Universidad de las Ciencias Informáticas

Facultad 3

Gestión semiautomática de metadatos bibliográficos desde corpus de texto en formato PDF

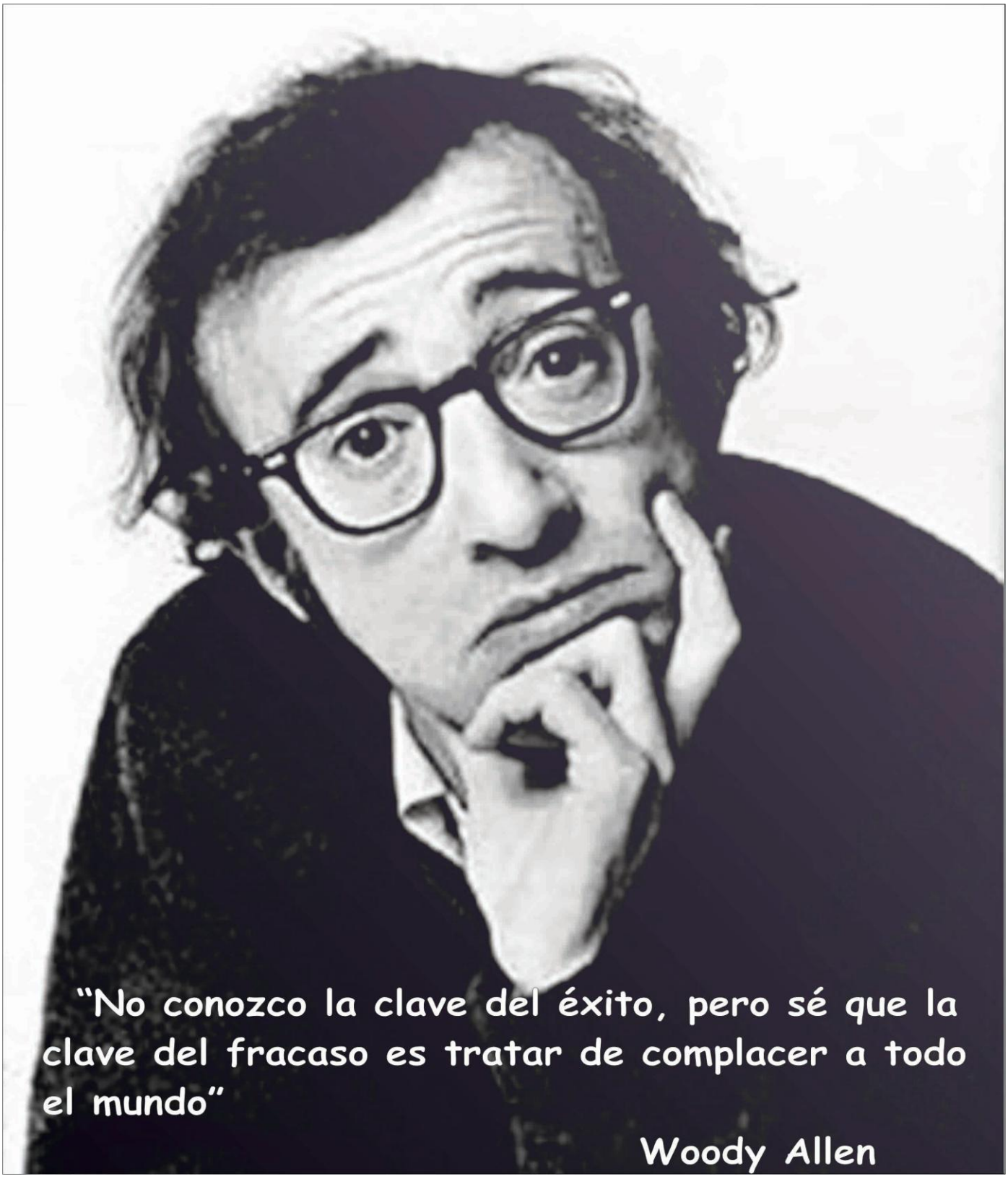
Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores: Rubén Barreto Bárzaga
Osmín Alejandro Velázquez Ariste

Tutores: Ing. Yusniel Hidalgo Delgado
MsC. Yarina Amoroso Fernández

La Habana, Cuba

Junio 2014


"No conozco la clave del éxito, pero sé que la clave del fracaso es tratar de complacer a todo el mundo"
Woody Allen

Declaración de autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos al Centro de Gobierno Electrónico (CEGEL) de 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 de _____ del año _____.

Autores:

_____ _____

Rubén Barreto Bárzaga Osmín Alejandro Velázquez Ariste

Tutores:

_____ _____

Ing. Yusniel Hidalgo Delgado MsC. Yarina Amoroso Fernández

Datos de contacto

Tutora:

Nombre y apellidos: MsC. Yarina Amoroso Fernández

Correo electrónico: yarina@uci.cu

Situación laboral: Profesor

Institución: Universidad de las Ciencias Informáticas (UCI).

Dirección: Carretera a San Antonio de los Baños, Km 2 1/2, Reparto Torrens, Boyeros.

Tutor:

Nombre y apellidos: Ing. Yusniel Hidalgo Delgado

Correo electrónico: yhdelgado@uci.cu

Situación laboral: Profesor

Institución: Universidad de las Ciencias Informáticas (UCI).

Dirección: Carretera a San Antonio de los Baños, Km 2 1/2, Reparto Torrens, Boyeros.

Agradecimientos

Traté de hacer los agradecimientos como los hace todo el mundo o al menos la mayoría de las tesis que he podido leer pero me pareció que el orden de prioridad era demasiado fuerte para mi así que lo hice de la forma más cómoda para mí y mis ojos.

A Osmín por si nadie lo conoce soy yo mismo, gracias por superarte y recuerda que nada es imposible y que el límite es el que uno se ponga.

A los amigos que hice en esta etapa que ya culmina: Carlos, Andrew, Pablo, Reinier, Michel, Joel gracias por ser parte de esto.

A los amigos que de tiempo: Argudin, Gabi créanlo o no en septiembre ya formamos parte CTC (Central de trabajadores de Cuba).

A mi compañero de tesis, gracias por todo aunque tú siempre supiste quien es el jefe.

A mi primo Juanito compañero de andanzas en San Cristóbal gracias por venir.

A mi novia: Aymee gracias por estar siempre ahí para mí y espero que nunca olvides que eres el amor de mi vida.

A mi abuela Eloina que no se encuentra con nosotros pero de seguro estuviera orgullosa de estar aquí presenciando este momento.

A mi hermano Alejandro gracias por todo y recuerda que todavía sigo siendo el mayor.

A mi madre y mi padre ustedes son los culpables del ingeniero en que me he convertido, estoy orgulloso de ser su hijo gracias por todo.

A todo aquel que de una forma u otra sabe que puso un granito de arena para lograr este resultado.

A todos muchas gracias.

Osmín

Agradecimientos

Primero que todo le agradezco a mi mamá por estar siempre presente, por decirme siempre que yo si podía, por demostrarme que de los malos momentos siempre se sale, por ser la mejor madre del mundo, te quiero mucho mami y para ti es este trabajo, gracias por todo.

A mi padrastro por quererme como su hijo, gracias por todos tus consejos, gracias por estar siempre ahí, gracias por ser un padre.

A mi papá no por sus virtudes como padre que son muchas, si no por ser mi modelo de hombre a seguir, mi ejemplo, mi faro.

A mi abuela, que no puede estar aquí hoy conmigo, a ti mi viejita linda muchas gracias por todo.

A mi hermano de toda la vida Guillermito, por estar no en los momentos buenos si no en los momentos que de verdad lo necesité.

A Liset, por ser la persona más comprensiva y paciente que conozco, gracias.

A Maira por ser mi madre sustituta en los momentos difíciles que mi mamá no podía estar.

A mi compañero de tesis, hermano sin ti nada de esto hubiese sido posible, gracias por todo.

A mis amigos: Pablo, Andrew, Yaniel, Artime, Ernesto, Roberto, Orlando. Ustedes son también autores de este trabajo.

A todo aquel que de una forma u otra sabe que puso un granito de arena para lograr este resultado.

A todos muchas gracias.

Rubén

Dedicatoria

A mis padres que han sido los cómplices y los culpables en gran medida de todo lo que he logrado.

A la Revolución y a todas las personas que han hecho posible que muchos jóvenes (incluyéndome) de este país puedan graduarse de una carrera universitaria.

Osmín

A la persona más especial de mi vida por ser mi luz y mi guía en todo momento, a mi mamá.

A Papo pues con su ayuda me he convertido en el hombre que hoy soy.

A los dos le dedico este trabajo porque también es suyo, a ambos gracias por todo.

