buenos para que los comparta conmigo.

A las amistades forjadas, en la Mini UCI, los compañeros de cuarto Erdin con sus griterías, Yendry kun por soportarme tanto, al tigre que tanto trabajamos en la tesis y no parábamos de fajarnos, Orlando, Dorian, Yosley, Anibita, Lázaro, Israel, a Ema y sus yuju friends, en fin a todos los amigos que he conocido en la universidad, gracias a todos nunca olvidaré esta etapa de mi vida.

RESUMEN

Actualmente en la Universidad de la Ciencias Informáticas, en el proyecto Repositorio Institucional, se desarrolla el sistema REPXOS 3.0 basado en DSpace. REPXOS 3.0 se encarga de centralizar, normalizar, almacenar y preservar la producción científica y académica de las instituciones. Este sistema tiene integrado un módulo dedicado al envío de documentos digitales, que presenta problemas en su configuración. Solo se realizan algunas validaciones como la de los campos obligatorios, no siempre se muestran al usuario los mensajes adecuados para corregir errores cometidos. De esta forma se hace necesario el desarrollo del Módulo Depósito de Documentos, que permita configurar y validar los metadatos asociados al envío de documentos hacia el sistema para repositorios digitales REPXOS 3.0. Para lograr este objetivo se realiza un estudio de diferentes sistemas para depósito de documentos y para repositorios digitales, además de un análisis de servicio web y protocolo apropiado para lograr el envío de documentos y metadatos. Se selecciona el estilo de arquitectura REST para realizar el envío de documentos, así como la metodología, herramientas y tecnologías para el desarrollo del módulo. Se obtiene como resultado una aplicación que permita a los usuarios del sistema para repositorios digitales REPXOS 3.0 realizar un correcto envío, validando los campos necesarios y los metadatos asociados.

Palabras claves: configuración, depósito, metadatos, REST, validación.

Tabla de contenido

Introducción 14
1.1. Conceptos fundamentales. 19
1.2. Servicio Web y Protocolo para realizar envíos. 20
1.2.1. SWORD. 20
1.2.2. REST. 20
1.2.3. Selección de la vía para depósitos. 21
1.3. Sistemas para Repositorios Digitales. 22
1.3.1. DSpace. 22
1.3.2. Excriba. 22
1.3.3. ArXiv. 23
1.3.4. RePEc 23
1.3.5. PubMed Central. 24
1.4. Sistemas para Depósito de Documentos. 24
1.4.1. EasyDeposit. 24
1.4.2. BibApp. 25
1.5. Metodologías, herramientas, tecnologías y lenguajes para el desarrollo del sistema depósito de documentos. 26
1.5.1. Metodología de desarrollo. 26
1.5.2. DSpace v5.0 27
1.5.3. Sistema gestor de base de datos: PostgreSQL v9.2.4. 27
1.5.4. Herramienta de modelado: Visual Paradigm v8.0. 27
1.5.5. Herramienta de Mapeo objeto-relacional (ORM): Hibernate v3. 27
1.5.7. Servidor de aplicaciones: Apache Tomcat v7.0. 28
1.5.8. Lenguaje de programación Java, JavaScript. 28
1.5.9. Lenguaje de modelado: UML v2.1. 29
CAPÍTULO 2. Análisis y Diseño 31
2.1. Propuesta de solución. 31
2.2. Modelo del dominio. 31

2.3. Requisitos funcionales. 33
2.4. Requisitos no funcionales. 33
2.6. Especificación y resumen de los casos de uso. 35
2.7. Modelo de diseño. 36
2.7.1. Diagrama de clases del diseño. 36
2.7.2. Diagrama de colaboración. 39
2.7.3. Diagrama de paquetes. 41
2.8. Patrones utilizados. 43
2.8.1. Patrón de arquitectura MVC. 43
2.8.2. Patrón de diseño GRASP 44
2.8.3. Patrón de diseño Singleton. 45
CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBA 46
3.1. Implementación 46
3.1.1. Diagrama de Componente 46
3.1.2. Diagrama de Despliegue 48
3.1.3 Estándares de codificación 49
3.2. Pruebas 49
3.2.1 Estrategia de prueba 49
3.2.2. Resultado de las pruebas 52
CONCLUSIONES 55
RECOMENDACIONES 56
1.1 Anexo.1 61
GLOSARIO DE TÉRMINOS 70

Índice de Figuras

Figura 1: Modelo del Dominio 32
Figura 2: Diagrama de Caso de Uso. 35
Figura 3: Diagrama de clases del diseño Autenticar Usuario. 37
Figura 4: Diagrama de clases del diseño Gestionar formulario para el envío. 38
Figura 5: Diagrama de clases del diseño Enviar Documento. 39
Figura 6: Diagrama de colaboración Autenticar Usuario. 40
Figura 7: Diagrama de colaboración Crear Formulario. 40
Figura 8: Diagrama de colaboración Actualizar Formulario. 40
Figura 9: Diagrama de colaboración Eliminar Formulario. 40
Figura 10: Diagrama de colaboración Enviar Documento. 41
Figura 11: Diagrama de Paquetes 42
Figura 12: Diagrama de componente del Módulo Depósito de Documentos. 47
Figura 13: Diagrama de Despliegue. 48

Índice de Tablas

Tabla 1: Descripción del caso de uso Autenticar Usuario. 35
Tabla 2: Descripción del caso de uso Gestionar Formulario. 35
Tabla 3: Descripción del caso de uso Enviar Documentos. 36
Tabla 4: Resultados de las pruebas de Caja Negra. 53

Introducción

Actualmente se reconoce la importancia que reviste el empleo de las Tecnologías de la Información y las Comunicaciones (TIC). Se refieren a todos los instrumentos, procesos y soportes que están destinados a optimizar la comunicación humana. En las empresas, instituciones y organizaciones contar con la información oportuna para tomar las mejores decisiones en el momento adecuado, resulta algo fundamental.

En Cuba las TIC se aplican en esferas de la vida tales como; el sector de la salud, los servicios, la educación, las investigaciones y la gestión económica. En las empresas cubanas se pretende que ayude en la toma de decisiones, en la seguridad de la información. Que satisfaga en gran medida al cliente, que disminuyan los gastos, aumente la productividad, las respuestas sean rápidas, las acciones acertadas, la población tenga un alto desarrollo de su capacidad y conocimientos.

Las empresas, instituciones y personas han logrado informatizar los servicios y el control de documentos, debido al constante desarrollo de la información. Esta abarca documentos, informes, metodologías, publicaciones y está relacionada con la significación que adquieren los datos como resultado de su categorización y procesamiento. La información se maneja en favor de los objetivos estratégicos de una organización y en el cumplimiento de las misiones encomendadas a cada uno de sus individuos.

En la actualidad los sistemas de gestión documental constituyen una de las alternativas para gestionar grandes volúmenes de documentos e información. Los sistemas de gestión de documentos comúnmente proporcionan almacenamiento, seguridad, así como capacidades de recuperación de archivos. Un ejemplo de ello son los repositorios digitales. Un repositorio, depósito o archivo es un sitio web centralizado donde se mantiene y almacena la información digital en las bases de datos o archivos informáticos. Pueden contener los archivos en su servidor o referenciar desde su web al alojamiento originario. (1)

Los repositorios son herramientas eficaces para almacenar, organizar y hacer uso eficiente de la información y el conocimiento. Los recursos almacenados en los repositorios deben poseer una estructura estandarizada de metadatos que asegure la accesibilidad de los recursos digitales almacenados en el sistema. Más concretamente, los repositorios digitales son realizados para organizar, gestionar, preservar y dar libre acceso

a la producción científica, académica, institucional y de cualquier otro ámbito cultural, generada por los miembros de la institución en soporte digital. (2)

Los repositorios de documentos digitales se clasifican en temáticos e institucionales:

- Temáticos: almacenan y proporcionan acceso a la producción académica de un área temática particular, por ejemplo una disciplina. (3)
- Institucionales: recogen la producción de una institución y es la forma más extendida. Actualmente, se centran en una organización (universidad, departamento, instituto, sociedades científicas). Es posible definir políticas para que los miembros añadan contenidos. En esta clasificación, también se incluyen los repositorios de tesis doctorales. (4)

En la Universidad de las Ciencias Informáticas existen varios proyectos de desarrollo, entre ellos se encuentra el Repositorio Institucional, en el que se desarrolla el sistema para repositorios digitales REPXOS 3.0. La estructura que posee REPXOS 3.0 le permite organizar la información en comunidades que a su vez, se segmentan en colecciones de documentos. Su uso es factible, pues la configuración de sus funcionalidades, responde a las necesidades específicas de cada organización. Está desarrollado sobre la base del sistema DSpace, por lo que cuenta con todas sus funcionalidades y características que están implementadas en este.

DSpace es una herramienta para biblioteca digital diseñada para capturar, almacenar, ordenar, conservar y redistribuir la producción intelectual y de investigación de una universidad en formato digital. Soporta una gran variedad de datos, incluyendo libros, tesis, fotografías, videos, datos de investigación y otras formas de contenido. Es un software de código abierto que provee herramientas para la administración de colecciones digitales; y comúnmente es utilizada como solución de repositorio institucional. (5)

Actualmente REPXOS 3.0 contiene un módulo dedicado al envío de documentos digitales. Dentro de sus funciones principales se encuentran llenar un formulario con los metadatos correspondientes y adjuntar el archivo para su posterior envío. Este módulo presenta dificultades en cuanto a su configuración. No se muestran los metadatos específicos de un tipo de documento que se desea enviar. Solo se realizan algunas validaciones, como la de los campos obligatorios. Muchas veces el envío no se completa correctamente, pues no siempre se muestra al usuario los mensajes adecuados para que corrija los errores cometidos. Debe llegar al último paso para que se le muestre alguna información necesaria para poder completar con éxito el envío.

Estos detalles denotan la existencia de una aplicación poco sugerente y trae consigo pérdida de tiempo para los usuarios del sistema. Estos problemas se pudieran resolver realizando cambios en el código fuente del sistema, pero de surgir versiones nuevas de REPXOS se perderían y habría que realizar nuevamente modificaciones.

De la situación antes expuesta se origina el siguiente **problema científico:** ¿Cómo realizar el correcto envío de documentos al sistema para repositorios digitales REPXOS 3.0?

Se define como **objeto de estudio** los procesos que gestionan los depósitos de documentos, teniendo como **campo de acción** los procesos asociados a la validación y configuración de los metadatos para el depósito de documentos.

Para la solución del problema se plantea como **objetivo general:** desarrollar un módulo que permita configurar y validar los metadatos asociados al depósito de documentos en el sistema para repositorios digitales REPXOS 3.0.

Para dar cumplimiento al objetivo planteado se proponen las siguientes **tareas de la investigación**:

- Caracterización de los sistemas informáticos relacionados con el depósito de documentos, existentes a nivel nacional e internacional, estableciendo similitudes con la investigación en curso.
- Obtención de los artefactos necesarios de acuerdo a la metodología seleccionada.
- Implementación de los elementos definidos en el diseño en términos de componentes de implementación.
- Realización de las pruebas al Módulo de Depósito de Documentos apoyándose en los diseños de casos de prueba.

Los métodos científicos utilizados en la investigación fueron:

Métodos Teóricos.

- ✓ **Analítico-sintético:** Es usado para estudiar y analizar documentos y bibliografías de diferentes autores con el objetivo de poder realizar una investigación sobre los elementos que se relacionan con los procesos de los depósitos de documentos.

- ✓ **Modelación:** Se utilizó para modelar los diferentes artefactos generados, de acuerdo a la metodología RUP, en las etapas por las que transitó el desarrollo del Módulo Depósito de Documentos.
- ✓ **Análisis histórico-lógico:** Se utilizó para estudiar la evolución y características de los productos informáticos existentes que se relacionan con los sistemas para el depósito de documento.

Métodos Empíricos.

- ✓ **Observación:** Para obtener la información relacionada de los sistemas de depósito y conocer los aspectos generales del funcionamiento de estas aplicaciones, así como sus características fundamentales.

El Módulo Depósito de Documentos puede ser utilizado por instituciones y personas, permitiendo realizar un correcto depósito. Su uso posibilita, que al surgir versiones nuevas de REPXOS, no sea necesario realizar cambios en el código fuente del sistema. El módulo valida de forma correcta los metadatos y permite añadirlos y eliminarlos según un tipo de documento. Se muestran mensajes para corregir errores cometidos, posibilitando que los usuarios realicen con mayor facilidad el envío de documentos. Los archivos o documentos que serán enviados a través del Módulo Depósito de Documentos, tienen un carácter científico e investigativo, por lo que se concluye que estos apoyarán las tareas de enseñanza y aprendizaje, así como la mejora de la comunicación científica para las comunidades universitarias.

El presente trabajo está compuesto por una introducción, 3 capítulos, conclusiones, recomendaciones, referencias bibliográficas, glosario de términos y anexos.

Capítulo.1 Fundamentación teórica: En este capítulo se dan a conocer los principales temas de la investigación realizada. Se realiza un estudio de los sistemas de depósitos de documentos existentes con algunas de sus características y desventajas de uso. Se plantea la metodología, las tecnologías, herramientas y lenguajes que serán utilizados en la implementación del Módulo Depósito de Documentos. Se dan a conocer las características del servicio web y protocolo utilizado para realizar envíos de archivos hacia otros sistemas.

Capítulo.2 Análisis y Diseño: En este capítulo se caracteriza el Módulo Depósito de Documentos, se dan a conocer las funcionalidades a implementar. Se especifican los diagramas de paquetes y de clases del diseño, patrones de diseño y arquitectura para la elaboración del sistema.

Capitulo.3 Implementación y Prueba: En este capítulo se realizan los diagramas de componentes y despliegue. Se muestran las pautas de estándares de codificación de java y se valida la solución propuesta a través de varias pruebas realizadas al módulo implementado, para verificar su correcto funcionamiento, calidad y satisfacción del cliente.

CAPÍTULO 1. Fundamentación Teórica

Introducción

En el capítulo se profundiza en la rama de los sistemas informáticos de depósito de documentos, para comprender la importancia de los sistemas utilizados en la gestión de documentos. Se dan a conocer los beneficios de los sistemas estudiados, además de la metodología, herramientas, tecnologías y lenguajes para el desarrollo del sistema depósito de documentos. Se estudia el servicio web y el protocolo para seleccionar el que realizará envíos de documentos, además de mostrar y explicar algunas de las características de los diferentes sistemas para depósitos de documentos y sistemas para repositorios digitales.

1.1. Conceptos fundamentales.

Metadatos: Un metadato es un dato estructurado sobre la información, o de forma más simple, datos que describen datos. Los metadatos en el contexto de la Web, son datos que se pueden guardar, intercambiar y procesar por medio del ordenador. Están estructurados de tal forma que permiten ayudar a la identificación, descripción, clasificación y localización del contenido de un documento o recurso web y que, por tanto, también sirven para su recuperación. (5)

Dublin Core: El estándar de metadatos *Dublin Core* es un simple pero eficaz conjunto de elementos para describir una amplia gama de recursos de red. Describe material digital como video, sonido, imagen, texto utilizando el lenguaje XML, HTML y medios compuestos como páginas web (6)

Formulario: Es un conjunto de campos solicitados por un determinado programa que los almacenará y que serán utilizados o bien manipulados, en caso de ser necesario ejercer algún tipo de modificación sobre ellos. (7)

Depósito: El término depósito puede tener varias acepciones:

- Colocación de un objeto en un lugar determinado.
- Lugar destinado a contener objetos para guardarlos o conservarlos. (8)

El Módulo Depósito de Documentos, no se encarga del almacenamiento de documentos. En este caso, el término depósito que se utiliza, es la colocación de un objeto en un lugar determinado, es decir, el módulo se dedica a realizar envíos de documentos hacia el sistema REPXOS 3.0.

1.2. Servicios Web y Protocolos para realizar envíos.

Un servicio web es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Es un sistema de software diseñado para soportar una interacción interoperable máquina a máquina sobre una red. Los servicios web suelen ser **APIs** (Interfaces de Programación de Aplicaciones) Web que pueden ser accedidas dentro de una red (principalmente Internet) y son ejecutados en el sistema que los contiene. (9)

Para realizar el envío de documentos digitales se realizó el estudio del protocolo SWORD y el servicio web REST, pues están implementados en el sistema REPXOS.