Rubén

Resumen

El presente trabajo tiene como objetivo implementar una herramienta para la gestión semiautomática de metadatos bibliográficos desde corpus documentales en formato PDF para la Sociedad Cubana de Derecho e Informática. La gestión de metadatos desde este tipo de corpus ha ganado en aplicabilidad en los últimos años, pero es un problema realizar esta tarea pues en esta entidad no existe una herramienta informática para ello. En esta investigación se propone una herramienta para realizar la gestión semiautomática de estos metadatos bibliográficos. Para llegar a la solución propuesta fue necesario realizar un estudio del arte y conocer los principales conceptos vinculados a la gestión semiautomática. La solución propuesta utiliza algunos de estos elementos para poder realizar la gestión de metadatos. Ejemplo de estos elementos son los metadatos bibliográficos definidos por el esquema de metadatos Dublin Core. Los resultados obtenidos demuestran la viabilidad de la herramienta propuesta y son similares a las aproximaciones existentes en la literatura consultada para herramientas de este tipo. Los experimentos realizados demuestran que la exactitud de la solución está por encima de un 70% considerándose estos valores aceptables.

Palabras claves: gestión de metadatos, metadatos bibliográficos, Dublin Core, herramienta.

Tabla de contenido

Introducción 1
Capítulo 1: Fundamentación teórica 5
1.1. Introducción 5
1.2. Análisis bibliométrico y documental 5
1.3. Conceptos fundamentales 6
1.3.1. Datos 6
1.3.2. Metadatos 6
1.3.3. Corpus de texto 6
1.3.4. Registro bibliográfico 6
1.3.5. Extracción de metadatos 7
1.3.6. Procesamiento del lenguaje natural 7
1.4. Principales herramientas existentes para la gestión de metadatos 8
1.4.1. Layout Aware PDF Text 8
1.4.2. Mendeley Desktop 1.10.1 9
1.4.3. Metadata Extraction Tool 9
1.4.4. Foca (Fingerprinting Organizations with Collected Archives) 10
1.4.5. CRODA: Gestión de metadatos para facilitar la recuperación de Objetos de Aprendizaje 10
1.4.6. SyGMe: Sistema para la Gestión de Metadatos Geográficos 10
1.4.7. Herramienta para la recolección de metadatos bibliográficos mediante el protocolo OAI-PMH 11
1.5. Modelo Dublin Core 11
1.6. Plataforma de desarrollo 12
1.6.1. Lenguaje de modelado 12
1.6.2. Herramientas de desarrollo 12
1.6.3. Metodología de desarrollo 13
1.6.4. Programación del lado del cliente 16
1.6.5. Programación del lado del servidor 17
1.6.6. Base de datos 18
1.7. Conclusiones parciales 20

Capítulo 2: Propuesta de solución **21**
2.1. Introducción **21**
2.2. Descripción general de la propuesta **21**
2.2.1. Requisitos de software **21**
2.2.1.1. Requisitos funcionales **23**
2.3. Fase de planificación **25**
2.3.1. Historias de usuario **25**
2.3.2. Estimación de esfuerzos por historias de usuario **27**
2.3.3. Plan de iteraciones **27**
2.3.4. Plan de entrega **28**
2.4. Fase de diseño **29**
2.4.1. Descripción de la arquitectura **29**
2.4.2. Diagrama de clases persistentes **30**
2.4.3. Tarjetas CRC **32**
2.4.4. Diseño de la base de datos **34**
2.4.5. Estándares de codificación **35**
2.4.6. Patrones de diseño **35**
2.5. Fase de desarrollo **37**
2.5.1. Diagrama de despliegue **38**
2.5.2. Tareas de ingeniería por iteraciones **39**
2.5.2.1. Tareas detalladas **39**
2.6. Conclusiones parciales **40**
Capítulo 3: Validación de la solución propuesta **42**
Introducción **42**
3. Validación de requisitos **42**
3.1 Métricas para la validación de la calidad de los requisitos **42**
3.2 Validación del diseño **44**
3.2.1 Tamaño Operacional de Clases (TOC) **44**
3.2.2 Relaciones Entre Clases (RC) **46**
3.3 Fase de pruebas **48**
3.3.1 Pruebas unitarias **49**

3.3.2 Pruebas de aceptación 53
3.3.3 Análisis de los resultados 54
3.4 Validación de la solución 55
3.4.1 Métricas para la evaluación 56
3.4.2 Resultados experimentales 57
3.5 Conclusiones parciales 62
Conclusiones 64
Recomendaciones 65
Referencias 66
Anexos 69
Anexo 1: Historias de usuario 69
Anexo 2: Tarjetas CRC 70
Anexo 3: Tareas de ingeniería por iteraciones 71
Anexo 4: Tareas detalladas 72
Anexo 5: Casos de prueba de aceptación 75
Anexo 6. Aval de aceptación 79

Introducción

En los últimos años y ante el avance progresivo de la sociedad de la información se ha experimentado una serie de cambios y transformaciones debido a la introducción de las nuevas tecnologías en la sociedad. Las nuevas tecnologías han promovido en millones de personas, hogares y oficinas, una comunicación electrónica mediante estándares universales y abiertos. Esta nueva tendencia ha posibilitado que empresas e instituciones generadoras de información como son los archivos, las bibliotecas y los centros de documentación, cambien sus estrategias y se adapten a las nuevas tecnologías.

La archivística moderna se concibe como una ciencia que estudia la naturaleza de los archivos, su organización, los principios para su conservación y los medios para prestar un servicio. La labor de la archivística tradicionalmente vinculada a la investigación histórica está perdiendo protagonismo frente al documento electrónico así como el resto de las disciplinas vinculadas a las ciencias de la información. Los cambios tanto en sus procedimientos como en las técnicas empleadas para gestionar la información han hecho de la gestión de metadatos una técnica a tener en cuenta para el trabajo con información de cualquier tipo.

Con el auge de las Tecnologías de Información y Comunicaciones (TIC), especialmente la Internet, la archivística adoptó nuevas formas de gestionar la información. La aparición de los documentos en formato PDF (siglas del inglés Portable Document Format, formato de documento portátil) aparece como una alternativa para la creación de documentos. La extracción y almacenamiento de metadatos desde este tipo de documentos ha ganado en aplicabilidad en los últimos años, sobre todo en dominios del conocimiento tales como la bibliotecología, la archivística y los museos.

El PDF es un formato de almacenamiento de documentos digitales independiente de las plataformas de software o hardware. Los PDF pueden contener cualquier combinación de texto, elementos multimedia como vídeos o sonido, elementos de hipertexto como vínculos y marcadores, enlaces y miniaturas de páginas, lo que lo hace el formato más difundido en el internet para el intercambio de documentos.

En los últimos años Cuba ha potenciado el desarrollo, consolidación y expansión de la industria del software por las grandes ventajas económicas y por la necesidad de tener un desarrollo propio en esta esfera. En este contexto surge en el año 2002 la Universidad de las Ciencias Informáticas (UCI), que tiene como objetivos principales la formación integral de profesionales y la producción de software de alta calidad.

En el Centro de Gobierno Electrónico (CEGEL) de la UCI se encuentran en ejecución varios proyectos productivos que satisfacen necesidades de diferentes organismos vinculados al gobierno, tanto dentro como

fuera del país, tal es el caso del proyecto Observatorio de Gobierno Electrónico. Este proyecto maneja grandes volúmenes de información, la cual años atrás hubiese sido almacenada en formato duro, pero con el uso actual de la tecnología se tiene una gran cantidad de documentos digitales formando varios corpus[1] en formato PDF. Estos documentos siguen una estructura definida con anterioridad. Sin embargo, la utilización de estos documentos y sus metadatos se ve limitada en terceros sistemas informáticos.

La definición de técnicas y métodos para la extracción automática o semiautomática de metadatos desde grandes volúmenes de documentos PDF constituye un área activa de investigación. Este tipo de soluciones proporcionan una solución viable para transformar un documento PDF o grupo de estos en metadatos que luego pueden ser empleados por otros sistemas informáticos con fines específicos, tales como búsqueda y recuperación de información, clasificación de documentos, extracción de patrones y minería de texto.

Teniendo en cuenta lo anteriormente planteado surge como **problema a resolver:** ¿Cómo gestionar los metadatos bibliográficos a partir de corpus de documentos en formato PDF de manera que se potencie su utilización por sistemas informáticos?

Para ello se centró la investigación en el **objeto de estudio:** la gestión de metadatos bibliográficos enmarcado en el **campo de acción:** gestión de metadatos bibliográficos a partir de documentos en formato PDF.

Para dar solución al problema planteado se define como **objetivo general:** desarrollar una herramienta informática para la gestión de metadatos bibliográficos a partir de corpus de documentos en formato PDF utilizando un enfoque semiautomático potenciando la utilización de este formato en sistemas informáticos.

Por lo que se define la siguiente **idea a defender:** si se desarrolla una herramienta informática que sea capaz de extraer metadatos bibliográficos de documentos en formato PDF se contribuirá a la utilización de los mismos por otros sistemas informáticos.

Con el fin de dar cumplimiento al objetivo general de la investigación se concibieron los siguientes **objetivos específicos:**

- Obtener el marco teórico y el estado del arte del objeto de estudio de la investigación mediante el análisis bibliográfico documental para identificar tendencias y adoptar posiciones al respecto.
- Elaborar los principales artefactos de ingeniería correspondientes a las fases de análisis y diseño de la solución informática.

[1] Corpus: Conjunto extenso de textos de diversas clases, ordenados y clasificados que sirven como base a una investigación.

- Codificar la solución informática en un lenguaje de programación utilizando el paradigma de la Programación Orientada a Objetos.
- Realizar pruebas de calidad de software a la herramienta implementada para comprobar su correcto funcionamiento.

Para la realización de la investigación se emplearon los siguientes **métodos científicos:**

Métodos teóricos:

- Analítico-Sintético: se utilizó para el análisis de documentos, materiales y temas relacionados con el dominio de la investigación. Este análisis permitió realizar una síntesis de los elementos más importantes relacionados con estos temas, lo que facilitó la selección de las herramientas y tecnologías necesarias para el desarrollo de la propuesta de solución.
- Histórico-Lógico: se utilizó para desarrollar un estudio del estado del arte de la problemática y cómo han venido comportándose históricamente los sistemas de gestión de metadatos, sus condiciones de uso y aplicación, características, fortalezas, entre otros aspectos.

Métodos empíricos:

- Experimentación: se utiliza para la realización de los experimentos diseñados con el objetivo de validar la propuesta de solución.
- Medición: se utiliza en el cálculo de la efectividad de la propuesta de solución y de la calidad de las respuestas finales.

El presente Trabajo de Diploma posee la siguiente estructura:

En el primer capítulo se realizara un análisis bibliométrico y documental de la bibliografía consultada. Se analizan los principales conceptos y definiciones asociados al dominio del problema que sirven de apoyo durante el desarrollo de la investigación. Como aspecto medular se ofrece un enfoque de los términos fundamentales relacionados con la gestión semiautomática de metadatos. Además se hace un análisis del uso de la gestión de metadatos en el resto del mundo.

En el segundo capítulo se describe la propuesta de solución. Se muestran los artefactos generados durante la fase de planificación, así como los requisitos funcionales y no funcionales con que debe contar la solución. Además, se muestran otros artefactos propuestos para la fase de diseño. Se elaboran las tarjetas CRC para guiar la construcción de la solución propuesta y se muestran otros artefactos de ingeniería de la etapa de implementación.

Finalmente, en el tercer capítulo se realiza la validación de la solución propuesta mediante la aplicación de métricas y pruebas definidas. Además se presenta la valoración de los resultados obtenidos por cada una de las pruebas realizadas demostrando el correcto funcionamiento de la aplicación y la aceptación de la misma por parte del cliente.

Capítulo 1: Fundamentación teórica

1.1. Introducción

En este capítulo se enuncian los conceptos fundamentales sobre la gestión semiautomática de metadatos los cuales permitirán un mejor entendimiento de este trabajo. Por último se muestra un análisis del estado del arte de las principales herramientas para la gestión de metadatos.

1.2. Análisis bibliométrico y documental

Para la realización de la presente investigación se llevó a cabo un estudio documental enmarcado principalmente en la literatura de los últimos 5 años. Se consultaron varias fuentes bibliográficas, entre las que se encuentran bases de datos referenciadas como SCOPUS[2], Springer[3], IEEE[4], además se visitaron varios sitios web oficiales con el objetivo de resolver el problema de la gestión de metadatos. A continuación se muestra una tabla resumen de la bibliografía consultada.

	Últimos 5 años	Años anteriores
Libros y monografías	13	8
Tesis de doctorado	5	3
Tesis de maestría	8	4
Artículos científicos	7	3
Páginas Web	8	5

Tabla 1. Análisis bibliométrico y documental

La tabla anterior muestra que se consultaron un total de 64 bibliografías distintas, de las cuales 41 de ellas fueron publicadas en los últimos 5 años lo cual representa un 64 % aproximadamente del total de la bibliografía consultada.

[2] http://www.scopus.com/home.url
[3] http://www.springer.com
[4] http://ieeexplore.ieee.org/Xplore/home.jsp

1.3. Conceptos fundamentales

En este epígrafe se definen los conceptos fundamentales asociados al dominio del problema en cuestión con el objetivo de avalar la investigación, posibilitando de esta forma crear una base de conocimiento necesaria para el desarrollo de la misma.

1.3.1. Datos

Según la Real Academia de la Lengua Española, los datos no son más que un antecedente necesario para llegar al conocimiento exacto de algo o para deducir las consecuencias legítimas de un hecho. Pueden ser documentos, testimonios y fundamentos o información dispuesta de manera adecuada para su tratamiento en un ordenador (1).

1.3.2. Metadatos

Los metadatos son datos altamente estructurados que describen información, el contenido, la calidad, la condición y otras características de los datos. “Son datos acerca de datos y denotan cualquier tipo de conocimiento que puede usarse para conseguir información sobre la estructura y el contenido de una colección de documentos” (2). Los metadatos organizan la estructura de un documento, es información de los datos, como su estructura y formato. Son datos estructurados que describen y permiten encontrar, administrar, comprender o preservar documentos archivísticos a lo largo del tiempo (3).

1.3.3. Corpus de texto

Los corpus de texto son usados en el ámbito de la lingüística o lexicografía o en la lingüística computacional en general. Se puede definir corpus como cualquier colección que contenga más de un texto (corpus como cuerpo textual). Un corpus no es más que una colección de texto en formato magnético. Un corpus debe estar compuesto por textos producidos en situaciones reales y la inclusión de los textos que componen el corpus debe estar guiada por una serie de criterios lingüísticos explícitos para asegurar que pueda usarse como muestra representativa de una lengua (4).

1.3.4. Registro bibliográfico

El registro bibliográfico es la representación de un documento mediante una serie de datos descriptivos establecidos por normas internacionales. Estos datos permiten identificar el documento y establecer puntos de acceso para su posterior recuperación de entre un conjunto de registros del mismo tipo. De esta manera, el registro bibliográfico va a interactuar entre el usuario y el documento, informando sobre aspectos como el autor, el título, el editor o el contenido (5).

En cuanto al formato, destacamos los más utilizados a escala internacional (5):

- ISBD, para la generación de catálogos manuales o en papel y en formato legible por máquina
- MARC 21, para la generación de catálogos automatizados

1.3.5. Extracción de metadatos

La extracción automática de metadatos consiste en obtener un conjunto de atributos o elementos necesarios que describan documentos digitales. Los metadatos extraídos se utilizan para la descripción e identificación de los materiales digitales. Una vez que este proceso se realice, estos elementos pueden ser depositados en una base de datos, repositorio o en un lugar donde se almacena información digital con el objetivo de preservarla.

Los metadatos pueden ser clasificados en tres tipos:

Metadatos descriptivos: se utilizan para la descripción e identificación de la información contenida en el recurso. Contienen atributos físicos (medios, condición de las dimensiones) y atributos bibliográficos (título, autor/ creador, idioma, palabras claves) (6).

Metadatos administrativos: se refieren a las características y propiedades del recurso, facilitando la gestión y procesamiento tecnológico y físico de las colecciones digitales tanto a corto como a largo plazo. Incluyen información sobre la creación y el control de calidad, gestión de derechos, control de acceso y utilización y condiciones de preservación (6).

Metadatos estructurales: proporcionan información sobre la estructura interna de los recursos electrónicos, como página, sección, capítulo, partes, índices, tabla de contenido, etc. y describen la relación entre los materiales. Facilitan la navegación y presentación de los recursos y relacionan las diferentes partes que lo componen (6).

De los tres tipos de metadatos existentes, en esta investigación se utilizarán los metadatos descriptivos, específicamente sus atributos bibliográficos, ya que estos serán los atributos que se gestionarán principalmente de los corpus de textos.

1.3.6. Procesamiento del lenguaje natural

El procesamiento de lenguaje natural (PLN) es el campo que combina la ciencia computacional (como la inteligencia artificial, el aprendizaje automático o la inferencia estadística) con la lingüística aplicada, con el objetivo de hacer posible la comprensión y el procesamiento asistidos por ordenador de información expresada en lenguaje natural para determinadas tareas, como la traducción automática de textos, los sistemas de diálogo interactivos, el análisis de opiniones, entre otras (7).

1.4. Principales herramientas existentes para la gestión de metadatos

A continuación se describen algunas de las aproximaciones encontradas en la literatura y que guardan relación con el objeto de estudio de la investigación.

1.4.1. Layout Aware PDF Text

El sistema *Layout Aware PDF Text* se centra solamente en el contenido textual de los artículos de investigación y que se entiende como una línea de base para futuros experimentos en métodos de extracción más avanzados que manejan contenidos multimodal, por ejemplo, imágenes y gráficos. El sistema trabaja en un proceso de tres etapas: (8) la detección de bloques de texto contiguos utilizando procesamiento de distribución espacial para localizar e identificar los bloques de texto contiguos, (9) la clasificación de los bloques de texto en categorías retóricas mediante un método basado en reglas y la clasificación de bloques de texto en el orden correcto para la extracción de texto a partir de bloques agrupados.