1.2.1. SWORD.

SWORD es un protocolo usado en repositorios para poder realizar envíos de contenidos desde otras aplicaciones. Sus siglas corresponde a *Simple Web-service Offering Repository Deposit*, es decir un Servicio Web simple que ofrece servicios de depósito en un repositorio. Activar el protocolo SWORD en el repositorio REPXOS permite depositar mediante un servicio web documentos**.**

Características

- Protocolo para depositar en repositorios entre aplicaciones.
- Implementado como un mecanismo para extraer metadatos (contenidos dentro de un archivo .Zip).
- Implementado para diversas plataformas (DSpace, Eprints, Fedora, IntraLibrary). (10)

Ventajas

- Se puede configurar el servicio de envío para simplificar u omitir pasos del proceso de envíos.
- Cualquier usuario registrado o no registrado en el repositorio pueda insertar sus contenidos de forma simple.
- Permite añadir en el fichero de configuración nuevos formatos fácilmente de acuerdo a las necesidades del repositorio. (10)

1.2.2. REST.

REST (Transferencia de Estado Representacional) es un estilo de arquitectura de software para sistemas hipermedias distribuidos tales como la Web. En realidad, REST se refiere estrictamente a una colección de principios para el diseño de arquitecturas en red. Estos principios resumen cómo los recursos son definidos y

diseccionados. El término frecuentemente es utilizado en el sentido de describir a cualquier interfaz que transmite datos específicos de un domino sobre HTTP sin una capa adicional, como hace **SOAP** (Protocolo de Acceso a Objetos Simple). (9)

Cabe destacar que REST no es un estándar, es tan solo un estilo de arquitectura basado en estándares:

- HTTP (Protocolo de Transferencia de Hipertexto).
- URL (Identificador de Recursos Uniforme).
- Representación de los recursos: XML/HTML/GIF/JPEG/. Cuando el servidor ejecuta (es decir, envía) un documento al navegador, envía también cierta información adicional junto con el documento, lo que se conoce como encabezado HTTP. En el campo *Content-Type* (Tipo de Contenido) del encabezado HTTP, se describe el tipo de formato de datos. Esta información se expresa con una etiqueta para un tipo de **MIME** (Extensiones Multipropósito de Correo de Internet) de medios. (11)

Características.

- Las operaciones se definen en los mensajes.
- Presenta una dirección única para cada instancia del proceso.
- Cada objeto soporta las operaciones estándares definidas.
- Sus componentes están débilmente acoplados.

Ventajas.

- Bajo consumo de recursos.
- Las instancias del proceso son creadas explícitamente.
- El cliente no necesita información de enrutamiento a partir de la **URI** inicial.
- Generalmente fácil de construir y adoptar.

1.2.3. Selección de la vía para depósitos.

Para lograr el envío del Módulo Depósito de Documentos hacia el sistema REPXOS 3.0 se utilizará el estilo de arquitectura REST, pues este es un sistema independiente, se comunican con un lenguaje de intercambio como **JSON** (Notación de Objetos de JavaScript). Puede ser desarrollado en tecnologías o lenguajes con la que se puede acortar el tiempo de desarrollo. Requiere menos recursos del servidor, pues no mantiene el

estado, no requiere memoria, se pueden atender más peticiones. El auge del estilo de arquitectura REST ha provocado que tenga gran aceptación en la comunidad de DSpace. La nueva versión DSpace v5.0 está compuesta por nuevas funcionalidades añadidas como la implementación REST API con CRUD (Crear, Leer, Actualizar y Eliminar recursos). Estos se utilizan para manipular los recursos HTTP, los cuales deben operar con los siguientes métodos:

- GET: Para consultar y leer recursos.
- POST: Para crear recursos.
- PUT: Para editar recursos.
- DELETE: Para eliminar recursos.

1.3. Sistemas para Repositorios Digitales.

1.3.1. DSpace.

DSpace es un software de código abierto para la gestión de repositorios. Provee herramientas para la administración de colecciones digitales, y comúnmente es usado como solución de repositorio institucional. Conserva y permite el acceso fácil y abierto a todo tipo de contenido digital, incluyendo texto, imágenes, videos y conjuntos de datos. Utiliza el estándar de metadatos Dublin Core. Presenta capacidades de importación y exportación de documentos digitales. El sistema los contiene los metadatos en un archivo de configuración, estos no se encuentran especificados de acuerdo a un tipo de documento. Realiza algunas validaciones, como la de los campos obligatorios.

La aplicación puede reconocer y manejar un gran número de formatos de archivos y tipos MIME (Extensiones Multipropósito de Correo de Internet). Algunos de los formatos más comunes que gestiona actualmente el entorno de DSpace están en formato PDF, Word, JPEG, MPEG. Archivos de cualquier formato pueden ser gestionados por Dspace. Este proporciona un registro de formatos de archivos simple donde se puede registrar cualquier formato no reconocido, de manera que se puede identificar en el futuro.

1.3.2. Excriba.

Excriba es un software para la gestión documental, diseñada para tramitar los documentos administrativos que se generan o reciben dentro de las organizaciones a partir de sus funciones. Involucra todas las áreas de

una organización, permitiendo gestionar de forma correcta la documentación como prueba, testimonio y evidencia de las actividades organizacionales.

El flujo de trabajo en Excriba comienza una vez que el usuario se autentica en el sistema, primero selecciona el informe que se va a enviar, permitiendo crear tres tipos de documentos: Texto plano, HTML y XML. Para facilitar la carga del contenido es posible arrastrar y soltar documentos dentro de la biblioteca o espacio de trabajo, así como cargar varios a la vez. Los documentos se almacenan directamente en el mismo sistema, para ser revisado por la persona encargada.

1.3.3. ArXiv.

ArXiv es un archivo para borradores finales de artículos científicos en la Física, las Matemáticas, la Informática y la Biología accesible a través de Internet. La mayoría de las publicaciones científicas se colocan en el archivo ArXiv.

El objetivo principal de ArXiv es almacenar artículos en formatos abierto que no ocultan la información, la mejor opción es TeX / LaTeX[1]. Los usuarios de los procesadores de texto como Microsoft Word (versiones anteriores a Word 2007) deben guardar sus documentos en formato PDF, para presentarlos. No acepta los documentos escaneados, independientemente del formato. Antes de realizar el proceso de envío, se debe revisar cuidadosamente los metadatos. Si se detecta un error después de ser enviado el artículo, solo se puede corregir antes de estar disponible públicamente. (12)

1.3.4. RePEc

RePEc son las siglas de *Research Papers in Economics* (Trabajos de Investigación para la Economía). Básicamente RePEc es un conjunto de herramientas conceptuales, protocolos, normas y software cuyo objetivo es la distribución electrónica y descripción bibliográfica de los documentos científicos producidos por una disciplina académica, en concreto la Economía.

Soporta archivos de texto plano a veces llamados archivos ASCII, y no los archivos en formato RTF (Formato de Texto Rico), de formato binario de Word (.doc), hipertexto (.html) ni los archivos con los nombres que terminan en (.txt).

[1] LaTeX: es un sistema de preparación de documentos de alta calidad especialmente diseñado para la producción de documentos científicos y técnicos.

1.3.5. PubMed Central.

PubMed Central (PMC) es el archivo digital gratuito de revistas de ciencias biológicas y biomédicas de los Institutos Nacionales de Salud. La participación de los editores en PMC es voluntaria, pero las revistas que participan deben cumplir determinadas normas editoriales y técnicas. El acceso a PMC es libre e irrestricto. (13)

PubMed Central publica sus artículos en formato XML, algunas revistas contienen lo que se puede clasificar en términos amplios como contenido administrativo – miembros del consejo editorial, lista de empleados, instrucciones a los autores, avisos, anuncios, y publicidad. En la actualidad, los contenidos de ese tipo sólo están disponibles en forma impresa o en una página HTML en el sitio web de la revista. PubMed está elaborando directrices para el etiquetado XML, de modo que el contenido administrativo también pueda ser incluido en el archivo de PMC. Acepta algunos materiales administrativos, si la revista los publica en forma de artículo y está en condiciones de suministrar el texto completo en XML.

1.4. Sistemas para Depósito de Documentos.

1.4.1. EasyDeposit.

Es un sistema capaz de efectuar depósitos en un repositorio DSpace. Se compone de una serie de pasos. Estos se pueden configurar en diferentes órdenes, según las necesidades. Un flujo típico de trabajo puede ser: inicio de sesión, seleccionar un repositorio, introducir algunos metadatos, cargar un archivo, comprobar que la información es correcta, realizar el depósito, enviar un correo electrónico de confirmación. Alternativamente, un flujo de depósito solo puede requerir un archivo para ser cargado y un título introducido. Los pasos se pueden agregar fácilmente mediante la creación de dos nuevos archivos, uno que contiene la lógica de negocio (codificado en PHP) y uno que contiene la vista (HTML). (14)

EasyDeposit se puede instalar en un equipo sencillo, requiere poca infraestructura. Su interfaz de usuario está escrito en el lenguaje de programación PHP, y no hay ninguna base de datos subyacente, por lo que es fácil de instalar, configurar y personalizar. EasyDeposit es un sistema para depósitos individuales, esto puede ser útil para los usuarios ocasionales, particularmente los usuarios que presentan un solo documento, como estudiantes que depositan sus tesis. (15)

1.4.2. BibApp.

Ayuda a recopilar, refinar, organizar y presentar la información sobre la producción académica. Las bibliotecas de las universidades pueden utilizar BibApp para comprender mejor los patrones de publicación de la facultad y fácilmente identificar material para un repositorio institucional.

BibApp es un sistema independiente y de código abierto que posibilita el envío de archivos y metadatos hacia repositorios tales como: DSpace, Eprints y Fedora. Permite promocionar la investigación, explotar los datos para ver colaboraciones y encontrar expertos en las áreas, con el fin de hacer más accesible la investigación en general. Permite reutilizar fácilmente los datos de publicación. Facilita ver qué publicaciones pueden ser archivadas para un mayor acceso e impacto, y hace que sea fácil enviar las publicaciones directamente a un repositorio institucional o de otro tipo. (16)

Valoración sobre los sistemas estudiados.

Luego de haber realizado un análisis de los sistemas informáticos estudiados se puede concluir que; DSpace no valida todos los metadatos, solo los campos obligatorios. No se muestran los metadatos específicos de acuerdo a un tipo de documento. Excriba almacena los archivos o documentos directamente en el mismo sistema. ArXiv solo almacena artículos en formato TeX / LaTeX, que no permite PDF creado a partir de esta fuente. RePEc no almacena el formato binario de Word (.doc), hipertexto (.html) ni los archivos con los nombres que terminan en (.txt). PubMed Central publica sus artículos en formato XML. Ninguno de los sistemas anteriores necesitan de la utilización de servicios web para realizar el envío de documentos hacia otros sistemas, todo el proceso de subida de documentos se realiza para ser almacenados en la misma aplicación. EasyDeposit es un sistema para depósitos individuales, solo se encarga del envío de los documentos. En el caso de BibApp existe inconsistencia entre los documentos publicados, asociándolos a más de un autor, provocando confusión de autores. Es por ello que ningunos de los sistemas mencionados constituyen una solución factible.

Por los aspectos antes mencionados, queda demostrada la necesidad de desarrollar el Módulo de Depósito de Documentos. Los sistemas estudiados brindan una guía para conocer los principales pasos del proceso de envío de documentos y se aprovechan las mejores características de ellos para el diseño del módulo. Se toma el modo en que ArXiv gestiona los formularios, utilizando el estilo de pestañas para realizar los diferentes pasos en el proceso de envío. Se utiliza la forma que DSpace describe los metadatos, para que los usuarios

conozcan la descripción de los campos que se llenarán y se acepta la licencia para depósito. Se cargan los archivos para su posterior envío, como lo realiza el sistema EasyDeposit a través de la opción “Browse”.

Metodologías, herramientas, tecnologías y lenguajes para el desarrollo del sistema depósito de documentos.

Las metodologías, herramientas y tecnologías utilizadas son muy importantes definirlas para lograr un mejor desarrollo del software y que el mismo tenga la calidad suficiente para la satisfacción de los usuarios del sistema.

1.4.3. Metodología de desarrollo.

Proceso Racional Unificado **(**RUP) es una metodología cuyo fin es entregar un producto de software. Se estructuran todos los procesos y se mide la eficiencia de la organización. Es un proceso de desarrollo de software el cual utiliza el lenguaje unificado de modelado UML. Es la metodología utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP es un conjunto de metodologías adaptables al contexto y necesidades de cada organización

RUP permite el desarrollo de software a gran escala, mediante un proceso continuo de pruebas y retroalimentación, garantizando el cumplimiento de ciertos estándares de calidad. Aunque con el inconveniente de generar mayor complejidad en los controles de administración del mismo. Sin embargo, los beneficios obtenidos recompensan el esfuerzo invertido en este aspecto. (17)

El proceso de desarrollo constituye un marco metodológico que define en términos de metas estratégicas, objetivos, actividades y artefactos (documentación) requeridos en cada fase de desarrollo. Esto permite enfocar esfuerzo de los recursos humanos en términos de habilidades, competencias y capacidades a asumir roles específicos con responsabilidades bien definidas. (17)

Herramientas, tecnologías y lenguajes.

Las herramientas, tecnologías y lenguajes juegan un papel fundamental para la elaboración del Módulo Depósito de Documentos, es por ello que a continuación se especifican y se describen cada una de ellas.

1.4.4. DSpace v5.0

Dspace soporta una gran variedad de datos, incluyendo libros, tesis, fotografías, filmes, video, datos de investigación y otras formas de contenido. Los datos son organizados como ítems que pertenecen a una colección; cada colección pertenece a una comunidad. Cuenta con algunas características de gran importancia, que permiten y facilitan su uso en una gran cantidad de instituciones de diversos tipos. Entre ellas se encuentra su distribución bajo Licencia BSD[2]; que es multiplataforma, por lo que se adapta a gran número de Sistemas Operativos; es basado en tecnología Web y es un sistema adaptable, además de que posee un sistema de *Handles* (Manejadores) de archivos. (18)

1.4.5. Sistema gestor de base de datos: PostgreSQL v9.2.4.

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. (19)

1.4.6. Herramienta de modelado: Visual Paradigm v8.0.

Visual Paradigm es una herramienta CASE[3]. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación.

Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas.

1.4.7. Herramienta de Mapeo objeto-relacional (ORM): Hibernate v3.

Hibernate es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java, que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) de las entidades que permiten establecer estas relaciones. Hibernate permite desarrollar clases persistentes orientados a objetos, incluyendo la herencia, polimorfismo, la asociación y la

[2] La licencia BSD: es la licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution). Es una licencia de software libre. La licencia BSD permite el uso del código fuente en software no libre.

[3] CASE: Por sus siglas en español Ingeniería de Software Asistida por Computadora.

composición. Permite cualquier clase o estructura de datos. No requiere de tablas de bases de datos especiales o campos, y genera gran parte del SQL en el sistema en tiempo de inicialización, en lugar de en tiempo de ejecución. Hibernate es conocido por su estabilidad y calidad, probado por la aceptación y es usado por los desarrolladores de Java. (20)

1.4.8. Entorno integrado de desarrollo: Eclipse Luna SR1.

Eclipse es un programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma. Incluye varias características, presenta una lista de tareas, soporte para unidades de pruebas con JUnit, e integración con la herramienta de construcción: Ant. Dispone de un editor de texto con un analizador sintáctico. La compilación es en tiempo real. Tiene control de versiones con CVS[4], asistentes (*wizards*) para creación de proyectos, clases, y pruebas. Así mismo, a través de *plugins* libremente disponibles es posible añadir control de versiones con *Subversion.4* e integración con *Hibernate 5*. (21)

1.4.9. Servidor de aplicaciones: Apache Tomcat v7.0.

Es multiplataforma, lo que lo convierte en un servidor prácticamente universal; es un contenedor de servlets/JSP, es decir implementa las especificaciones de los **servlets** y de JavaServerPages (JSP), utiliza la tecnología Apache que es gratuita y de código abierto lo que le da transparencia al software. Permite escribir y desplegar aplicaciones web complejas de forma sencilla, proporcionando el soporte para características de desarrollo que de otra forma tendrían que ser desplegadas manualmente. Resuelve las opciones de autenticación de usuarios, los registros de log, así como la infraestructura y los filtrados a nivel de aplicación y es configurable. (22)

1.4.10. Lenguaje de programación Java, JavaScript.

Java es un lenguaje simple. Orientado al objeto, distribuido, interpretado, sólido, seguro, de arquitectura neutral, portable, de alto desempeño, de multihilos y dinámico. (23)

Es una tecnología que no sólo se reduce al lenguaje sino que además provee una máquina virtual Java que permite ejecutar código compilado, sea cual sea la plataforma que exista por debajo; plataforma tanto hardware, como software (el sistema operativo que soporte ese hardware). (24)

[4] CVS: Por sus siglas en español Control de versiones.