La herramienta cuenta con seis funcionalidades principales:

- blockify: obtiene del documento PDF un XML para cada bloque y sin clasificación
- blockifyClassify: obtiene del documento PDF un XML para cada bloque y la clasificación basada en un archivo de reglas
- blockStatistics: obtiene del documento PDF estadísticas para cada bloque en base a las características espaciales:
- Cantidad de bloques por página
- Fuentes que predominan en el bloque
- Tamaño de letra que predomina en el bloque
- Palabra que más predomina en el bloque
- Número de líneas del bloque
- extracFullText: obtiene del documento PDF un XML que tiene una lista de títulos de las secciones (designado en el archivo de reglas)
- imagifyBlocks: obtiene del documento en PDF una imagen en formato PNG por cada página del mismo, donde señala los bloques por páginas con el tipo y el tamaño de letra de estos
- imagifySections: btiene del documento en PDF una imagen en formato PNG por cada página del mismo, donde señala los bloques, palabras y signos de puntuación

Cada una de estas funciones extrae los datos de documentos en formato PDF de forma distinta, solo blockify, blockifyClassify, blockStatistics, extracFullText lo hacen exportándolos en un archivo en formato XML[5]. La función blockify es la única que extrae todo el contenido del documento en formato PDF, esta función captura los datos mediantes bloques nombrados en el XML como *Chunk*[6]. Un bloque puede ser una palabra, una oración, un párrafo (siempre que tengan el mismo tipo y tamaño de letra) o un signo de puntuación. La forma en que se captura la información mediante blockify hace que se pueda diseñar un conjunto de reglas para dar solución al problema de la extracción, por lo que hace sea esta la función elegida para el desarrollo de la solución.

1.4.2. Mendeley Desktop 1.10.1

Mendeley Desktop es una herramienta para el trabajo con artículos científicos en formato PDF. Esta herramienta facilita la organización de estos artículos importándolos en carpetas tras leer sus metadatos, los artículos se pueden marcar como favoritos o leídos con un clic. Posee también una columna que muestra el tiempo transcurrido desde que se agregaron a la biblioteca y el panel izquierdo de Mendeley Desktop ayuda a clasificar los artículos gracias a filtros rápidos y clasificadores similares a listas de reproducción.

El diseño de la interfaz es limpio y mucho más agradable de usar en el día a día que el de clásicos como *EndNote* o *RefMan*. Las citas se pueden exportar a multitud de formatos mediante operaciones simples.

Mendeley Desktop se conecta con su propia red social (se debe crear una cuenta primero). Las carpetas se pueden sincronizar y compartir con otros investigadores. Las cuentas gratuitas disponen de 500 megabytes para almacenar la bibliografía. Mendeley Desktop soporta formatos tales como PDF, BIB, XML, RIS, *Endnote* (10).

1.4.3. Metadata Extraction Tool

Es una herramienta desarrollada por la Biblioteca Nacional de Nueva Zelanda inicialmente en 2003 y lanzado como software de código abierto en 2007. Los objetivos del software consisten en (11):

- Extraer automáticamente metadatos para la preservación de archivos digitales
- Dar salida a los metadatos en un formato estándar XML para su uso en actividades de conservación y de recuperación de información

La herramienta extrae metadatos desde los siguientes formatos:

- Imágenes: BMP, GIF, JPEG y TIFF

[5] XML: Lenguaje de Marcas Extensible, por sus siglas en inglés.
[6] Chunk: es un fragmento de información contenido en algunos formatos de archivo.

- Documentos de Office: Microsoft Word (versión 2, 6), Word Perfect, Open Office (version 1), MS Works, MS Excel, MS PowerPoint y PDF
- Audio y video: WAV, MP3 (normal y con ID3Tags), BFW, FLAC
- Lenguajes de marcado: HTML y XML

La herramienta se distribuye como software libre bajo la Licencia Pública Apache (versión 2.0) (11).

1.4.4. Foca (Fingerprinting Organizations with Collected Archives)

Es una herramienta para la recolección de archivos publicados en los sitios web; extrae los metadatos y los analiza. Permite trabajar con una base de datos SQL server. Ayuda en las labores de *Footprinting*[7] y *Fingerprinting*[8] a los auditores de seguridad, además elimina y modifica los metadatos (12).

La herramienta extrae metadatos e información oculta en los siguientes formatos:

- Microsoft Office de la versión 97 a la 2007
- *Open Office*
- *Portable Document Format* (pdf)
- *Corel Word Perfect document* (wpd)
- *Joint Photographic Experts Group* (jpg)

1.4.5. CRODA: Gestión de metadatos para facilitar la recuperación de Objetos de Aprendizaje

La herramienta CRODA a través del <<*Learning Object Metadata*>>, que es un módulo para la gestión de metadatos en los objetos de aprendizaje. Su utilización permite especificar sesenta y cuatro elementos de información, lo cual aumenta la posibilidad de búsqueda y localización. Está caracterizada por brindar a los profesores funcionalidades que facilitan el etiquetado, contribuyendo de esta manera a su motivación para realizar esta actividad. «*Learning Object Metadata*» se encuentra entre las principales soluciones para la catalogación de objetos de aprendizaje y al facilitar la descripción de este recurso a los profesores, incide positivamente en su recuperación y en todo el proceso de gestión de información (13).

1.4.6. SyGMe: Sistema para la Gestión de Metadatos Geográficos

El SyGMe: Sistema para la Gestión de Metadatos Geográficos es uno de los sistemas más importantes en los procesos de captura de metadatos geográficos, principalmente en el país. Permite interpretar y procesar estos datos geográficamente referenciados a través de una aplicación. El sistema implementa varias normas para la creación de metadatos geográficos como es el núcleo de la norma ISO 19115 y su versión extendida,

[7] Obtención de la huella
[8] Obtención de la huella genética

con la que el usuario podrá gestionar fácilmente los metadatos geográficos. Cuenta además con una agenda de contactos que registra los datos de los autores de los metadatos y otro módulo encargado de la gestión de tesauros y las funcionalidades para la publicación, consulta y búsqueda (14).

1.4.7. Herramienta para la recolección de metadatos bibliográficos mediante el protocolo OAI-PMH

Con el desarrollo de las Tecnologías de la Información y las Comunicaciones se ha incrementado el número de revistas científicas en formato digital, lo que acelera la difusión y el acceso a sus contenidos. Un gran número de revistas científicas se han unido a la Iniciativa de Acceso Abierto, dando acceso al texto completo de sus artículos científicos. Además, algunas de estas revistas ofrecen sus metadatos bibliográficos a través del protocolo OAI-PMH. Este protocolo utiliza el protocolo HTTP para obtener metadatos del proveedor de datos (o depósito), tales como colecciones y registros bibliográficos. Con la integración de metadatos bibliográficos se proporcionará a la comunidad científica servicios tales como: búsqueda de expertos, detección de nuevos temas de investigación, análisis de redes científicas, entre otros. Esta herramienta es utilizada para la recolección de metadatos bibliográficos utilizando el protocolo OAI-PMH (15).

Valoración de las herramientas estudiadas:

Las herramientas estudiadas resuelven un problema en específico, son herramientas de gran importancia en cada uno de sus escenarios, pero ninguna de ellas puede ser utilizada en su totalidad para resolver el problema que se plantea en esta investigación. No obstante de la herramienta Layout Aware PDF Text se utilizará una de las funciones que esta nos provee, específicamente la función de blockify. Esta función será utilizada para la extracción de texto de documentos en formato PDF los cuales se guardaran en un archivo de formato XML al cual se le realizará el proceso de captura de metadatos. Las demás herramientas no se de utilizaran puesto que no se ajustan a las necesidades de la investigación, aunque permiten un mejor entendimiento de cómo funciona el proceso de extracción de metadatos de los distintos tipos de archivos, principalmente de archivos en formato PDF que son en los que está enmarcada la investigación.

1.5. Modelo Dublin Core

Dublin Core es un modelo de metadatos elaborado por la DCMI (Dublin Core Metadata Initiative), una organización dedicada a fomentar la adopción extensa de los estándares de interoperables de los metadatos y a promover el desarrollo de los vocabularios especializados de metadatos para describir recursos (16).

Las implementaciones de Dublin Core usan generalmente XML, se define por la norma ISO 15836 del año 2003, y la norma NISO Z39.85-2007 (16).

El conjunto de elementos Dublin Core se centra en 15 descriptores como resultado de un consenso y un esfuerzo interdisciplinar e internacional. El conjunto de elementos se divide en tres apartados: contenido, propiedad intelectual y aplicación (16).

Los campos de datos que se utilizarán de Dublin Core para el presente trabajo de investigación son: autor, tutor, titulo, resumen y palabras claves estos son los que serán generados por la herramienta implementada.

1.6. Plataforma de desarrollo

1.6.1. Lenguaje de modelado

El Lenguaje Unificado de Modelado (UML)[9] es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir (17) (18).

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Este lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML es una notación con la cual se construyen sistemas por medio de conceptos orientados a objetos. Este prescribe un conjunto de notaciones y diagramas estándares, y describe la semántica esencial de lo que estos diagramas y símbolos significan (19).

1.6.2. Herramientas de desarrollo

Visual Paradigm 5.0

Durante la etapa de modelación y para diseñar el sistema se utilizó Visual Paradigm, una herramienta de modelado multiplataforma que no se inclina por ninguna metodología específica. Además ofrece un entorno de creación de diagramas para UML, con soporte para los 13 diagramas de la última versión (UML 2.0). Por otro lado el diseño es centrado en casos de uso y enfocado al negocio, con soporte para los Diagramas de Procesos de Negocios (BPD)[10] y Diagramas de Flujos de Datos (DFD)[11], lo cual genera un software de mayor calidad (20).

[9] Unified Modeling Language
[10] Business Process Diagram
[11] Data Flow Diagram

Usa un lenguaje estándar común para todo el equipo de desarrollo que facilita la comunicación y está capacitado para la ingeniería directa e inversa en Java, C++ y PHP, además de la capacidad de generación de código en estos lenguajes (21).

Presenta dos tipos de diagramas de modelado de bases de datos: entidad-relación (ERD)[12] y mapeo objeto-relacional (ORM)[13]. Los diagramas ERD modelan la base de datos a nivel físico y los ORM muestran la relación entre las clases y las entidades (20).

NetBeans 7.3

NetBeans es un entorno de desarrollo integrado (IDE)[14] de código abierto para desarrolladores. Posee todas las herramientas necesarias para crear aplicaciones web, de escritorio y para teléfonos móviles con los lenguajes de programación Java, C, C++ e incluso lenguajes dinámicos como: PHP, Javascript, Groovy, y Ruby. Es fácil de usar e instalar. Puede ser ejecutado en múltiples plataformas como Windows, Linux, Mac OS X y Solaris (22).

NetBeans IDE 7.3 soporta todas las características estándar, como el autocompletado de código, el resaltado de sintaxis, refactorización, plantillas de código, documentación de pop-up, navegación de código, las advertencias de editor y lista de tareas (22).

Entre sus principales características podemos mencionar:

- Brinda completamiento automático de código PHP, así como coloreado de código sintáctico y semántico
- Permite depurar el código usando Xdebug
- Genera fragmentos de código para bases de datos MySQL

Dadas las características, ventajas y soporte que brinda este potente entorno de desarrollo y sobre todo por tener Licencia Pública General (GPL)[15] versión 2, se eligió como IDE a utilizar para el desarrollo del sistema a implementar.

1.6.3. Metodología de desarrollo

Para desarrollar un producto de software es necesario aplicar una metodología que guíe el proceso de desarrollo. Según Mario Piattini una metodología de desarrollo de software es “un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software” (23). Una metodología es la que define Quién debe hacer Qué, Cuándo y Cómo (23).

[12] Entity-Relationship Diagram
[13] Object-Relational Mapping
[14] Integrated Development Environment
[15] General Public License

Las necesidades principales que persiguen las metodologías son (23):

- Mejores aplicaciones, tendientes a una mejor calidad, aunque a veces no es suficiente
- Un proceso de desarrollo controlado, que asegure uso de recursos y costo apropiados
- Un proceso estándar en la organización, que no sienta los cambios del personal

Como objetivos principales de las metodologías se tiene:

- Brindar un método sistemático, de modo que se controle el progreso del desarrollo
- Especificar los requerimientos de un software en forma apropiada
- Construir productos bien documentados y de fácil mantenimiento
- Ayudar a identificar las necesidades de cambio lo más pronto posible
- Proporcionar un sistema ágil que satisfaga a todas las personas involucradas

Seleccionar una metodología apropiada, es un factor esencial para obtener un software con la calidad esperada. No existe una metodología que puede ser aplicada en todos los proyectos y su selección depende de las características de cada uno de ellos.

Metodología ágil: Programación Extrema

Programación Extrema (por sus siglas en inglés, XP) es una metodología de desarrollo de software creada por Kent Beck. Esta metodología se basa en la comunicación, la claridad y la reutilización continua de código. Tiene como objetivos fundamentales la satisfacción del cliente y promover el trabajo en equipo.

Características de XP (24):

- XP es una metodología “liviana” que no tiene en cuenta la utilización de elaborados casos de uso, la exhaustiva definición de requerimientos y la generación de una extensa documentación.
- XP tiene asociado un ciclo de vida y es considerado a su vez un proceso.
- La tendencia de entregar software en espacios de tiempo cada vez más pequeños con exigencias de costos reducidos y altos estándares de calidad.
- XP define Historias de Usuario como base del software a desarrollar, estas historias las escribe el cliente y describen escenarios sobre el funcionamiento del programa, a partir de las historias de usuarios y de la arquitectura perseguida se crea un plan de liberaciones entre el equipo de desarrollo y el cliente.

XP consta de cuatro fases (24):

- Planificación
- Diseño
- Desarrollo

- Pruebas

Los roles propuestos por Kent Beck (25) son:

- Programador: el programador escribe las pruebas unitarias y produce el código del sistema.
- Cliente: escribe las historias de usuario y las pruebas funcionales para validar su implementación. Asigna la prioridad a las historias de usuario y determina cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.
- Encargado de pruebas (Tester): ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.
- Encargado de seguimiento (Tracker): proporciona realimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.
- Entrenador (Coach). es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.
- Consultor: es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto en el que puedan surgir problemas.
- Gestor (Big boss): es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

El ciclo de desarrollo consiste en los siguientes pasos (26):

1. El cliente define el valor de negocio a implementar
2. El programador estima el esfuerzo necesario para su implementación
3. El cliente selecciona que construir, de acuerdo con sus prioridades y las restricciones de tiempo
4. El programador construye ese valor de negocio
5. Vuelve al paso 1

Para el desarrollo de la solución propuesta se decidió utilizar la metodología XP. Esta metodología fue escogida porque es la definida por el Grupo de Especialistas del proyecto Observatorio de Gobierno Electrónico, el cual rige la investigación en curso. Además está diseñada para equipos de trabajo pequeños, centrada en vincular al cliente en el ciclo de desarrollo, incrementando la posibilidad de éxito, minimizando los riesgos de no conformidades y de obtener un producto final rechazado por el cliente por no cumplir con los objetivos y especificaciones trazadas. La metodología XP permite responder de manera eficiente a los

cambios que se puedan presentar durante todo el desarrollo de la herramienta, proponiendo un ciclo de vida dinámico. El proceso de prueba de XP posibilita probar cada funcionalidad al finalizar cada iteración comprobando si cumple con los requisitos de la herramienta.

1.6.4. Programación del lado del cliente

Chromium Browser 24.0

Chromium es el proyecto de software libre con el que se ha desarrollado Google Chrome y es de participación comunitaria para fundamentar las bases del diseño y desarrollo del navegador Chrome, además del sistema operativo Google Chrome OS (27).

La porción realizada por Google está amparada por la licencia de uso BSD, con otras partes sujetas a una variedad de licencias de código abierto permisivas que incluyen MIT License[16], Ms-PL[17] y la triple licencia MPL[18]/GPL/LGPL[19] (28).

En esencia, los aportes hechos por el proyecto Chromium fundamentan el código fuente del navegador base sobre el que está construido Chrome y por tanto tendrá sus mismas características (29), pero con un logotipo ligeramente diferente y sin el apoyo comercial o técnico de la compañía Google (30).

jQuery 1.8

jQuery es una biblioteca de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privativos. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio (31).

Características (31):

- Selección de elementos DOM
- Interactividad y modificaciones del árbol DOM, incluyendo soporte para CSS 1-3 y un plugin básico de XPath
- Eventos

[16] Massachusetts Institute of Technology License
[17] Microsoft Public License
[18] Mozilla Public License
[19] Lesser General Public License

- Manipulación de la hoja de estilos CSS
- Efectos y animaciones
- Animaciones personalizadas
- AJAX
- Soporta extensiones
- Utilidades varias como obtener información del navegador, operar con objetos y vectores, funciones para rutinas comunes, etc
- Compatible con los navegadores Mozilla Firefox 2.0+, Internet Explorer 6+, Safari 3+, Opera 10.6+ y Google Chrome 8+

Bootstrap 2.0

Bootstrap 2.0 es un framework diseñando para simplificar el proceso de creación de diseños web. Para ello ofrece una serie de plantillas CSS y de ficheros JavaScript, los cuales permiten conseguir (32):

- Interfaces que funcionen de manera brillante en los navegadores actuales y correctamente en los no tan actuales
- Un diseño que pueda ser visualizado de forma correcta en distintos dispositivos y a distintas escalas y resoluciones
- Una mejor integración con las librerías que se suelen usar habitualmente, como por ejemplo jQuery
- Un diseño sólido basado en herramientas actuales y potentes como LESS o estándares como CSS3/HTML5

1.6.5. Programación del lado del servidor

Apache 2.0

El servidor Apache es un servidor web HTTP[20] de código abierto para plataformas Unix, Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual (33) (34). Es el servidor web por excelencia. Su usabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa (34).

[20] HTTP: Hypertext Transfer Protocol

Es un software que está estructurado en módulos, así los programadores asumen que el software puede ser ampliado por otros desarrolladores, los cuales pueden escribir pequeñas partes del código que se integrará en Apache de una manera fácil. Esto se lleva a cabo al haber creado un API modular y una serie de fases bien definidas por las que cada petición al servidor debe atravesar. Estas fases van desde la inicialización del servidor (cuando Apache lee los ficheros de configuración), hasta la traducción de una URL en un nombre de fichero del servidor, o registrar los resultados de la transacción (35).

PHP

PHP[21] es un lenguaje interpretado del lado del servidor, de propósito general, ampliamente usado y que está diseñado especialmente para desarrollo web y puede ser embebido dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida (36).