JavaScript es un lenguaje interpretado, multiplataforma, orientado a eventos con manejo de objetos, cuyo código se incluye directamente en el mismo documento, usado para el desarrollo de aplicaciones cliente-servidor en páginas HTML.

Las principales características del JavaScript son:

- Está orientado a objetos de forma limitada, no maneja los conceptos como la herencia y los métodos. Se puede definir un objeto dentro de la página web y sobre ese objeto definir diferentes eventos que producirán una salida. Un ejemplo puede ser: presionar un botón, pasar el puntero del mouse sobre un determinado texto o el simple hecho de cargar la página web, eventos que darán una gran versatilidad a la hora de crear programas en JavaScript.
- Es dinámico, como se menciona antes JavaScript responde a eventos, esos eventos son producidos por el propio usuario y el JavaScript reacciona a ellos en tiempo real. (25)

1.4.11. Lenguaje de modelado: UML v2.1.

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. La especificación de UML no define un proceso estándar pero puede ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos. (26)

Conclusiones

Con la realización del capítulo se describieron y caracterizaron varios aspectos relacionados con los sistemas de depósito de documentos. Se logró recopilar información relevante sobre los sistemas de depósito de documentos existentes, para así conocer que ninguno es una solución factible. Permitió comprender la importancia y beneficios de los sistemas de depósitos de documentos para la comunidad universitaria. Se estudiaron las características fundamentales de las herramientas, tecnologías, lenguajes y metodologías de

desarrollo que se utilizarán, sentando las bases para el desarrollo del módulo que forma parte de la propuesta de solución. El estudio de los servicios web permitió seleccionar el estilo de arquitectura REST para realizar envíos de documentos hacia el sistema REPXOS 3.0.

CAPÍTULO 2. Análisis y Diseño

Introducción

En este capítulo se describe la solución propuesta del módulo y su funcionamiento, se plantean los requisitos funcionales y no funcionales del sistema, además de exponer los diagramas de casos de uso del Módulo Depósito de Documentos. Se muestran los diagramas de paquetes y de clases del diseño, además se dan a conocer los patrones de diseños y de arquitectura utilizados para la implementación y elaboración del sistema.

2.1. Propuesta de solución.

Después de realizar el estudio de la situación problemática y conocer las necesidades del proyecto de desarrollo Repositorio Institucional, se propone como solución la implementación del Módulo Depósito de Documentos. El módulo permitirá realizar de forma correcta la configuración y validación de los metadatos asociados a los documentos que se envíen hacia REPXOS. Para ello se muestran los metadatos específicos según el tipo de documento que se desea enviar. Se realiza la validación de los pasos necesarios para lograr el correcto depósito de los documentos, mostrando en todo momento mensajes sugerentes de acuerdo a los posibles errores que pudiera cometer un usuario durante el envío.

Previo al envío de documentos digitales se debe crear un formulario, llenando los campos necesarios según el tipo de documento definido. Este formulario puede ser guardado con los elementos básicos y posteriormente retomarlo para realizar el envío. Además podrá ser actualizado o eliminado por el usuario en el momento que desee.

El módulo se desarrolla independiente del sistema REPXOS y el envío se realiza a través del servicio web REST. Esta forma de envío evitaría que, en caso de que se libere una nueva versión del DSpace, no se tengan que realizar cambios en el código fuente. Como el módulo funciona independiente, se hace necesario además poder controlar el acceso al sistema. Para lograr la seguridad de este cada persona que acceda al sistema contará con un usuario y una contraseña. La autenticación será una condición necesaria para poder realizar cualquier acción.

2.2. Modelo del dominio.

Un modelo de dominio es una representación visual de clases conceptuales o de objetos reales en un dominio de interés. Puede utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como

paso previo al diseño de un sistema. El modelo de dominio es utilizado por el analista como un medio para comprender el sector de negocios al cual el sistema va a servir. Ayuda a comprender los conceptos clave de un negocio o un dominio de problema. Disminuye la brecha de representación entre cómo ven los clientes el problema y la representación en software de la solución, usando modelado Orientado por Objetos. (27)

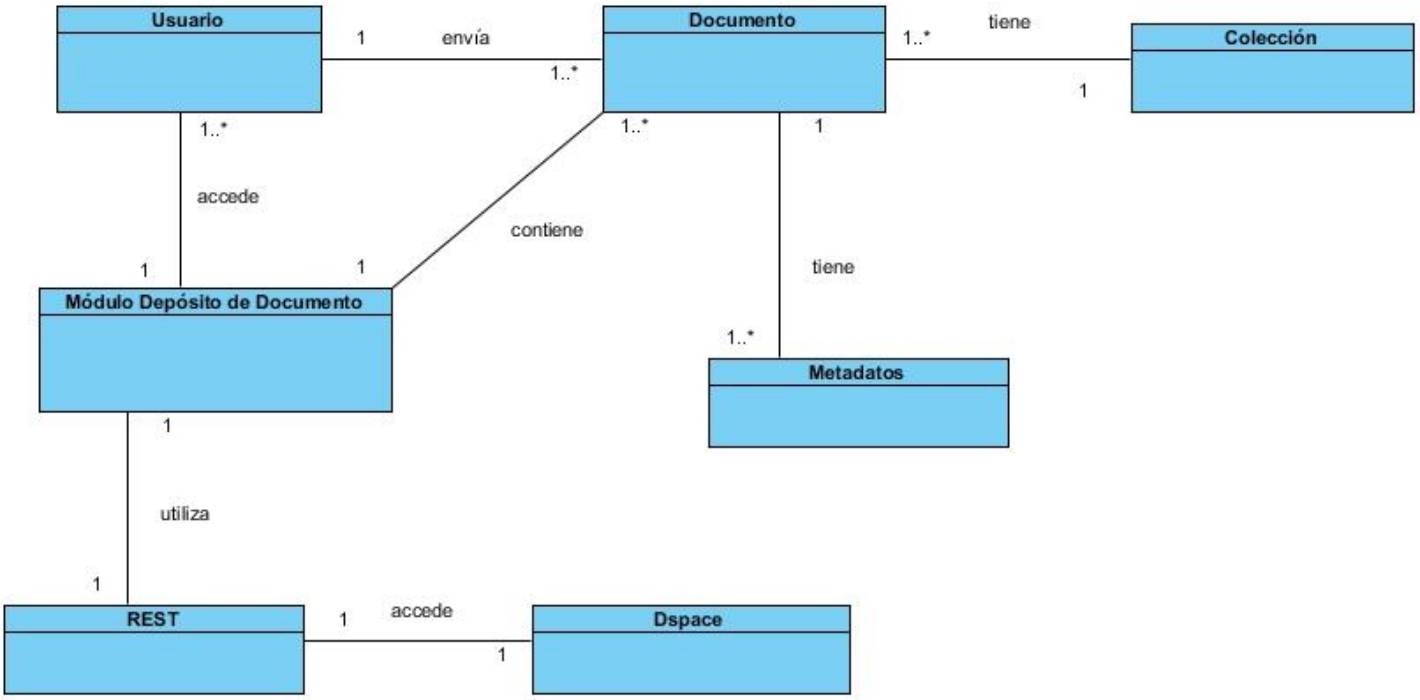


Figura 1: Modelo del Dominio

Definición de conceptos del modelo del dominio:

- **Colección:** es un conjunto de documentos recopilados por una persona o una institución.
- **Dspace:** sistema al que serán enviados los documentos digitales.
- **Documento:** es el tipo de archivo que será enviado al sistema REPXOS 3.0.
- **Módulo Depósito de Documentos:** es el sistema que permite gestionar los metadatos de acuerdo al tipo de documento que se desea enviar.
- **Metadatos:** datos que describen un documento.

- **REST:** servicio mediante el cual se conectará el Módulo Depósito de Documentos con el sistema REPXOS 3.0 para realizar el envío de documentos.
- **Usuario:** es la persona que interactúa con el sistema y sus funcionalidades.

2.3. Requisitos funcionales.

Los requerimientos funcionales son declaraciones de los servicios que debe proporcionar el sistema, la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. (28)

A continuación se definen los siguientes requisitos funcionales:

RF1: Autenticar usuario.

RF2: Gestionar formulario para el envío.

RF2.1: Crear formulario para el depósito de documentos.

RF2.2: Actualizar formulario para el depósito de documentos.

RF2.3: Eliminar formulario para el depósito de documentos.

RF3: Enviar documentos.

2.4. Requisitos no funcionales.

Los requerimientos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema. (28)

Eficiencia:

- **RnF 1. Capacidad:** El sistema soporta 200 usuarios conectados. Esto fue probado con las pruebas de carga. Se utilizó la herramienta JMeter 2.13.
- **RnF 2 Recursos:** Para el correcto funcionamiento del sistema se debe tener:

  Servidor de Base de Datos:

  Procesador: Inter Core i3 a 3.20 Ghz.

  Memoria RAM: 2Gb mínimo.

Disco Duro: 40 Gb

Servidor de Aplicaciones:

Procesador: Inter Core i3 a 3.20 Ghz.

Memoria RAM: 2Gb mínimo.

Disco Duro: 40 Gb

El sistema donde se despliega el Módulo Depósito de Documento debe cumplir como mínimo con los recursos antes planteados. De lo contrario no se garantiza el correcto funcionamiento del módulo.

Restricciones del diseño:

- **RnF 3 Lenguaje de programación:** Se utiliza para la construcción del sistema los lenguajes de programación Java, HTML, JavaScript y las herramientas que se utilizan son de distribución bajo licencias libres, para favorecer el desarrollo de nuevos módulos o subsistemas.

Interfaz:

- **RnF 4 Interfaces de usuario:** Para acceder al sistema debe usarse una versión del navegador Mozilla/Firefox 33.0, Internet Explorer 10.0. No se garantiza la correcta visualización en otros navegadores.
- **RnF 5 Interfaces de Hardware:** El hardware donde se instaló el sistema posee una interfaz de red cuya velocidad de transferencia es de 100 Mbps. Para garantizar el correcto funcionamiento de la aplicación, se sugiere que se utilice una interfaz de red con la velocidad de transferencia antes mencionada.
- **RnF 6 Interfaces de Software:** El sistema debe integrarse con los siguientes productos de software: PostgreSQL 9.2.4. La aplicación se podrá ejecutar sobre el sistema operativo Linux y Windows.

Seguridad:

- **RnF 7 Seguridad del Sistema:** El software estará protegido contra accesos no autorizados. Se utilizarán mecanismos de validación que puedan ofrecer el cumplimiento de esto: cuenta, contraseña y nivel de acceso, de manera que cada usuario pueda tener disponible solamente las opciones relacionadas con su actividad y tenga datos de acceso propios, garantizando así la confidencialidad.

2.5. Diagrama de Caso de Uso.

Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y otros sistemas. Es un diagrama que muestra la relación entre los actores y los casos de uso en un sistema. A continuación se observa el Diagrama de Caso de Uso del Módulo de Depósito de Documentos.

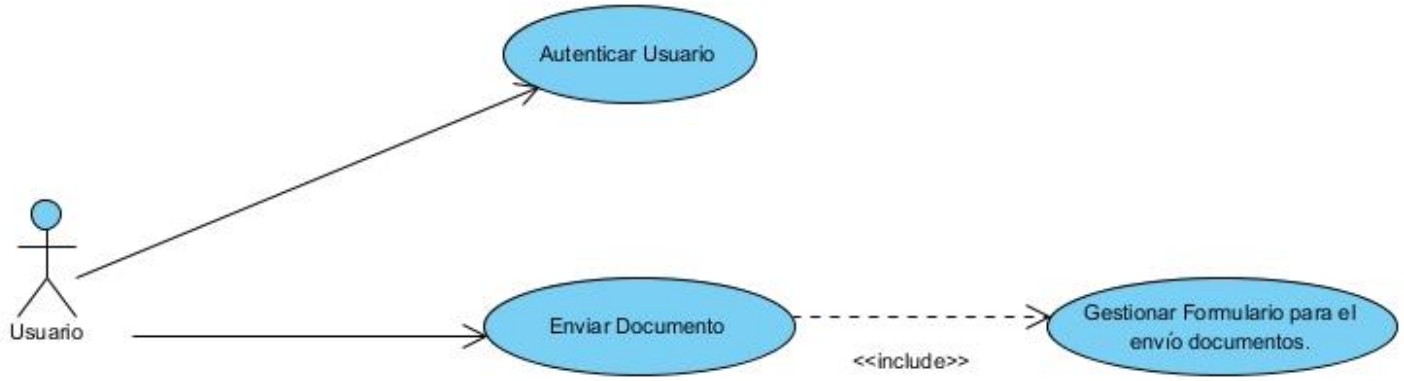


Figura 2: Diagrama de Caso de Uso.

2.6. Especificación y resumen de los casos de uso.

Un caso de uso contiene una descripción textual de todas las maneras que los actores previstos podrían trabajar con el software o el sistema. A continuación se muestra un resumen de los casos de usos del sistema.

Tabla 1: Descripción del caso de uso Autenticar Usuario.

Objetivo	Identificar usuario para la entrada al sistema
Actores	Usuario
Resumen	El caso de uso se inicia cuando el actor especificado desea acceder a al sistema. El mismo introduce su usuario y contraseña y en vista del rol y permisos que tenga definido accede a las funcionalidades específicas según los privilegios que le fueron asignados.
Complejidad	Baja
Prioridad	Media
Precondiciones	El sistema debe estar funcionando correctamente.
Postcondiciones	El usuario accede al sistema.

Tabla 2: Descripción del caso de uso Gestionar Formulario.

Objetivo	Crear, actualizar y eliminar formulario.
Actores	Usuario
Resumen	El caso de uso se inicia cuando el actor desea crear, actualizar o eliminar un formulario. Si el

	actor selecciona la opción crear, el sistema le permite registrar los datos perteneciente a dicho formulario. Si desea modificar, el sistema muestra los datos para realizar los cambios. Si el actor selecciona la opción eliminar el sistema elimina el formulario creado para cualquier documento que se desee depositar. Termina el caso de uso.
Complejidad	Media
Prioridad	Media
Precondiciones	-El sistema debe estar funcionando correctamente. -Un usuario se ha autenticado en el sistema.
Postcondiciones	Se creó, modificó y eliminó un formulario.

Tabla 3: Descripción del caso de uso Enviar Documentos.

Objetivo	Enviar documento	
Actores	Usuario.	
Resumen	El caso de uso se inicia cuando el actor desea depositar un documento. El actor llena todos los campos del formulario y podrá enviar el documento.	
Complejidad	Alta	
Prioridad	Media	
Precondiciones	-El sistema debe estar funcionando correctamente. -Un usuario se ha autenticado en el sistema. -Debe haber llenado el formulario del documento a enviar.	
Postcondiciones	Se ha enviado el documento.	
Relaciones	CU Incluidos	Gestionar formulario para el envío.
	CU Extendidos	

2.7. Modelo de diseño.

El modelo de diseño comprende los diagramas de clases de software, diagramas de interacción, diagramas de paquetes, ofreciendo una perspectiva de especificación o implementación, como quiere el modelador. *(29)*

2.7.1. Diagrama de clases del diseño.

Un diagrama de clases de diseño representa las especificaciones de las clases e interfaces software en una aplicación. Entre la información general se encuentra:

- Clases, asociaciones y atributos.
- Interfaces, con sus operaciones y constantes.
- Métodos.
- Información acerca del tipo de los atributos.