Es un lenguaje multiplataforma con capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, tiene capacidad de expandir su potencial utilizando la enorme cantidad de módulos, llamados extensiones, es libre, por lo que se presenta como una alternativa de fácil acceso para todos, además de que permite la utilización de las técnicas de Programación Orientada a Objetos.

Symfony 2.0.10

Symfony es un marco de trabajo para el desarrollo de aplicaciones web escrito en PHP. Fue concebido originalmente por la agencia interactiva Sensio Labs para el desarrollo de sitios web para sus propios clientes. Symfony fue publicado por la agencia en 2005 bajo la licencia de código abierto MIT y hoy es uno de los mejores marcos de trabajo disponibles para el desarrollo en PHP (37).

Con el apoyo de Sensio Labs y una gran comunidad de usuarios Symfony tiene muchos recursos: abundante documentación, soporte comunitario mediante listas de correos, soporte profesional mediante consultantes y mucho más (37).

1.6.6. Base de datos

Doctrine 2.0

Doctrine es un ORM escrito en PHP que proporciona una capa de persistencia para objetos de este lenguaje. Es una capa de abstracción que se sitúa justo encima de un sistema gestor de bases de datos (38).

[21] PHP Hypertext Preprocessor

Una característica de Doctrine es el bajo nivel de configuración que necesita para empezar un proyecto. Doctrine puede generar clases a partir de una base de datos existente y después el programador puede especificar relaciones y añadir funcionalidad extra a las clases autogeneradas. No es necesario generar o mantener complejos esquemas XML de base de datos como en otros marcos de trabajo. Otra característica importante de Doctrine es la posibilidad de escribir consultas de base de datos utilizando un dialecto de SQL[22] denominado DQL[23] que está inspirado en HQL[24] y en JPQL[25] (39).

PostgresSQL 9.2

PostgreSQL es un sistema de bases de dato relacional potente y de código abierto. Tiene más de 15 años de desarrollo activo y una arquitectura probada que ha ganado una sólida reputación de fiabilidad, integridad de datos y exactitud. Se ejecuta en los sistemas operativos más importantes, incluyendo Linux, Unix, y Windows. Tiene soporte completo para llaves foráneas, uniones, vistas, disparadores, y procedimientos almacenados (en múltiples lenguajes).

PostgreSQL incluye la mayor parte de los tipos de datos de SQL 2008, incluyendo INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL y TIMESTAMP. También soporta el almacenamiento de objetos binario grandes, incluyendo imágenes, sonidos y videos. Tiene interfaces nativas de programación para C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, entre otros y una excepcional documentación (40).

PostgreSQL, como base de datos de clase empresarial, cuenta con características avanzadas como Control de Concurrencia Multiversión, espacios de tablas, replicación asincrónica, transacciones anidadas, salvas en línea, un planificador de consultas sofisticadas y se provee de tolerancia a fallos. Es altamente escalable tanto en la enorme cantidad de datos que puede manejar y en el número de usuarios simultáneos que puede acomodar. Hay sistemas activos de PostgreSQL en entornos de producción que manejan más de 4 terabytes de datos (40).

[22] Sentence Query Language.
[23] Doctrine Query Language.
[24] Hibernate Query Language.
[25] Java Persistence Query Language.

1.7. Conclusiones parciales

- Se realizó la fundamentación teórica de la investigación por medio de los principales conceptos respecto a la gestión semiautomática de metadatos bibliográficos desde corpus de texto en formato PDF.
- De los sistemas informáticos que gestionan metadatos existentes en nuestro país ninguno realiza la extracción de metadatos bibliográficos desde corpus de textos en formato PDF. Se evidencia la necesidad de desarrollar sobre la base de estos una nueva aplicación que incluya nuevas funcionalidades y procesos capaces de realizar el trabajo en cuestión.
- De los sistemas estudiados a nivel mundial se utilizará como herramienta de apoyo para desarrollar la solución propuesta el *Layout Aware PDF Text*, para la conversión de los corpus en formato PDF a archivos XML.
- Se estudiaron y se seleccionaron varias herramientas para definir la plataforma de desarrollo de acuerdo a las necesidades de esta investigación.
- Se escogió como metodología de desarrollo de software para guiar el proceso la metodología XP.

Capítulo 2: Propuesta de solución

2.1. Introducción

En el capítulo anterior se seleccionó XP como la metodología a utilizar para guiar el desarrollo de la solución propuesta. En este capítulo se realiza una descripción referente a la fase planificación que propone dicha metodología. Se escriben las historias de usuario, además de hacer referencia a otros artefactos generados en esta fase. Se describen los requisitos funcionales y no funcionales, así como las principales características del sistema.

2.2. Descripción general de la propuesta

2.2.1. Requisitos de software

"Un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste. En el otro extremo, es una definición detallada y formal de una función del sistema" (41). Esta es la forma en la que Ian Somerville define un requisito de software en su libro *Ingeniería del software*. La calidad con la que se realiza la captura de requisitos afecta de manera directa todo el proceso de desarrollo del software repercutiendo en todas las fases del mismo. Además de que ayuda a tomar mejores decisiones de diseño y arquitectura.

Para la captura de requisitos funcionales y no funcionales existen cuatro técnicas fundamentales. Las técnicas más habituales en la captura de requisitos son las entrevistas, el Desarrollo Conjunto de Aplicaciones, la tormenta de ideas y la utilización de escenarios más conocidos como casos de uso (42).

A continuación se describen brevemente las técnicas utilizadas por el equipo de desarrollo para llevar a cabo la captura de requisitos funcionales y no funcionales que regirán el proceso de desarrollo.

Entrevista

La entrevista es la técnica de captura de requisitos más utilizada, y de hecho son prácticamente inevitables en cualquier desarrollo, ya que son una de las formas de comunicación más naturales entre personas.

En las entrevistas se pueden identificar tres fases: preparación, realización y análisis (43).

Las entrevistas no deben ser improvisadas. Estas cuentan con tres tareas principales y varias subtareas. A continuación se muestran cuáles son estas tareas:

- Preparación de entrevistas

1. Estudiar el dominio del problema
2. Seleccionar a las personas a las que se va a entrevistar

3. Determinar el objetivo y contenido de las entrevistas
4. Planificar las entrevistas

- Realización de entrevistas

1. Apertura
2. Desarrollo

- Preguntas abiertas
- Utilizar palabras apropiadas
- Mostrar interés en todo momento

3. Terminación

- Análisis de entrevistas

Las entrevistas realizadas al cliente fueron entrevistas informales y sirvieron de ayuda para la identificación de los requisitos tanto funcionales como no funcionales y para definir la prioridad en que estos serán implementados.

Tormenta de ideas

La tormenta de ideas es una técnica de reuniones en grupo cuyo objetivo es la generación de ideas en un ambiente libre de críticas o juicios (44). Las sesiones de la tormenta de ideas suelen estar formadas por un número de cuatro a diez participantes, uno de los cuales es el jefe de la sesión, encargado más de comenzar la sesión que de controlarla.

La tormenta de ideas consta de varias fases y varias tareas dentro de estas fases, las cuales se listan a continuación (44):

1. Preparación
2. Generación
3. Consolidación
   3.1. Revisar ideas
   3.2. Descartar ideas
   3.3. Priorizar ideas
4. Documentación

En la tormenta de ideas estuvieron presentes los autores de esta investigación los cuales conforman el equipo de desarrollo, además del cliente y el Ing. Yusniel Hidalgo Delgado el cual funcionó como jefe de sesión. En esta tormenta de ideas se propusieron los requisitos que fueron identificados por el cliente en las

entrevistas realizadas, llegando a cambios importantes en algunos de estos, además se definió el esfuerzo que conlleva la implementación de cada uno de estos requisitos.

2.2.1.1. Requisitos funcionales

Somerville plantea en su libro que los requisitos funcionales "son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares" (41).

Requisitos funcionales que se identificaron:

No	Funcionalidades
1	Añadir usuario
2	Autenticar usuario
3	Modificar usuario
4	Listar usuario
5	Eliminar usuario
6	Añadir documento
7	Listar documentos
8	Mostrar estado del documento
9	Eliminar documento
10	Extraer metadatos del documento
11	Extraer metadatos manualmente
12	Actualizar metadatos del documento
13	Guardar metadatos del documento
14	Exportar metadatos del documento en XML

Tabla 2. Requisitos funcionales del software

2.2.1.2. Requisitos no funcionales

Según Somerville, los requisitos no funcionales son las "restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares" (41).