- Navegabilidad.
- Dependencias.

A diferencia de las clases conceptuales del modelo del dominio, las clases del diseño de los diagrama de clases de diseño muestran las definiciones de las clases software en lugar de los conceptos del mundo real. (30) A continuación se muestran los diagramas de clases del diseño del Módulo Depósito de Documentos.

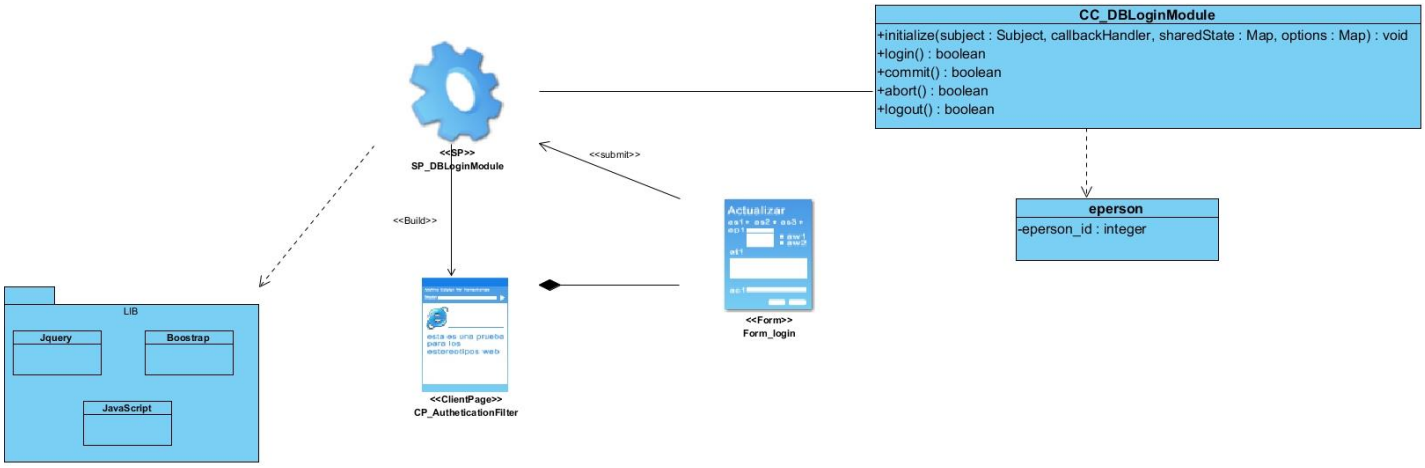


Figura 3: Diagrama de clases del diseño Autenticar Usuario.

Definición de clases del diagrama de clases:

AuthenticationFilter: Se encarga de confirmar si el usuario y contraseña se encuentran en la Base de Datos de REPXOS 3.0.

DBLoginModule: Controla los módulos para permitir la autenticación en el sistema.

Login: Es la vista para que el usuario se registre en el sistema.

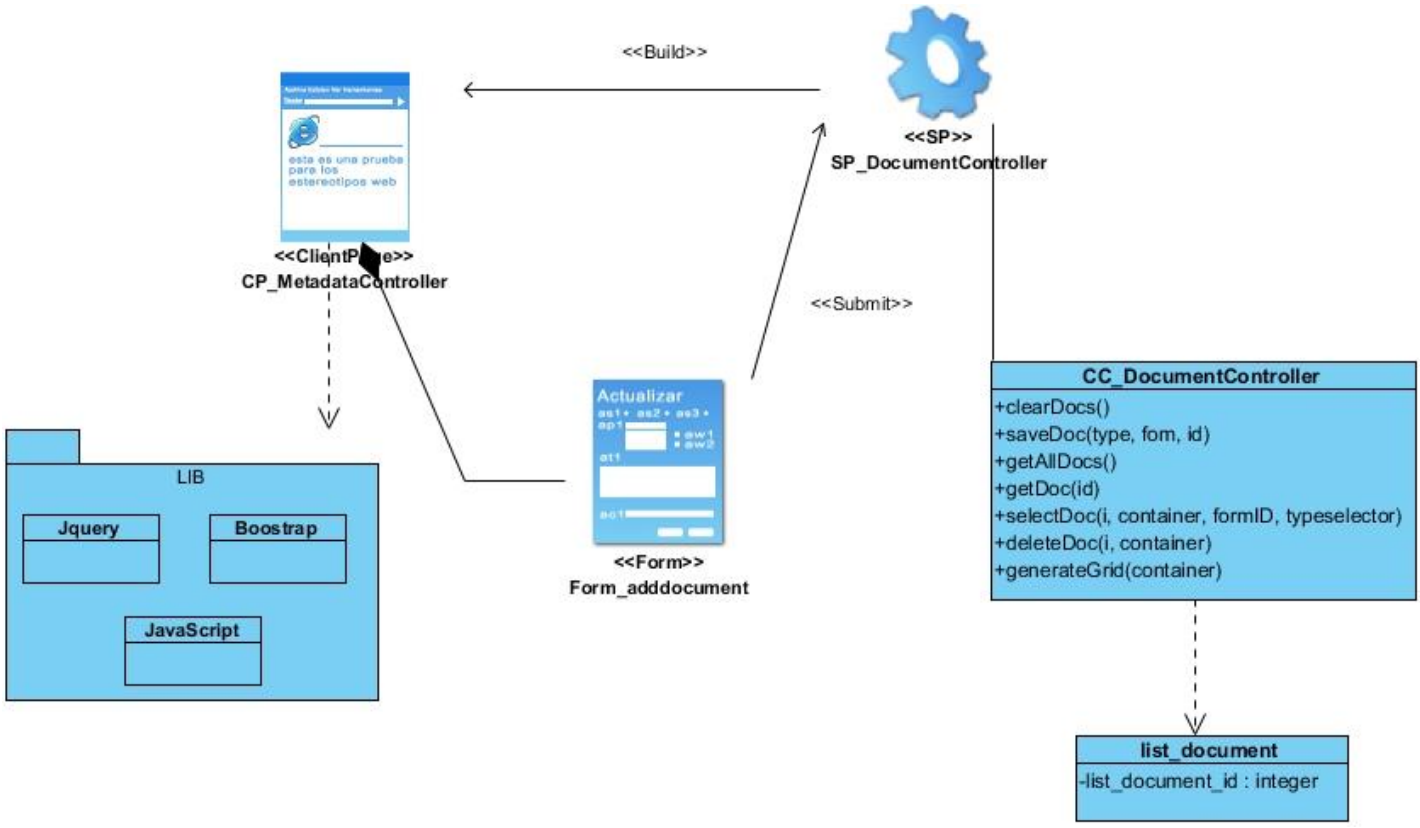


Figura 4: Diagrama de clases del diseño Gestionar formulario para el envío.

Definición de clases del diagrama de clases:

adddocument: Es la vista para la gestión de los formularios.

DocumentController: Se encarga de gestionar los formularios para su posterior envío.

MetadataController: Se encarga de la validación y configuración de los metadatos.

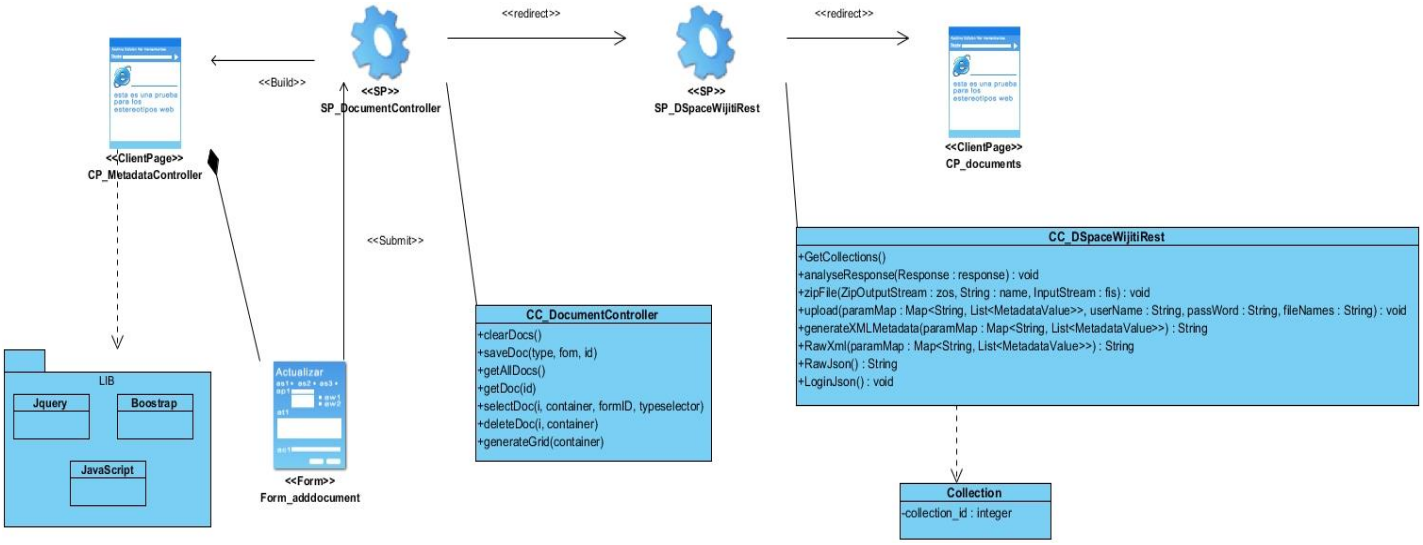


Figura 5: Diagrama de clases del diseño Enviar Documento.

Definición de conceptos del diagrama de clases:

adddocument: Es la vista para la gestión de los formularios.

DocumentController: Se encarga de gestionar los formularios para su posterior envío.

DSpaceWijitiRest: Se encarga del envío de documentos.

MetadataController: Se encarga de la validación y configuración de los metadatos.

2.7.2. Diagrama de colaboración.

Con el uso de un diagrama de colaboración se muestra la implementación de una operación. La colaboración muestra los parámetros y las variables locales de la operación, así como asociaciones más permanentes. Cuando se implementa el comportamiento, la secuencia de los mensajes corresponde a la estructura de llamadas anidadas y el paso de señales del programa. Un diagrama de colaboración muestra las relaciones entre roles geométricamente, y relaciona los mensajes con las relaciones, pero las secuencias temporales están menos claras, porque vienen dadas por los números de secuencia. (26)

Figura 6: Diagrama de colaboración Autenticar Usuario.

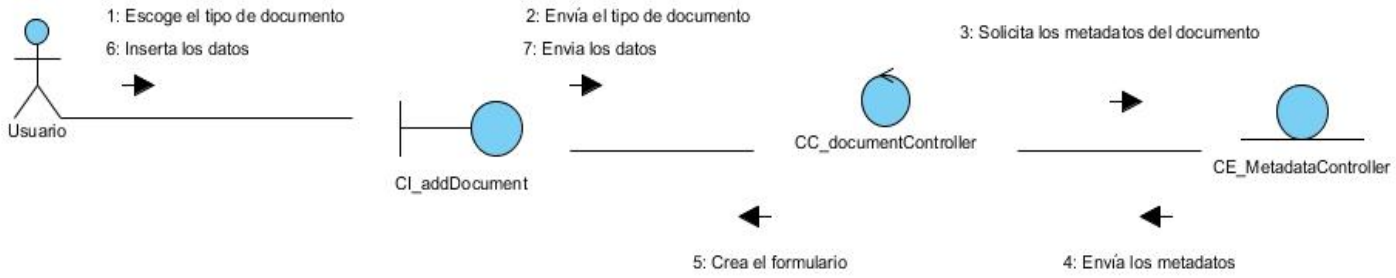


Figura 7: Diagrama de colaboración Crear Formulario.

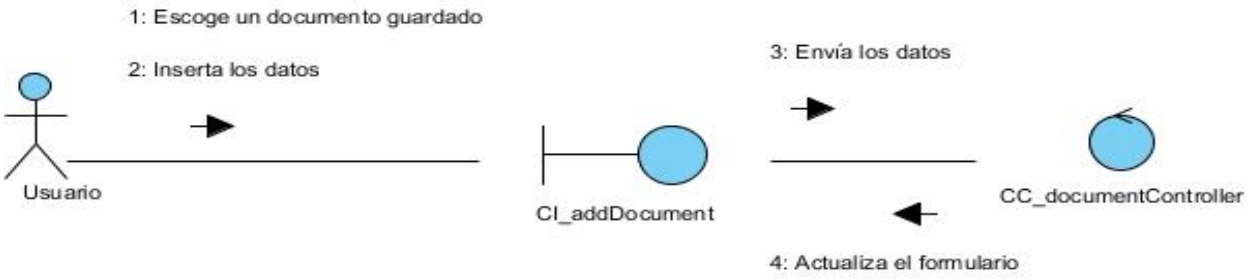


Figura 8: Diagrama de colaboración Actualizar Formulario.

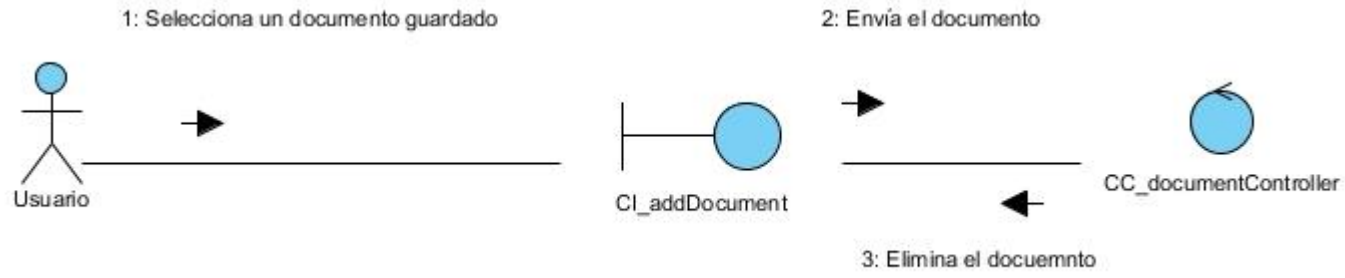


Figura 9: Diagrama de colaboración Eliminar Formulario.

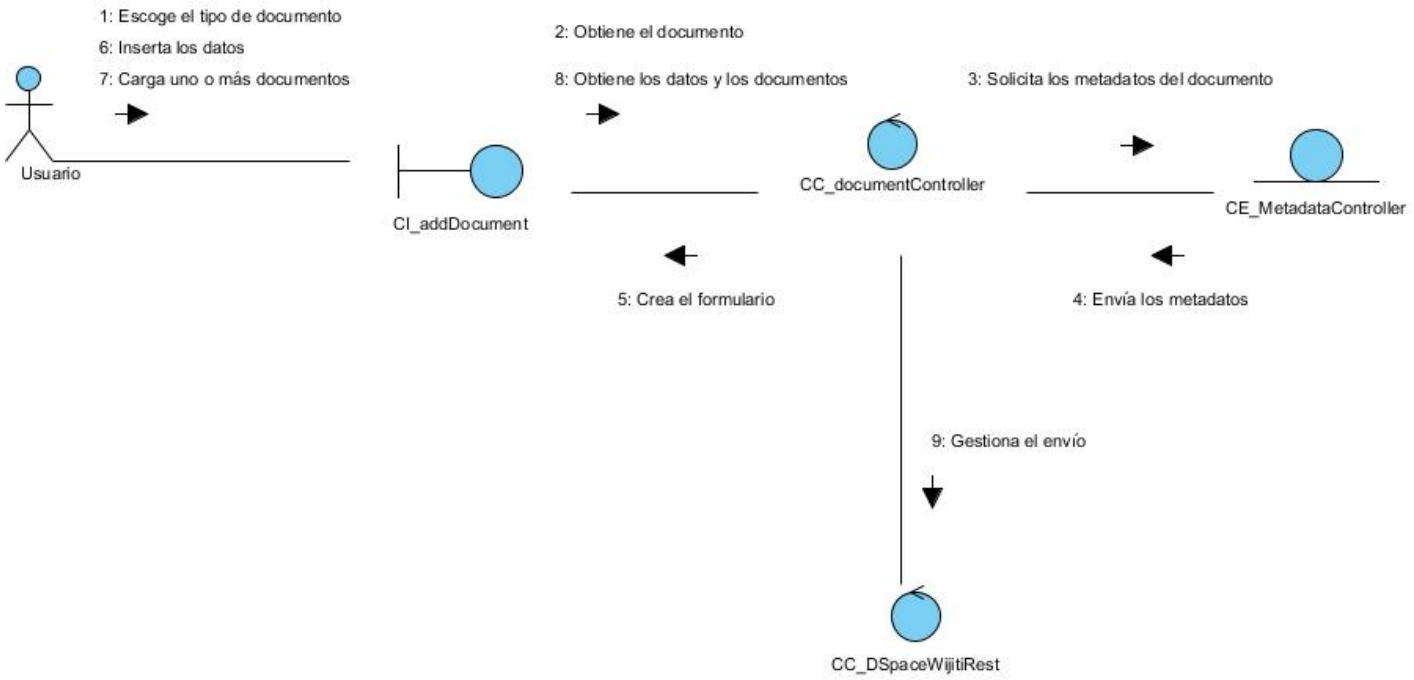


Figura 10: Diagrama de colaboración Enviar Documento.