Se identificaron como requisitos no funcionales (RNF) del sistema los siguientes:

No.	Requisitos no funcionales
Apariencia o interfaz	

RNF 1	La interfaz debe ser lo más sencilla y amigable posible para que permita la fácil interacción con el sistema por cualquier tipo de usuario.
RNF 2	Debe de presentar una correcta combinación de colores.
Usabilidad	
RNF 3	El sistema debe poder ser usado por cualquier persona que tenga conocimientos básicos de informática y del proceso de extracción de metadatos bibliográficos.
Seguridad	
RNF 4	Para permitir el trabajo de cualquier usuario con la aplicación este debe de estar correctamente autenticado.
RNF 5	Confiabilidad: La información manejada en el sistema debe de estar protegida de acceso no autorizado.
RNF 6	Integridad: La información manejada por el sistema debe ser objeto de cuidadosa protección contra la corrupción y estados de inconsistencia.
Fiabilidad	
RNF 7	Garantizar capacidad para capturar excepciones.
Portabilidad	
RNF 8	El sistema será multiplataforma, haciéndose énfasis en la plataforma Linux.
Restricciones del diseño	
RNF 9	El producto de software final debe diseñarse sobre una arquitectura MVC[26].
RNF 10	Emplear los estándares establecidos (diseño de interfaces, bases de datos y codificación).
Rendimiento	
RNF 11	Los tiempos de respuesta del sistema serán rápidos, no mayores de 5 segundos para cualquier operación realizada por el usuario.
Soporte	
RNF 12	Tiene que brindar soporte para grandes volúmenes de datos y velocidad de procesamiento.
Eficiencia	

[26] MVC: Model View Controller, por sus siglas en inglés.

RNF 13	El sistema debe ser capaz de dar respuestas a las peticiones con un nivel aceptable de desempeño. Teniendo en cuenta el nivel de concurrencia que pueda existir, debe ser capaz de prestar servicio sin que se deterioren los tiempos de respuestas.
Hardware	
RNF 14	Memoria RAM 256 MB o superior.
RNF 15	Disco duro de 160 GB o superior.
RNF 16	Procesador a 1 GHz. de velocidad de procesamiento o superior.
Software	
RNF 17	Debe tener el sistema operativo con entorno de escritorio gráfico, preferentemente Windows o GNU/Linux en cualquiera de sus versiones.
RNF 18	Debe poseer un navegador web para acceder al sistema informático, preferentemente el Mozilla Firefox 24.0 o versión superior.

Tabla 3. Requisitos no funcionales

2.3. Fase de planificación

La metodología XP define como primera fase la planificación. En esta etapa se lleva a cabo el trabajo de identificación y descripción de las historias de usuario, así como la familiarización del equipo de trabajo con las tecnologías y herramientas seleccionadas para el desarrollo del software. También el equipo de desarrollo en conjunto con el cliente especifica la prioridad en que se deben implementar las historias de usuario, así como una estimación del esfuerzo que costará implementar cada una ellas. El resultado de la fase es un plan de entregas donde se realiza una estimación de las versiones que tendrá el producto en su realización, de manera tal que guíe el desarrollo del mismo (25).

2.3.1. Historias de usuario

XP se basa fundamentalmente en las historias de usuario para representar los requisitos funcionales del sistema. Las historias de usuarios (HU) son escritas por el cliente, en su propio lenguaje, se realiza una por cada característica principal del sistema, se emplean para hacer estimaciones de tiempo y para el plan de lanzamientos, reemplazan un gran documento de requisitos y preceden a la creación de las pruebas de aceptación. La idea es que sean sencillas y que satisfagan al cliente (25).

Las HU están compuestas por varios atributos:

- Número: muestra el número correspondiente a la HU en cuestión
- Nombre: muestra el nombre de la HU

- Usuario: muestra el usuario que utilizará dicha HU
- Estimación: muestra una estimación del tiempo que demorará implementar la HU
- Iteración: muestra a la iteración que pertenece la HU, es decir en la iteración en la que se implementará esta
- Prioridad de negocio: describe la prioridad que tiene la HU para el negocio
- Riesgo en desarrollo: muestra el riesgo en el desarrollo que tiene la realización de la HU para el equipo de desarrollo
- Descripción: describe la funcionalidad que realizará el requisito funcional que está en correspondencia con esa HU
- Observaciones: describe si la implementación de la HU depende de otra con anterioridad

A continuación se muestran dos de las HU de mayor importancia en la solución propuesta por el peso que tienen en el correcto funcionamiento de la solución propuesta, las demás se pueden encontrar en el Anexo 1.

Historia de usuario	
Número: 4	**Nombre:** Mostrar estado del documento
Usuario: usuario	
Estimación: 1	**Iteración:** 2da
Prioridad de negocio: Media	**Riesgo en desarrollo:** Medio
Descripción: Muestra el estado en que se encuentran los metadatos extraídos, si hay algún campo vacío o no se capturó la información que se esperaba	
Observaciones: Previamente debe haberse realizado la HU 4	

Tabla 4. Historia de usuario número 4: Mostrar estado del documento

Historia de usuario	
Número: 5	**Nombre:** Extraer metadatos del documento
Usuario: usuario	
Estimación: 3	**Iteración:** 2da
Prioridad de negocio: Alta	**Riesgo en desarrollo:** Alto
Descripción: Realiza la extracción de los metadatos del documento o los documentos seleccionados	
Observaciones:	

Tabla 5. Historia de usuario número 5: Extraer metadatos del documento

2.3.2. Estimación de esfuerzos por historias de usuario

En el siguiente epígrafe se realiza la estimación del esfuerzo por cada HU para lo cual se hace necesario tener en cuenta que estas deben ser programadas en un tiempo entre una y tres semanas, pues así lo define la metodología escogida para guiar este proceso. Si la estimación es superior a tres semanas, se divide en dos o más historias. Si es menor de una semana, se combina con otra historia. Estas estimaciones permiten tener una medida de la velocidad del proyecto y ofrecen una guía a la cual ajustarse. Estas estimaciones las realiza el equipo de desarrollo en conjunto con el cliente. Los resultados de la estimación realizada utilizando dicha técnica son mostrados en la siguiente tabla.

Historias de usuario	Puntos de estimación (semanas)
Gestionar usuarios	1
Gestionar documentos	1
Mostrar estado del documento	1
Extraer metadatos del documento	3
Extraer metadatos manualmente	1
Actualizar metadatos del documento	2
Guardar metadatos del documento	2
Exportar metadatos en XML	2
Autenticar usuario	1

Tabla 6. Estimación de esfuerzos por historia de usuario

2.3.3. Plan de iteraciones

Luego de identificar y describir las historias de usuario y estimar el esfuerzo propuesto para la realización de cada una de ellas, se pasa a la etapa de implementación del sistema. Se seleccionan las historias de usuario que se implementan en cada iteración, de acuerdo al nivel de prioridad de las mismas, así como las posibles fechas de sus entregas.

2.3.3.1. Plan de duración de las iteraciones

Este plan tiene como objetivo principal mostrar la duración de cada iteración, así como el orden en que serán implementadas las historias de usuario en cada una de las iteraciones.

Iteraciones	Orden de las historias de usuario a implementar	Duración de las iteraciones (semanas)
Iteración 1	Gestionar usuarios	2
	Gestionar documentos	
	Autenticar usuario	
Iteración 2	Extraer metadatos del documento	4
	Actualizar metadatos del documento	
	Guardar metadatos del documento	
	Mostrar estado del documento	
Iteración 3	Exportar metadatos en XML	2
	Extraer metadatos manualmente	
Total		**8**

Tabla 7. Plan de duración de las iteraciones

2.3.4. Plan de entrega

Plan de entregas ("*Release Plan*"): el cronograma de entregas establece qué historias de usuario serán agrupadas para conformar una entrega y el orden de las mismas (45). En este plan se concentran las funcionalidades referentes a un mismo tema en módulos, esto permite un mayor entendimiento de la fase de implementación. Este plan tiene como objetivo principal el de definir el número de liberaciones que se realizarán en el transcurso del proyecto y las iteraciones que se requieren para realizar cada una. De esta forma se puede trazar un plan de entrega teniendo en cuenta dos parámetros fundamentales: el tiempo de desarrollo ideal y el grado de importancia para el cliente.

Módulos	Historia de usuario que abarca
Usuarios	Gestionar usuarios
	Autenticar usuarios
Documentos	Gestionar documentos
	Mostrar estado del documento
Metadatos	Extraer metadatos del documento
	Actualizar metadatos del documento

	Guardar metadatos del documento
	Exportar metadatos en XML
	Extraer metadatos manualmente

Tabla 8. Funcionalidades por módulos.

	Final 1ra iteración	Final 2da iteración	Final 3ra iteración
Módulos	**1ra semana de Marzo**	**4ta semana de Marzo**	**3ra semana de Abril**
Usuarios	v 1.0	v 1.2 Final	Finalizado
Documentos	v 1.0	v 1.2 Final	Finalizado
Metadatos		v 1.3 Final	v 1.4

Tabla 9. Plan de duración de entrega

2.4. Fase de diseño

La metodología de desarrollo XP plantea prácticas especializadas que reaccionan directamente en la realización del diseño para lograr un sistema robusto y reutilizable, tratando de conservar en todo momento su simplicidad, es decir, crear un diseño evolutivo que va mejorando incrementalmente y que permite hacer entregas pequeñas y frecuentes de valor para el cliente, basado principalmente en la selección de una adecuada arquitectura y el desarrollo de las tarjetas CRC.

2.4.1. Descripción de la arquitectura

Para la implementación de la solución se propone utilizar el estilo en capas (capa de presentación, capa de negocio, capa de acceso a datos y capa de datos) y el patrón Modelo-Vista-Controlador (MVC) el cual está formado por tres niveles:

- El modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio. Integra las clases que contienen las consultas a la base de datos, así como la definición de las tablas, sus relaciones y atributos.
- La vista transforma el modelo en una página web que permite al usuario interactuar con ella. Además determina la interfaz que se muestra finalmente al cliente para interactuar con la aplicación.
- El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista. Gestiona todas las peticiones del usuario y se encarga de darle respuesta.