2.7.3. Diagrama de paquetes.

Los diagramas de paquetes se utilizan para reflejar la organización de paquetes y sus elementos. Cuando se usan para representaciones, los diagramas de paquete de los elementos de clase proveen una visualización de los espacios de nombres. Los usos más comunes para los diagramas de paquete son para organizar diagramas de casos de uso y diagramas de clase. *(31)* Están constituidos por dos tipos de elementos:

- **Paquetes**: Permiten dividir un modelo en partes manejables mediante la agrupación de elementos que pueden ser casos de uso, clases o componentes. Pueden anidar otros paquetes dentro de sí.
- **Dependencias**: Indican que un elemento de un paquete requiere a otro de otro paquete distinto.

A continuación se presenta el diagrama de paquete correspondiente al Módulo Depósito de Documentos para REPXOS 3.0.

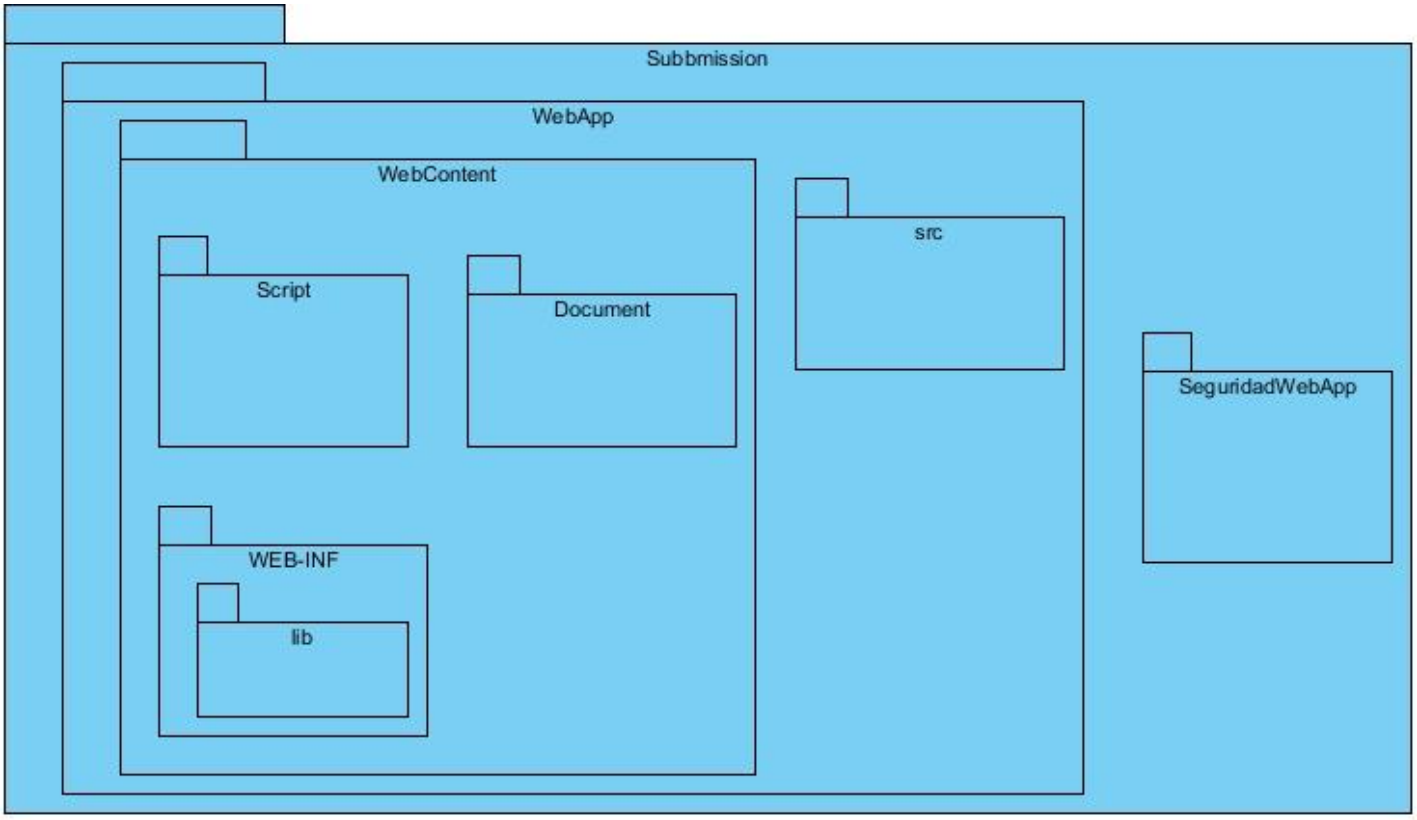


Figura 11: Diagrama de Paquetes

A continuación se muestra el propósito de cada paquete para lograr un mejor entendimiento.

- **Document**: Contiene las vistas del Módulo Depósito de Documentos.
- **lib:** Paquete que contiene las librerías utilizadas por el módulo.
- **Script***:* Contiene las clases para gestionar los metadatos de los formularios y sus validaciones.
- **src:** Es el paquete que contiene las clases para el proceso de envío y el control de los formularios.
- **SeguridadWebApp:** Es el paquete que garantiza la seguridad del módulo.
- **Submission:** Es el paquete que contiene el Módulo Depósito de Documentos.
- **WEB-INF:** Es el paquete principal que contiene los elementos de configuración que permiten ejecutar las operaciones en el sistema.

- **WebApp:** Contiene las principales clases del sistema.
- **WebContent**: Contiene las vistas y los elementos de configuración.

2.8. Patrones utilizados.

2.8.1. Patrón de arquitectura MVC.

El patrón MVC (Modelo – Vista – Controlador) separa a diseñadores (autores del HTML) de los programadores. Los diseñadores pueden cambiar el aspecto de una página sin que los programadores tengan que cambiar o recompilar el código, porque se separan la lógica del uso (programas de Java) y el diseño de la página.

Modelo: Representa la parte de la aplicación que implementa la lógica de negocio. Esto significa que es responsable de la recuperación de datos convirtiéndolos en conceptos significativos para la aplicación, así como su procesamiento, validación, asociación y cualquier otra tarea relativa a la manipulación de dichos datos. A primera vista los objetos del modelo pueden ser considerados como la primera capa de la interacción con cualquier base de datos que podría utilizar la aplicación. Pero en general representan los principales conceptos en torno a los cuales se desea implementar un programa. Un ejemplo de una clase del modelo, en el módulo Depósito de Documentos es metadatacontroller.class.

Vista: Hace una presentación de los datos del modelo estando separada de los objetos del modelo. Es responsable del uso de la información de la cual dispone para producir cualquier interfaz de presentación de cualquier petición que se presente. La capa de la vista no se limita únicamente a HTML o texto que represente los datos, sino que puede ser utilizada para ofrecer una amplia variedad de formatos en función de sus necesidades tales como videos, música, documentos y cualquier otro formato. Un ejemplo de la vista es adddocument.jsp.

Controlador: Gestiona las peticiones de los usuarios. Es responsable de responder la información solicitada con la ayuda tanto del modelo como de la vista. Los controladores pueden ser vistos como administradores cuidando de que todos los recursos necesarios para completar una tarea se deleguen a los trabajadores más adecuados. Espera peticiones de los clientes, comprueba su validez de acuerdo a las normas de autenticación o autorización, delega la búsqueda de datos al modelo y selecciona el tipo de respuesta más adecuado según las preferencias del cliente. Finalmente delega este proceso de presentación a la capa de la vista. Es decir el

controlador es un bloque de código que realiza llamadas al modelo para obtener los datos y se los pasa a la vista para que los muestre al usuario. (32) Un ejemplo en el Módulo Depósito de Documentos es DocumentController.class.

2.8.2. Patrón de diseño GRASP

Los patrones de diseño **GRASP** (*General Responsibility Assignment Software Patterns*) son patrones generales de software para asignación de responsabilidades. Aunque se considera que más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software. (30)

Patrón experto: Es el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. Esto permite que los sistemas tiendan a ser más fáciles de entender, mantener y ampliar además presenta la oportunidad de reutilizar los componentes en futuras aplicaciones. (33)

En el Módulo Depósito de Documentos se pone de manifiesto, cuando la responsabilidad de gestionar el envío de documentos lo tiene la clase DspaceWijitiRest.class, de esta forma la clase se comunica mediante un formato de intercambio de información JSON con REPXOS 3.0.

Patrón controlador: La mayor parte de los sistemas reciben eventos de entrada externa, los cuales generalmente incluyen una interfaz gráfica para el usuario operado por una persona. En todos los casos, si se recurre a un diseño orientado a objetos, hay que elegir los controladores que manejen esos eventos de entrada. Este patrón ofrece una guía para tomar decisiones apropiadas que generalmente se aceptan. La misma clase controlador debería utilizarse con todos los eventos sistémicos de un caso de uso, de modo que se pueda conservar la información referente al estado del caso. (33)

Un controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación. En el Módulo Depósito de Documentos un ejemplo de controlador es la clase DocumentController.class donde se declaran los principales métodos para gestionar los formularios.

Alta cohesión: Caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un gran trabajo. (34)

En el Módulo Depósito de Documentos se pone de manifiesto este patrón, por tener cada clase una función única definida.

Bajo acoplamiento: El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. El acoplamiento bajo significa que una clase no depende de muchas clases. (34)
En el Modulo Depósito de Documentos las clases fueron implementadas de forma independiente, posibilitando que en el caso de realizarse modificaciones en algunas de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

2.8.3. Patrón de diseño Singleton.

El propósito del patrón Singleton es asegurar que una clase tiene una única instancia y proporciona un punto de acceso global a la misma. Este se usa cuando debe haber exactamente una instancia de una clase, que debe ser accesible a los clientes a través de un punto de acceso conocido. La instancia única puede tener subclases y los clientes deben ser capaces de usar las subclases sin modificar su propio código. (35)

En el Módulo Depósito de Documentos se utiliza este patrón, en la clase HibernateHelper.class, esta es responsable de crear una única instancia. Permite el acceso global a dicha instancia y se declara el constructor de la clase como privado para que no sea instanciable directamente.

Conclusiones.

Durante el análisis y diseño en base a la metodología RUP, patrones de diseño y de la arquitectura definida se puede concluir que se definieron un total de seis requisitos funcionales, lo cuales fueron agrupados en tres Casos de Uso (CU). A partir de la obtención de los CU se elaboró el diagrama de CU y se describieron textualmente para lograr una mejor comprensión. Se definieron los requisitos no funcionales del sistema. Los mismos fueron elaborados teniendo en cuenta capacidad, recursos, lenguaje de programación, hardware, software y seguridad del sistema. Se construyeron los artefactos fundamentales del flujo de trabajo del diseño. Entre ellos se encuentra el diagrama de clases del diseño y el diagrama de paquetes, estos a la vez mostraron una propuesta para el desarrollo del módulo. Además dichos artefactos sirvieron de base para la implementación de sistema propuesto. La utilización de patrones de diseños conocidos y la ayuda del patrón arquitectónico MVC (Modelo-Vista-Controlador) logró la elaboración de un sistema sencillo para los usuarios y de fácil mantenimiento para los programadores, aspecto importante para la utilización del sistema.

CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBA

Introducción

En este capítulo se abordan temas referentes a los flujos de trabajo de implementación y prueba, los cuales son importantes para el proceso de desarrollo de software. Para ello, en la implementación se elaboran los diagramas de componentes y despliegue. Se dan a conocer los estándares de codificación, importantes para la elaboración del Módulo de Depósito de Documentos. Se muestran las pruebas realizadas, utilizando el método de Caja Negra y la técnica Partición de Equivalencia.

3.1. Implementación

El modelo de implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes se pueden encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. (36)

3.1.1. Diagrama de Componente

El diagrama de componente es una parte física de un sistema (módulo, base de datos, programa ejecutable). Se puede decir que un componente es la materialización de una o más clases, porque una abstracción con atributos y métodos pueden ser implementados en los componentes, es donde se modela la estructura de implementación de la aplicación, su organización en componentes y su despliegue en nodos de ejecución. (37) A continuación se muestra el diagrama de componente del Módulo Depósito de Documentos, elaborado según las capas del patrón arquitectónico utilizado.

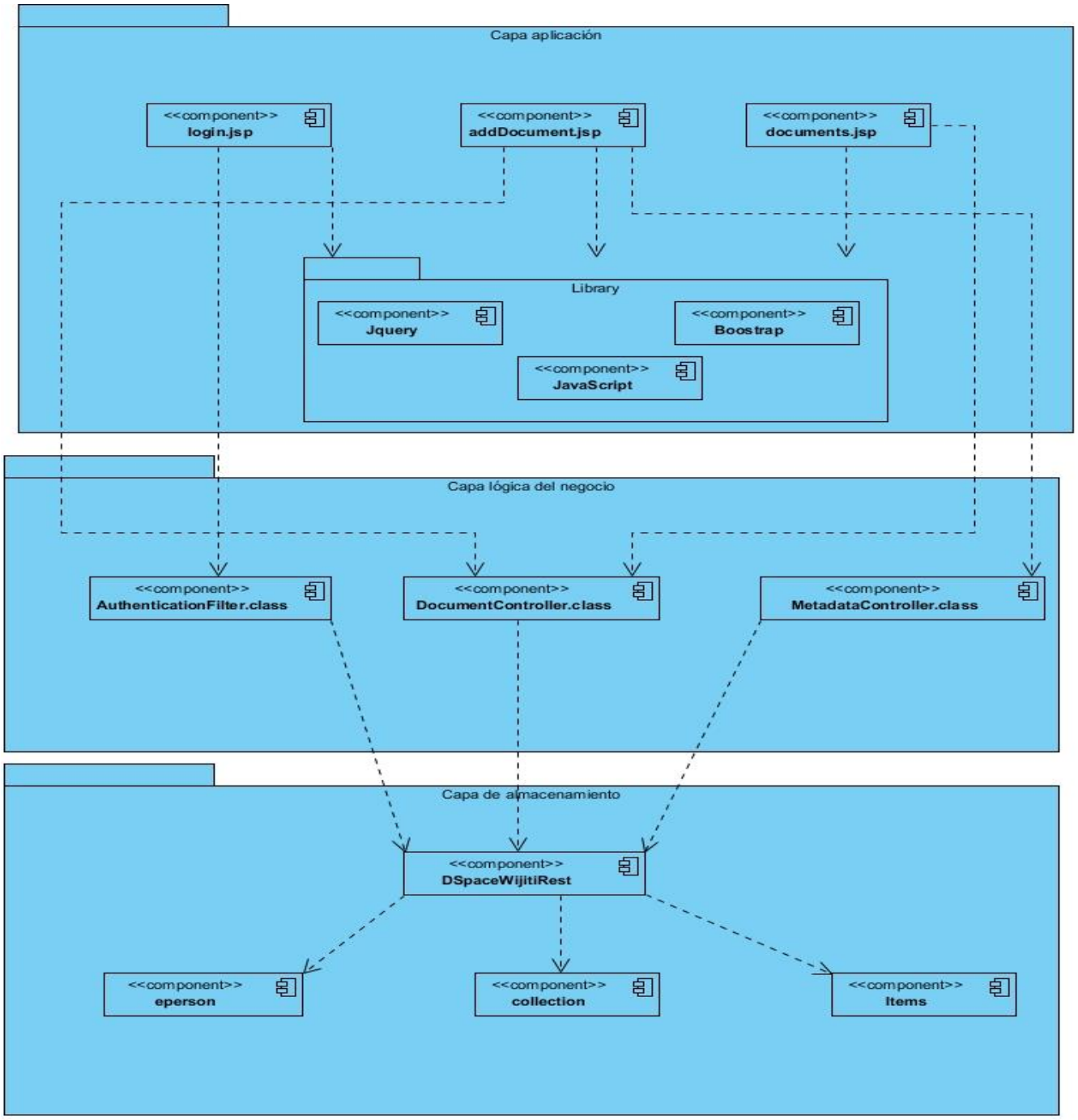


Figura 12: Diagrama de componente del Módulo Depósito de Documentos.

3.1.2. Diagrama de Despliegue

Un diagrama de despliegue modela la arquitectura en tiempo de ejecución de un sistema. Este muestra la configuración de los elementos de *hardware* (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos (38)

A continuación se muestra el diagrama de despliegue del Módulo Depósito de Documentos para el sistema de repositorios digitales REPXOS 3.0.

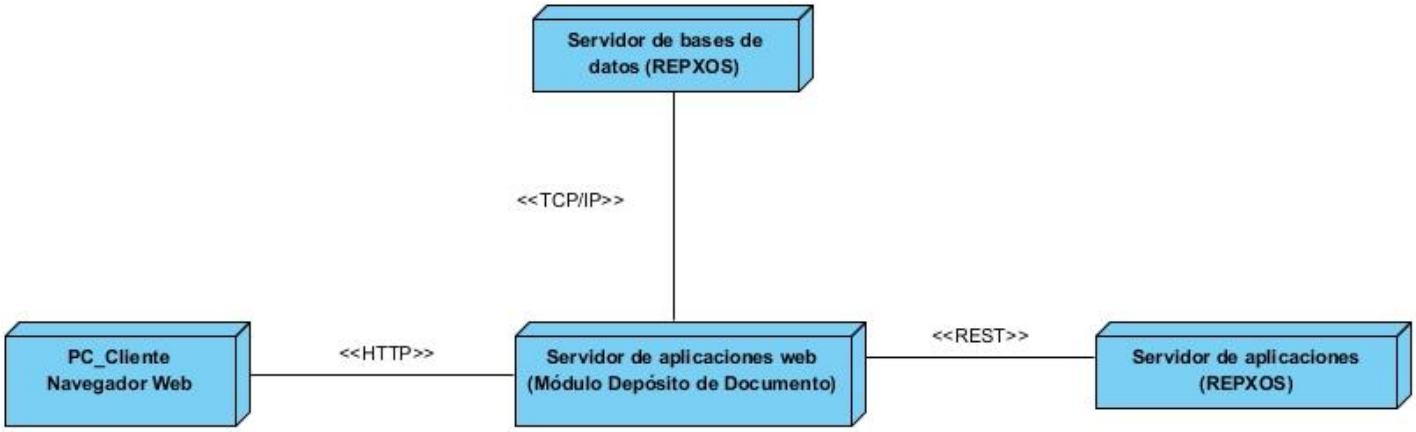


Figura 13: Diagrama de Despliegue.

Componentes o nodos que interactúan en el despliegue del sistema:

PC_Cliente: representa las computadoras clientes que se conectan al servidor de aplicaciones mediante el protocolo HTTP.

Servidor de Aplicaciones Web: se encarga de atender las peticiones del usuario y proporcionar respuestas a este. Es el servidor donde se despliega la aplicación web. Este accede al servidor de base datos de REPXOS para el manejo de la información mediante el protocolo TCP/IP y accede al servidor de aplicaciones de REPXOS mediante REST para enviar documentos y sus metadatos.

Servidor de Bases de Datos DSpace: se utiliza para obtener los usuarios con permisos de acceso al módulo.

Servidor de Aplicaciones de DSpace: es donde se encuentra el sistema REPXOS, que se encarga de almacenar los documentos y metadatos enviados.

3.1.3 Estándares de codificación

Al comenzar un proyecto de software, se debe establecer un estándar de codificación para asegurar que todos los programadores del proyecto trabajen de forma coordinada. A continuación se muestran algunas pautas del estándar de codificación de java, utilizadas en la implementación del Módulo Depósito de Documentos.

- ✓ Utilizar nombres en inglés para las clases y métodos.
- ✓ Dejar espacio de separación dentro del código, para que sea entendible.
- ✓ Utilizar guion bajo ("_") y escribir en inglés los nombres de las variables compuestas. Ejemplo: doc_type.
- ✓ Los nombres de las variables deben ser cortos y significativos.
- ✓ Los nombres de variables declaradas como constante deben ser todas en mayúsculas con palabras separadas por guion bajo ("_"). Ejemplo: HTML_OK_RESPONSE_CODE.
- ✓ Los nombres de clases tipo **DAO** deberán tener el sufijo "DAO", como por ejemplo CollectionDAO.

3.2. Pruebas

Uno de los instrumentos adecuados para determinar la calidad de un producto software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos. En las pruebas se usan casos de prueba, especificados de forma estructurada mediante técnicas de prueba. (39)

3.2.1 Estrategia de prueba

Las pruebas juegan un papel muy importante para el desarrollo de software, permitiendo detectar y corregir errores desde etapas tempranas. Para determinar la calidad de software se aplican medidas o pruebas que permitan comprobar el grado de cumplimiento respecto de las especificaciones y funcionalidades iniciales del sistema. Las pruebas son aplicadas en diferentes niveles de trabajo, agrupadas por niveles de pruebas, aplicadas por etapas.

A continuación se muestran los diferentes niveles de prueba: (40)

Prueba de Unidad: se concentra en el esfuerzo de verificación de la unidad más pequeña de diseño del software: el componente o módulo del software. Las pruebas de unidad se concentran en la lógica del procesamiento interno y en la estructura de datos dentro de los límites de un componente. (41)

Prueba de Integración: son realizadas sobre agrupaciones de componentes/funciones. Se examinan las interfaces para asegurar que los componentes individuales son llamados cuando es necesario y que los datos que se transmiten entre dichos componentes son los requeridos. Asegura que no hay errores de interfaces y encuentra defectos en el sistema. (42)

Prueba del Sistema: verifican el comportamiento del sistema en su conjunto. Se prueban interfaces externas con otros sistemas, utilidades, unidades físicas y entorno operativo. (41)

Pruebas de Aceptación: su propósito es certificar que los requisitos del cliente se han cumplido satisfactoriamente y por lo tanto se puede iniciar la producción. Deben estar a cargo de las pruebas un equipo independiente del desarrollo. (42)

Pruebas de Regresión: aseguran que los cambios en el software no introducen nuevos defectos. Se ejecutan luego de modificar el sistema. Consisten en repetir las pruebas que ya se han hecho al sistema en una versión anterior, y comparar el resultado actual con el anterior. (42)

De los niveles antes mencionados son utilizadas las pruebas de **integración** para verificar que las especificaciones funcionales están implementadas correctamente; las del **sistema** para comprobar los requisitos no funcionales; las pruebas de **aceptación** para examinar el comportamiento del sistema frente a los requisitos del cliente y las de **regresión** para detectar los fallos del sistema durante la modificación de este.

Pruebas de Integración

Se realizaron pruebas de integración mediante el diseño de casos de pruebas (**DCP**). Se utilizó el método de Caja Negra, introduciendo un conjunto de entradas y resultados esperados. Las pruebas resultaron satisfactorias.

DCP: Integrar Sistema.

Escenario	Descripción	Respuesta del sistema
EC 1.1 Enviar documento.	Se realiza el envío de documentos a través del estilo de arquitectura REST hacia el sistema DSpace.	Muestra el mensaje "Se ha enviado el documento".

EC 1.2 Usuario o contraseña inválida.	Un actor especificado desea acceder al sistema, el mismo introduce su usuario o contraseña incorrectos.	Muestra el mensaje “ Usuario o contraseña incorrectas”
EC 1.3 No se adjunta un archivo.	Se comete un error al realizar el envío de documentos a través del estilo de arquitectura REST hacia el sistema Dspace.	Muestra el mensaje "Debe adjuntar un archivo".

Tipos de pruebas

Existen varios tipos de pruebas, cada una tiene un objetivo específico y una técnica, dentro de estas la que se le realiza al sistema es la que evalúa la funcionalidad y el rendimiento del mismo. En este trabajo se realizan las pruebas funcionales y dentro de ellas la siguiente:

- ✓ **Pruebas Funcionales**
  - **Función:** Las pruebas de función fijan su atención en la validación de las funciones, métodos, servicios y casos de usos.
- ✓ **Pruebas de Rendimiento**

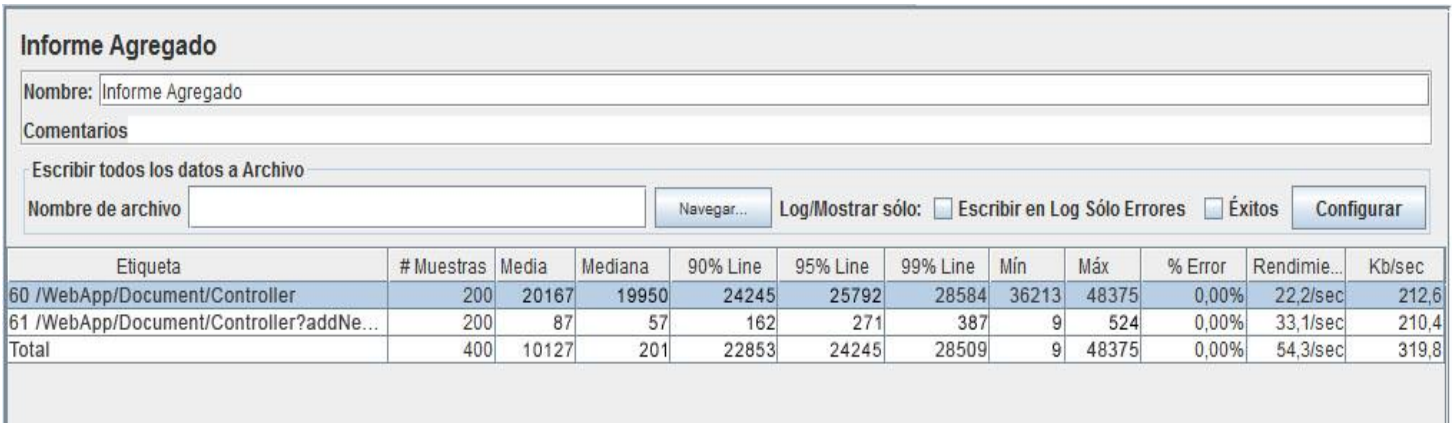
El servicio de pruebas de rendimiento de software se centra en determinar la velocidad con la que el sistema bajo pruebas realiza una tarea en las condiciones particulares del escenario de pruebas. El objetivo de estas pruebas es determinar si el usuario está satisfecho con la velocidad de la aplicación, bajo condiciones de uso esperadas. Existen tres tipos de pruebas de rendimiento que se pueden realizar: carga, stress y volumen.

Al Módulo Depósito de Documentos se le aplicará específicamente las pruebas de carga. Estas fueron realizadas con la herramienta apache JMeter 2.13.

- ✓ **JMeter:** es una herramienta *Open Source* (Código Abierto) realizada en java, que se utiliza para realizar pruebas de rendimiento, normalmente contra aplicaciones web. Permite realizar simulaciones de gran carga en el servidor, red o aplicación para comprobar su “fuerza” y para analizar el rendimiento ante diferentes tipos de sobrecarga. (44)

Resultados de las Pruebas de Carga

Para la realización de las pruebas de carga se introducen 200 usuarios o grupos de hilos, en un período de subida de 1 segundo. Se obtiene como resultado 0.00% de error. Los resultados arrojaron además que el rendimiento se comportó con un total de 54,3 peticiones por segundo. De esta forma se demuestra que el Módulo Depósito de Documentos soporta 200 usuarios concurrentemente.



Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimie...	Kb/sec
60 /WebApp/Document/Controller	200	20167	19950	24245	25792	28584	36213	48375	0,00%	22,2/sec	212,6
61 /WebApp/Document/Controller?addNe...	200	87	57	162	271	387	9	524	0,00%	33,1/sec	210,4
Total	400	10127	201	22853	24245	28509	9	48375	0,00%	54,3/sec	319,8

✓ *Figura 14: Pruebas de carga.*

Método de prueba: Caja Negra

Este método se utiliza para realizar las pruebas al Módulo Depósito de Documentos. Se lleva a cabo sobre la interfaz del software, su objetivo principal es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene. (43)

Técnica de Prueba

Dentro del método de **Caja Negra** se utiliza la técnica “partición de equivalencia”, esta permite examinar los valores válidos e inválidos de las entradas existentes en el software. Descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. (43)

3.2.2. Resultado de las pruebas

Luego de haber realizado todas las pruebas mencionadas anteriormente, se obtuvieron los siguientes resultados:

Caja Negra

En la siguiente tabla se muestran los resultados de las pruebas de Caja Negra realizadas al Módulo Depósito de Documentos.

No.Iteración	Aplicación			Casos de Prueba	Resueltas	No procede	Total NC
	Alta	Media	Baja				
1ra Iteración	0	2	1	4	7	0	7
2da Iteración	0	0	0	0	0	0	0

Tabla 4: Resultados de las pruebas de Caja Negra.

Se realizaron 2 iteraciones de pruebas. Las principales causas de no conformidades detectadas fueron:

- Las variables en el escenario (1.1) del Caso de Prueba Enviar Documentos todas deben estar en estado válido (V).
- Las variables en el escenario campos inválidos del Caso de Prueba Enviar Documento deben ponerse con valor inválido (I).
- La descripción del escenario debe estar en correspondencia con el nombre del escenario.
- El flujo central del Caso de Prueba Autenticar Usuario debe estar ordenado por pasos y enumerados ascendentemente.
- Debe mostrarse un mensaje preguntando al usuario si desea eliminar el formulario. Caso de Prueba Gestionar Formulario.
- El mensaje que muestra la aplicación “No ha cargado ningún archivo” debe aparecer en la interfaz correspondiente a la pestaña Archivo y no en la interfaz principal del módulo. Caso de Prueba Enviar Documento.
- El mensaje que se muestra en la aplicación correspondiente al Caso de Prueba Enviar documento “Se ha enviado correctamente el documento” posee dos puntos finales.

Se realizaron dos iteraciones de pruebas. En la primera iteración se detectaron 3 no conformidades (NC) de aplicación y 4 de casos de pruebas. De ellas fueron resueltas las 7 NC satisfactoriamente. En la segunda iteración se detectaron 0 NC. De esta forma se garantiza un mejor funcionamiento del Módulo Depósito de Documentos.

Conclusiones

Se puede concluir que con el desarrollo de este capítulo, se obtuvieron los diagramas de componente y despliegue. De esta forma se reflejaron las relaciones establecidas entre los elementos físicos que conforman el sistema y quedó definida la distribución del sistema. Se definieron pautas del estándar de codificación de java, para lograr una mejor comprensión del código del Módulo Depósito de Documentos. Se traza una guía con las estrategias de pruebas necesarias para adecuar las pruebas al entorno y la arquitectura del software a probar.

CONCLUSIONES

Con la realización de este trabajo se satisfacen en su totalidad los objetivos trazados, por lo que se puede llegar a las siguientes conclusiones:

- ✓ Con el estudio del marco teórico se evidenciaron características de los sistemas de depósito de documentos existentes tanto nacionales e internacionales, permitiendo afirmar que ninguno de ellos constituye una solución factible para darle solución a la problemática planteada.
- ✓ Se diseñó el Módulo Depósito de Documentos, mediante el cual se generaron los artefactos fundamentales del flujo de trabajo del diseño, propuestos por la metodología de desarrollo RUP.
- ✓ Se implementaron todas las funcionalidades definidas para el Módulo Depósito de Documentos, utilizando para ello las herramientas y lenguajes seleccionados.
- ✓ Se realizaron pruebas a las funcionalidades del Módulo Depósito de Documentos, para verificar la calidad de la solución propuesta. Se comprobó que el software cumple con todas las funcionalidades definidas.
- ✓ Se desarrolló el Módulo Depósito de Documentos, el cual permite a los usuarios del sistema para repositorios digitales REPXOS 3.0 la configuración y validación de los metadatos, para lograr un correcto depósito, y además la solución puede ser utilizada según las necesidades la institución.

RECOMENDACIONES

Se recomienda para versiones posteriores del Módulo Depósito de Documentos:

- ✓ Permitir que los usuarios se registren y obtengan sus colecciones mediante servicios web REST, sin necesidad de utilizar la base de datos del sistema DSpace.
- ✓ Agregar el protocolo SWORD para realizar envíos al sistema.
- ✓ Incorporar al módulo el Control de Autoridades.

REFERENCIAS BIBLIOGRÁFICAS

1. **Repositorios digitales. [En línea] [Citado el: 05 de Junio de 2015.] http://bibliotecabiologia.usal.es/tutoriales/catalogos-repositorios-bibliosvirtuales/repositorios_digitales.html.**

2. Aroca Cámara, Mercedes, Checa Claudel, José y Guzmán Pérez, Catalina. Helvia: La apuesta de la Universidad de Córdoba por el conocimiento abierto. [En línea] [Citado el: 12 de Febrero de 2015.] http://helvia.uco.es/xmlui/handle/10396/2344.

3. Tipos de repositorios. [En línea] [Citado el: 05 de Junio de 2015.] http://www.ssoar.info/es/home/sobre-el-acceso-abierto/tipos-de-repositorios.html.

4. Sánchez Tarragó, Nancy . El movimiento de acceso abierto a la información y las políticas nacionales e institucionales de autoarchivo. [En línea] [Citado el: 12 de Febrero de 2015.] http://bvs.sld.cu/revistas/aci/vol16_3_07/aci05907.html.

5. Lamarca Lapuente,María Jesús. EL NUEVO CONCEPTO DE DOCUMENTO EN LA CULTURA DE LA IMAGEN. [En línea] [Citado el: 04 de Junio de 2015.] http://www.hipertexto.info/documentos/metadatos.htm.

6. Hillmman,Diane. Using Dublin Core. [En línea] [Citado el: 14 de Mayo de 2015.] http://dublincore.org/documents/usageguide/.

7. Definición abc. [En línea] [Citado el: 04 de Junio de 2015.] http://www.definicionabc.com/general/formulario.php.

8. The Free Dictionary. [En línea] [Citado el: 05 de Junio de 2015.] http://es.thefreedictionary.com/deposit%C3%B3.

9. Navarro Marset, Rafael. REST vs Web Services - RestVsWebServices.pdf. [En línea] [Citado el: 20 de Marzo de 2015.] http://users.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf.

10. Allinson, Julie y Sebastien, François. *SWORD Ariadne Jan 2008.pdf.*

11. Ishida y Richard. Ejecución de HTML & XHTML. [En línea] [Citado el: 25 de Abril de 2015.] http://www.w3.org/International/articles/serving-xhtml/Overview.es.php.

12. arXiv.org. [En línea] [Citado el: 11 de Febrero de 2015.] http://arxiv.org/help/submit.

13. ¿Qué es PubMed Central? — Centro de Ayuda de la Biblioteca Virtual en Salud. [En línea] 11 de Febrero de 2015. http://bvsayuda.sld.cu/ayudas/faq/bfque-es-pubmed-central/.

14. EasyDeposit. [En línea] [Citado el: 06 de Junio de 2015.] http://easydeposit.swordapp.org/about/.

15. Conferencia de Acceso Abierto. [En línea] [Citado el: 06 de Junio de 2015.] http://www.acessoaberto.pt/c/index.php/confoa2013/2013/paper/view/330.

16. BibApp. [En línea] [Citado el: 06 de Junio de 2015.] http://bibapp.org/.

17. Flores López , Mónica y Santiago, María de Lourdes . Desarrollando aplicaciones informáticas con el Proceso de Desarrollo Unificado (RUP). [En línea] [Citado el: 11 de Febrero de 2015.] http://www.utvm.edu.mx/OrganoInformativo/orgJul07/RUP.htm.

18. Alvarez Fernández, Luis Carlos y Valdés Inerarte, Eduardo . Repositorio Institucional. [En línea] 17 de Febrero de 2015. http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_03552_10/1/TD_03552_10.pdf.

19. Sobre PostgreSQL \textbar www.postgresql.org.es. [En línea] 16 de Febrero de 2015. http://www.postgresql.org.es/sobre_postgresql.

20. Hibernate ORM - Hibernate ORM. [En línea] [Citado el: 18 de Mayo de 2015.] http://hibernate.org/orm/.

21. Eclipse. [En línea] 21 de Marzo de 2105. https://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/lunasr2.

22. Definición Tomcat, Apache Tomcat, Jakarta Tomcat Enciclopedia Proyecto AjpdSoft. [En línea] [Citado el: 21 de Febrero de 2015.] http://www.ajpdsoft.com/modules.php?name=Encyclopedia&op=content&tid=769.

23. JAVA. [En línea] 11 de Febrero de 2015. http://www.infor.uva.es/ jmrr/tgp/java/JAVA.html.

24. Principales Características de Java. [En línea] 18 de Febrero de 2015. http://personales.upv.es/rmartin/cursoJava/Java/Introduccion/PrincipalesCaracteristicas.htm.

25. Qué es JAVA Script ? [En línea] 11 de Febrero de 2015. http://www.pablin.com.ar/computer/cursos/cursojs/js1.htm.

26. Rumbaugh, James , Jacobson, Ivar y Booch, Grady. *El Lenguaje Unificado de Modelado.*

27. clase6_AJ2010.pdf. [En línea] [Citado el: 17 de Marzo de 2015.] http://ldc.usb.ve/~martinez/cursos/ci3715/clase6_AJ2010.pdf.

28. Sommerville, Ian. *Sommerville_Parte_II_Requerimientos.*

29. Microsoft Word - Modelo Diseño.docx - ModeloDiseno.pdf. [En línea] [Citado el: 17 de Marzo de 2015.] http://is.ls.fi.upm.es/docencia/is2/documentacion/ModeloDiseno.pdf.

30. Larman, Craig. *UML y Patrones. Introducción al análisis y diseño orientado a objeto. Mexico : Prentice Hall,1999. ISBN/970-17-0261-1.*

31. Sparx Systems - Tutorial UML 2 - Diagrama de Paquete. [En línea] [Citado el: 17 de Marzo de 2015.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_packagediagram.html.

32. Entendiendo el Modelo - Vista - Controlador — documentación de CakePHP Cookbook - 2.x. [En línea] [Citado el: 7 de Febrero de 2015.] http://book.cakephp.org/2.0/es/cakephp-overview/understanding-model-view-controller.html.

33. Visconti, Marcello y Astudillo, Hernan . Patrones.pdf. [En línea] 25 de Marzo de 2015. http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.

34. Pavón Mestras, Juan. *Patrones de diseño orientado a objetos.*

35. Guerra Sánchez,Esther . 06 Creacion.pdf. [En línea] [Citado el: 18 de Mayo de 2015.] http://arantxa.ii.uam.es/~eguerra/docencia/0809/06%20Creacion.pdf.

36. Rumbaugh, James y Grady Jacobson, Booch. El Proceso Unificado de Desarrollo de Software. 2000. ISBN 84-7829-036-2. [En línea] [Citado el: 16 de Abril de 2015.]

37. Quispe Cruz, Fabio Victor y Mamani Gutiérrez, Dino Ever. componen 1. [En línea] 16 de Abril de 2015.

38. Sparx Systems - Tutorial UML 2 - Diagrama de Despliegue. [En línea] [Citado el: 16 de Abril de 2015.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html.

39. Las pruebas de software forman el único instrumento adecuado para determinar la calidad de software. [En línea] [Citado el: 03 de Mayo de 2015.] http://www.pruebasdesoftware.com/laspruebasdesoftware.htm.

40. Microsoft Word - Dinámicas.doc - get.php. [En línea] [Citado el: 06 de Mayo de 2015.] http://www.lsi.us.es/docencia/get.php?id=361.

41. Blanco Bueno, Carlos. Tema01. Construcción y Pruebas de Software. [En línea] [Citado el: 12 de Junio de 2015.] http://ocw.unican.es/ensenanzas-tecnicas/ingenieria-del-software-ii/materiales/tema1-pruebasSistemasSoftware.pdf.

42. Ochoa, Sergio. CC51A – Ingeniería de Software. Pruebas de Software. [En línea] [Citado el: 12 de Junio de 2015.] https://www.u-cursos.cl/ingenieria/2005/1/CC51A/1/material_docente/bajar?id_material=65585.

43. Disciplina de prueba. [En línea] [Citado el: 11 de Mayo de 2015.]

44. Marco de Referencia. [En línea] [Citado el: 13 de Junio de 2015.] http://cdigital.udem.edu.co/TESIS/CD-ROM55662010/07.Capitulo2.pdf.

45. ACIMED - Opendoar: un directorio de repositorios institucionales. [En línea] [Citado el: 11 de Febrero de 2015.] http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1024-94352008000700010.

46. Fedora. [En línea] [Citado el: 10 de Marzo de 2015.] http://www.fedora-commons.org/about.

47. Eclipse. [En línea] [Citado el: 21 de Marzo de 2015.] https://www.eclipse.org/ide/.

48. PubMed, Contenido, Características, Las interfaces alternativos. [En línea] 11 de Febrero de 2015. http://lasaludfamiliar.com/caja-de-cerebro/conocimiento-9120.html.

49. *CIGED_eXcriba_Plantilla_manual_usuario.pdf.*

50. Ibarias Cruz, Dario. *¿Qué es un formulario?*

51. UAB. [En línea] [Citado el: 05 de Junio de 2015.] https://www.uab.cat/servlet/Satellite?c=Page&cid=1260947893056&pagename=BibUAB%2FPage%2FTemplatePlanaBibUAB.

52. Dspace. [En línea] [Citado el: 14 de Febrero de 2015.] http://www.dspace.org.

53. Barton, Mary y Waters, Margaret. Como crear un repositorio institucional. Manual LEADIRS II. [En línea] [Citado el: 10 de Marzo de 2015.] http://www.r020.com.ar/docs.php?id=1067.

54. Dspace, la plataforma lider de código abierto para los Repositorios. [En línea] [Citado el: 05 de Junio de 2015.] http://www.xercode.es/productos/dspace.

55. Beatriz LLanusa Ruiz, Susana. Evaluación del Programa de Introducción de Tecnologías de Información y Comunicación en la Atención Primaria de Salud. [En línea] http://www.sld.cu/galerias/pdf/sitios/revsalud/tesis_maestria_susana_llanusa.pdf.

56. Dans Moreno, Juan Javier. ***Modelación e implementación del módulo Documentos del subsistema Depósitos de Aduana.*** **2015.**

ANEXOS

1.1 Anexo.1

CU1. Autenticar Usuario

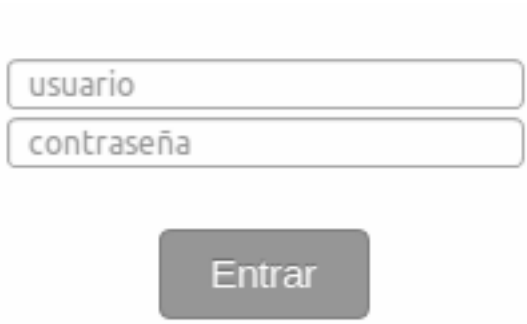
Objetivo	Identificar usuario para la entrada al sistema
Actores	Usuario
Resumen	El caso de uso se inicia cuando el actor especificado desea acceder a al sistema. El mismo introduce su usuario y contraseña y en vista del rol y permisos que tenga definido accede a las funcionalidades específicas según los privilegios que le fueron asignados.
Complejidad	Baja
Prioridad	Media
Precondiciones	El sistema debe estar funcionando correctamente.
Postcondiciones	El usuario accede al sistema.

Flujo de eventos		
Flujo básico Autenticar Usuario		
	Actor	**Sistema**
1.	Accede al sistema	
2.		Brinda la posibilidad de introducir los datos elementales para autenticarse: - Usuario - Contraseña
3.		Permite: • Aceptar
4.	Introduce los datos y presiona el botón aceptar	
5.		Valida los datos.
6.		Termina el caso de uso.
Flujos alternos		
5a Validación de campos vacíos en una consulta.		
	Actor	**Sistema**
1.		Muestra un mensaje de información: “No debe dejar campos en blanco”.
2.		Regresa al paso 4 del flujo básico.
Flujos alternos		
5b El usuario o la contraseña es incorrecta		
	Actor	**Sistema**
		Muestra un mensaje de información: “Datos de inicio de sesión

		incorrectos”.
		Regresa al paso 4 del flujo básico.

Prototipo de interfaz de usuario

Autenticar usuario



CU2. Gestionar Formulario

Objetivo	Crear, actualizar y eliminar formulario.
Actores	Usuario.
Resumen	El caso de uso se inicia cuando el actor desea crear, actualizar o eliminar un formulario. Si el actor selecciona la opción crear, el sistema le permite registrar los datos perteneciente a dicho formulario. Si desea actualizar, el sistema muestra los datos para realizar los cambios. Si el actor selecciona la opción eliminar el sistema elimina el formulario creado para cualquier documento que se desee depositar. Termina el caso de uso.
Complejidad	Media
Prioridad	Media
Precondiciones	-El sistema debe estar funcionando correctamente. -Un usuario se ha autenticado en el sistema.
Postcondiciones	Se creó, actualizó y eliminó un formulario.
Flujo de eventos	
Flujo básico Crear Formulario	

	Actor	**Sistema**
1.	Desea realizar las siguientes acciones	
		Permite realizar las siguientes acciones sobre el formulario. • Crear un formulario. • Actualizar el formulario. Ver Sección 1: “Actualizar

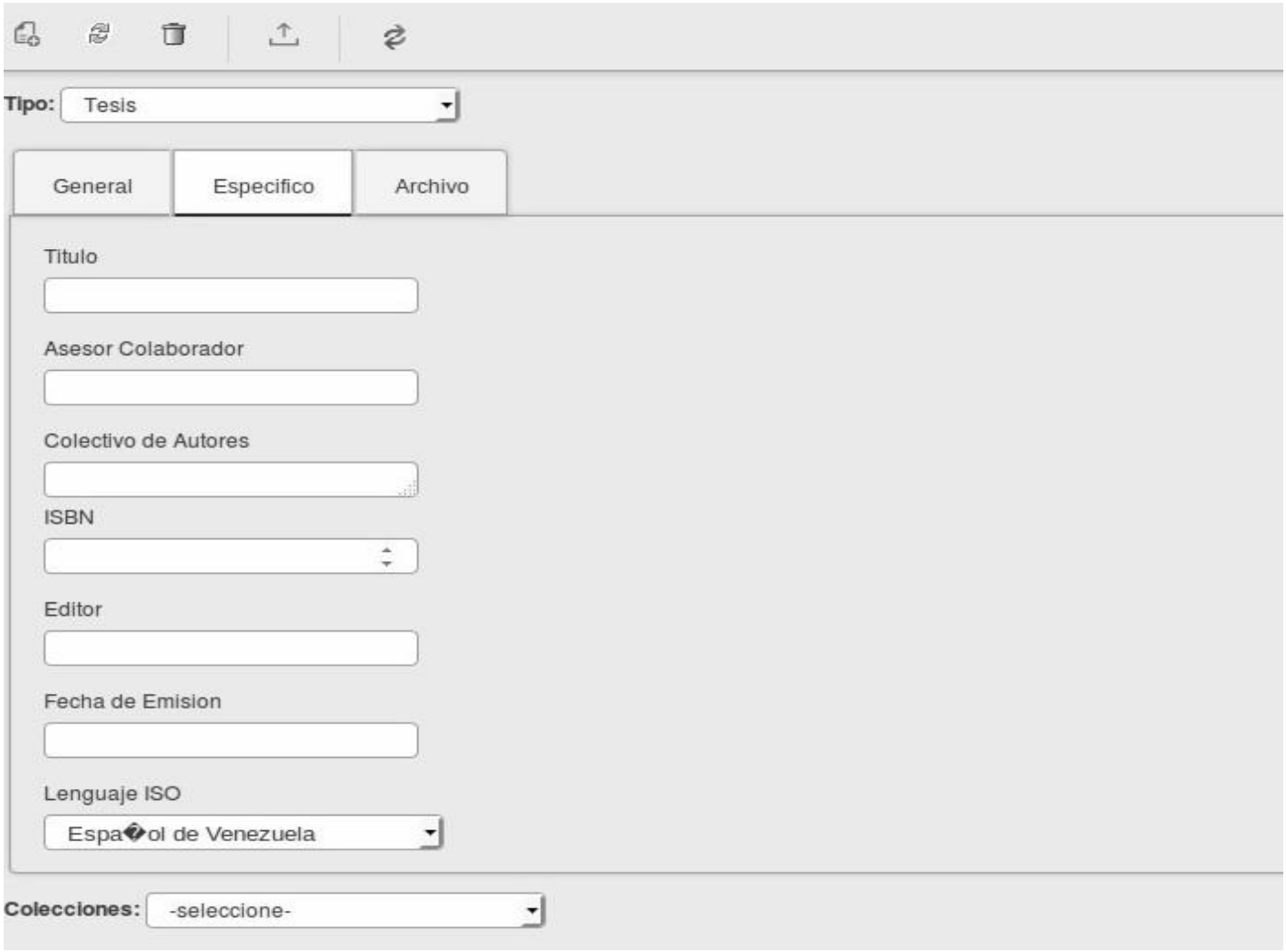
		formulario” • Eliminar formulario. Ver Sección 2: “Eliminar formulario”
2.	Selecciona la opción “Documentos”	
3.		Permite seleccionar la opción “aquí” para crear un formulario.
4.	Selecciona la opción “aquí”	
5.		Permite seleccionar los siguientes tipos de documentos: • Libro, permite introducir los siguientes datos específicos: -Título, Título alternativo, Autor Contribuyente, Editor Contribuyente, Formato de Páginas, Lenguaje ISO. • Tesis, permite introducir los siguientes datos específicos: -Asesor Colaborador, ISBN, Lenguaje ISO. • Revista, permite introducir los siguientes datos específicos: -Cita Bibliográfica, Volúmen, Primera Página, Última Página, ISSN, Fecha de Emisión, Número de Artículo. • Libro y Monografías, permite introducir los siguientes datos específicos: --Título, Título alternativo, Autor Contribuyente, Editor Contribuyente, Formato de Páginas, Lenguaje ISO, Fecha de Emisión • Actas, permite introducir los siguientes datos específicos: -Nombre de Conferencia, Lugar, Titulo, Primera Página, Última Página, ISSN, ISBN. • Reporte de Investigación, permite introducir los siguientes datos específicos: -Número de Páginas, Editor.

		• Patente, permite introducir los siguientes datos específicos: -Autores, Autores corporativos, Título, Fecha, Lugar de publicación, Número de página, Palabras libres, Tipo de patente, Derechos, Número de Patente. • Registro de Producto, permite introducir los siguientes datos específicos: -Autores, Autores corporativos, Título, Fecha, Lugar de publicación, Número de Páginas, Tipo de registro, Derechos, Número de Registro. Además permite introducir por cada tipo de documento los siguientes datos generales: • Autor • Autores corporativos • Título • Título alternativo • Lenguaje ISO • Fecha de publicación Permite seleccionar las colecciones a la que se desea enviar.
6.	Selecciona el tipo de documento.	
7.	Introduce los datos del formulario.	
8.	Selecciona la colección	
9.		Valida que se hayan llenado los campos y que haya seleccionado la colección. Y permite: • Salvar
10.	Selecciona el botón de salvar	
11.		Termina el caso de uso.
Sección 1: "Actualizar Formulario"		
Flujo básico Actualizar Formulario		
	Actor	**Sistema**
1.	Selecciona el botón de los documento en procesos.	

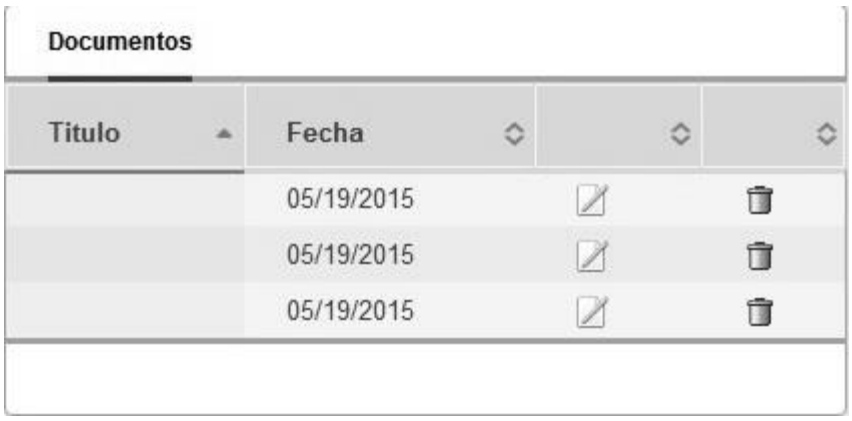
2.		Muestra todos los formularios creados para escoger al que se le desea realizar cambios.
3.	Selecciona el formulario a modificar.	
4.		Permite: • Modificar • Eliminar
5.	Selecciona el botón que permite modificar un formulario.	
6.		Muestra el formulario con los metadatos generales y específicos.
7.	Realiza los cambios en el formulario	
8.		Permite: • Salvar
9.	Selecciona el botón de Salvar.	
10.		Actualiza los cambios en el formulario.
11.		Termina el caso de uso.
12.		
Sección 2: "Eliminar Formulario"		
Flujo básico Eliminar Formulario		
	Actor	**Sistema**
	Selecciona el botón de los documento en procesos.	
		Muestra todos los formularios creados para escoger al que se le desea eliminar.
1.	Selecciona el formulario a eliminar.	
2.		Permite: • Modificar • Eliminar
3.	Selecciona el botón que permite eliminar un formulario.	
4.		Elimina el formulario.
5.		Termina el caso de uso.

Prototipo de interfaz de usuario

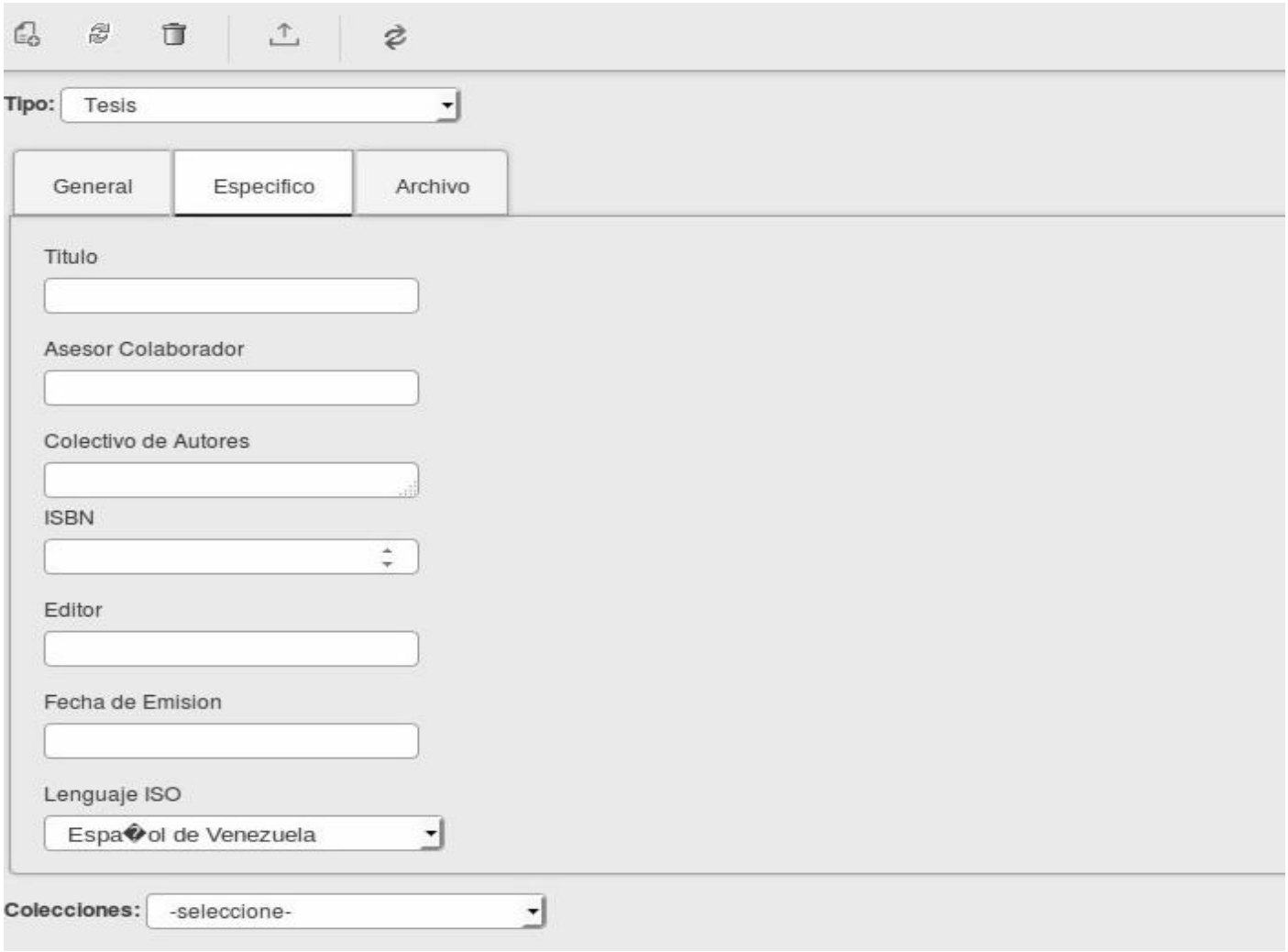
Crear formulario



Eliminar formulario



Modificar formulario



CU3. Enviar Documento.

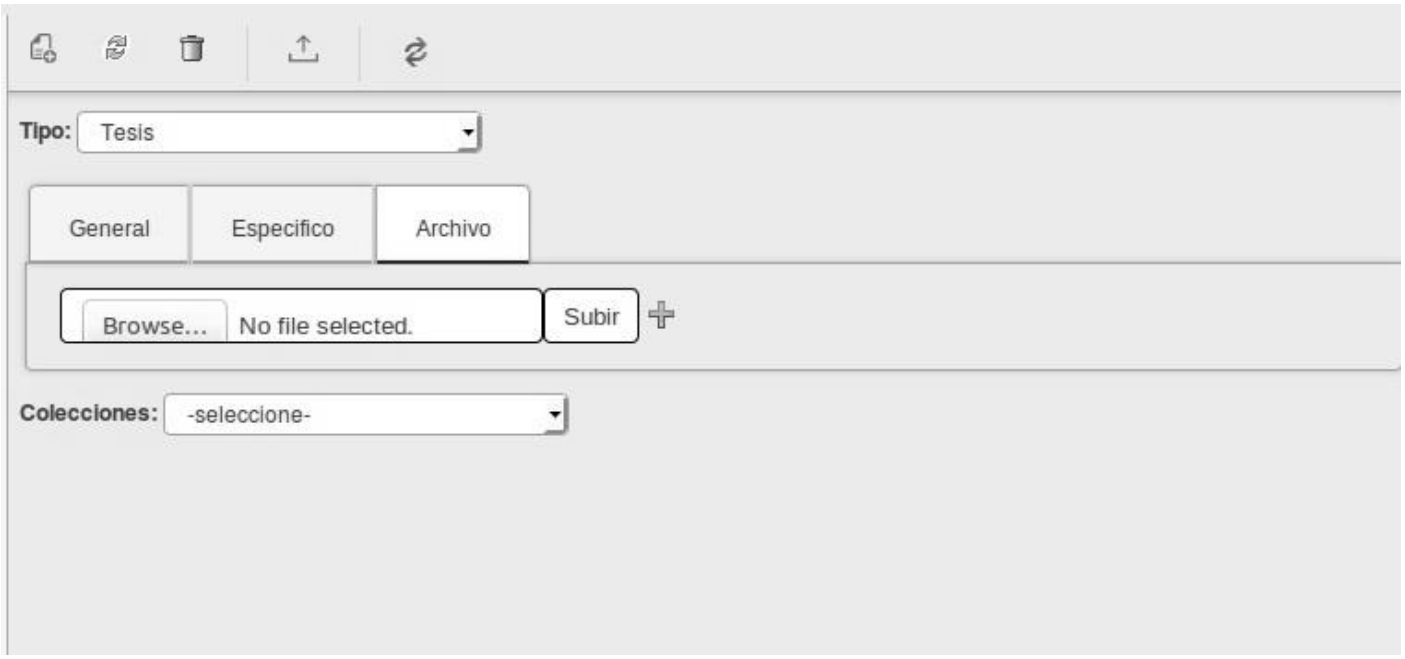
Objetivo	Enviar documento	
Actores	Usuario	
Resumen	El caso de uso se inicia cuando el actor desea depositar un documento. El actor llena todos los campos del formulario y podrá enviar el documento.	
Complejidad	Alta	
Prioridad	Media	
Precondiciones	-El sistema debe estar funcionando correctamente. -Un usuario se ha autenticado en el sistema. -Debe haber llenado el formulario del documento a enviar.	
Postcondiciones	Se ha enviado el documento.	
Relaciones	CU	Gestionar formulario para el envío.

	Incluidos	
	CU Extendidos	

Flujo de eventos		
Flujo básico Enviar documento		
	Actor	**Sistema**
1.	Desea enviar un documento (Ver CU: Gestionar formulario para el envío)	
2.		Permite: • Adjuntar uno o más archivos.
3.	Adjunta él o los archivos	
4.		Permite: • Enviar el documento
5.	Selecciona el botón de enviar.	
6.		Valida los datos.
7.		Termina el caso de uso.
Flujos alternos		
3a. No sube archivos		
	Actor	**Sistema**
1.		Muestra mensaje de advertencia: “No se ha adjuntado ningún archivo”
2.		Regresa al paso 3 del flujo básico.
Flujos alternos		
Nº *b. El actor selecciona el botón de documentos procesados para Retomar envío		
	Actor	**Sistema**
1.		Muestra los formularios guardados.
2.		Regresa al paso 1 del flujo básico.
Flujos alternos		
6a. Campos Vacíos		
	Actor	**Sistema**
1.		Muestra mensaje “No puede dejar campos vacíos”
2.		Regresa al paso 1 del flujo básico.
Flujos alternos		
6b. No selecciona colección		
	Actor	**Sistema**
1.		Muestra mensaje “Debe seleccionar una colección”.
2.		Regresa al paso 1 del flujo básico.

Prototipo de interfaz de usuario

Enviar documento



GLOSARIO DE TÉRMINOS

MIME: es una serie de convenciones o especificaciones dirigidas al intercambio a través de Internet de todo tipo de archivos.

DAO: es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una Base de datos o un archivo.

APIs: es un conjunto de reglas (código) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas: sirviendo de interfaz entre programas diferentes, de la misma manera en que la interfaz de usuario facilita la interacción humano-software.

URI: se utiliza para identificar recursos en Internet. Dicho identificador tiene un formato estándar y su propósito es permitir interacción con recursos disponibles en Internet o en alguna red de cómputo, como por ejemplo páginas, servicios, imágenes, vídeos.

SOAP: es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

JSON: es un formato ligero de intercambio de datos. Básicamente describe los datos con una sintaxis que se usa para identificar y gestionar los datos.

Servlet: es una clase en el lenguaje de programación Java, utilizada para ampliar las capacidades de un servidor.