La siguiente figura muestra las cuatro capas de la arquitectura antes mencionada. En la capa de presentación podemos observar que se encuentran todas las vistas de la aplicación, además del usuario el cual interactuará con estas vistas. En la capa de negocio se encuentran todas las entidades, clases

controladoras y validaciones de la aplicación las cuales se relacionaran con las vistas para ofrecerle al usuario las respuestas deseadas, esta capa tendrá una relación directa con la capa de acceso a datos en las cuales se encuentran las clases encargadas de interactuar con los datos mediante el ORM Doctrine. Por último esta capa se relaciona con la capa de datos en la cual se encuentra el gestor de base de datos PostgreSQL que será el encargado de almacenar todos los datos de dicha aplicación

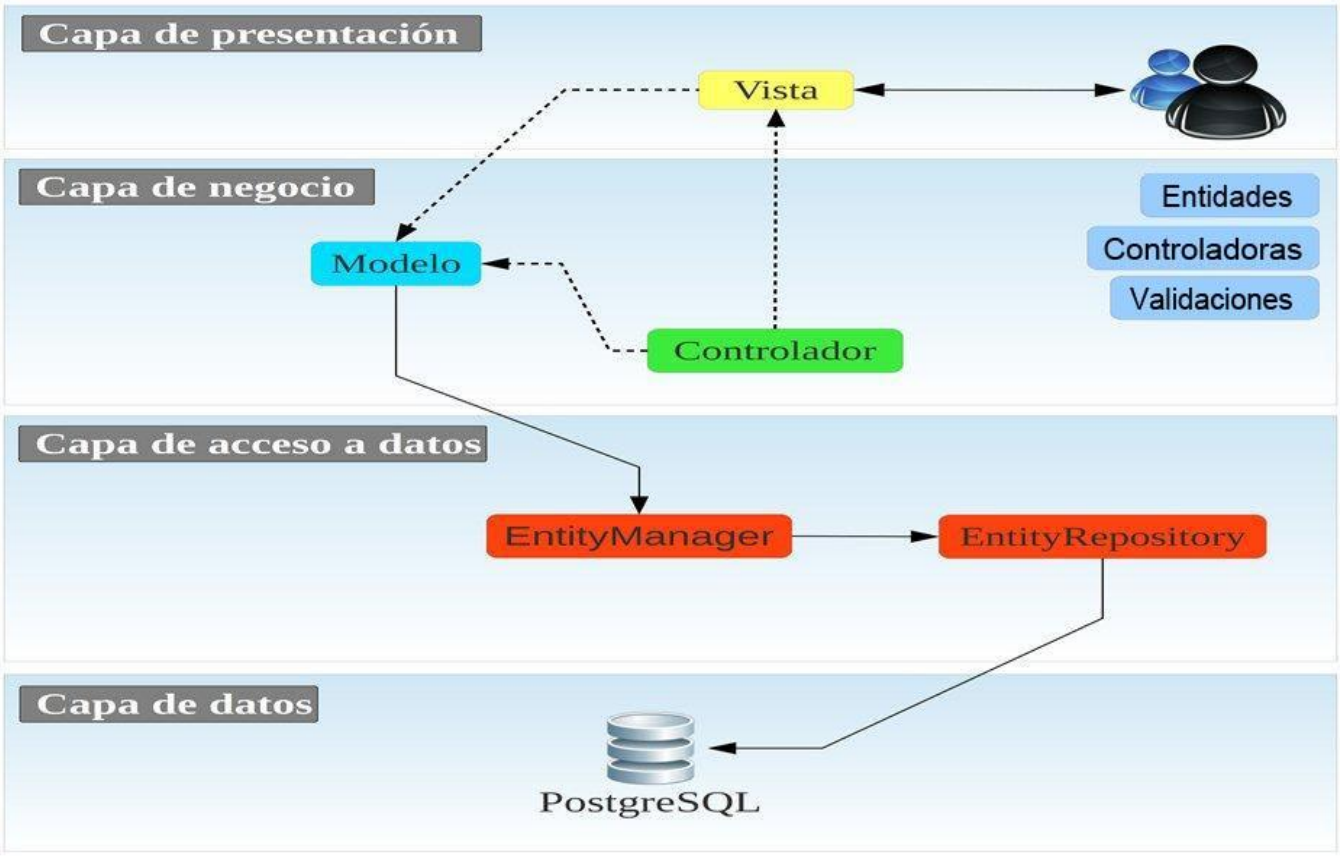


Figura 1. Diseño arquitectónico

2.4.2. Diagrama de clases persistentes

A través del diagrama de clases persistentes se definen los conceptos que se manejan en el sistema y que sirven para describir la estructura de la base de datos, es decir, en dicho modelo se representan las entidades, sus atributos y tipos, sus relaciones y las restricciones que deben cumplirse sobre ellas.

A continuación se muestran los dos diagramas de clases desarrollados. Primero se muestra el diagrama perteneciente a la clase controladora DefaultController y la relación que tiene esta con todas las clases, por

último se muestra el diagrama perteneciente a la clase DocumentoController y las relaciones que esta posee.

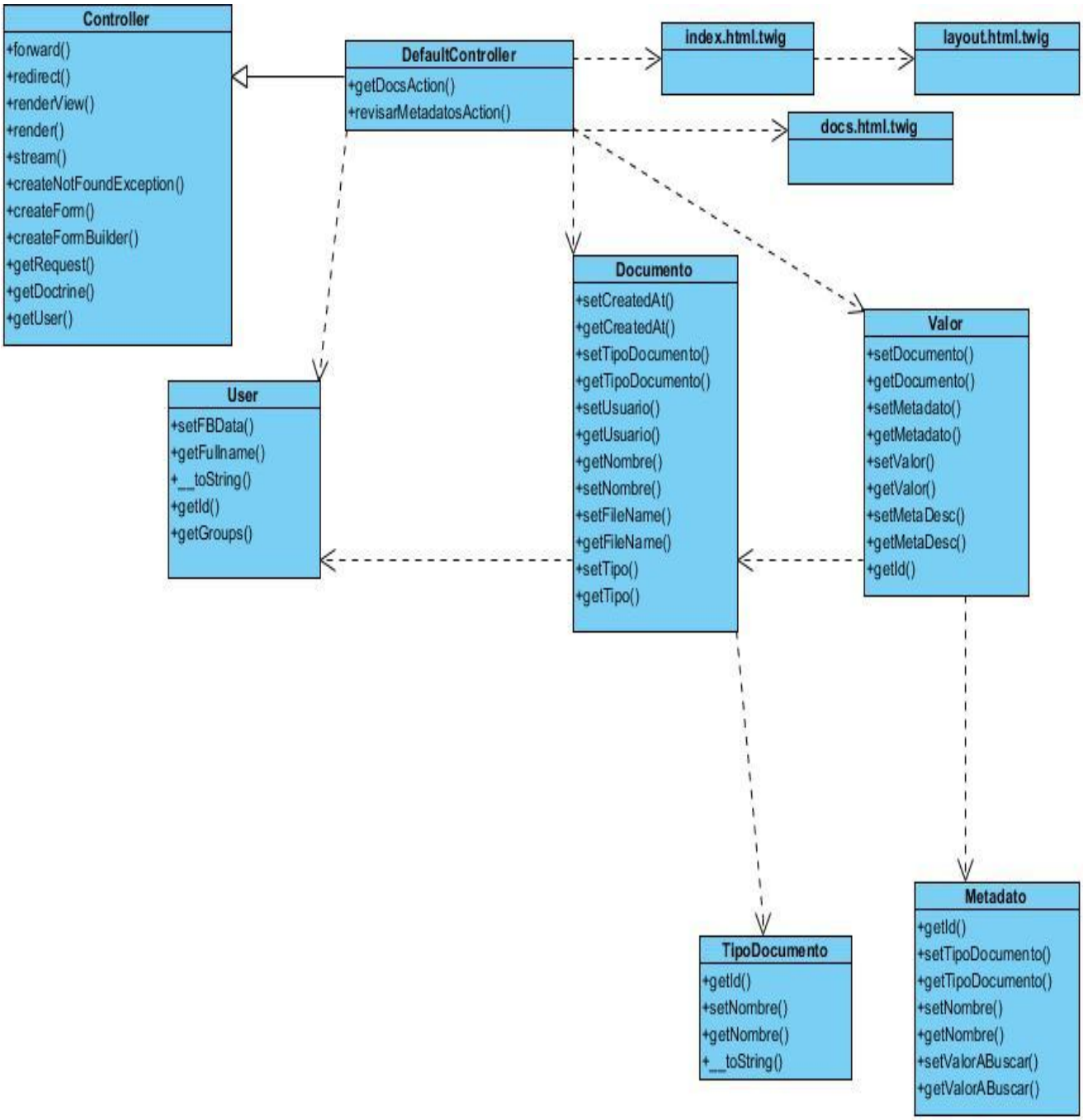


Figura 2. Diagrama de clases perteneciente a la clase DefaultController

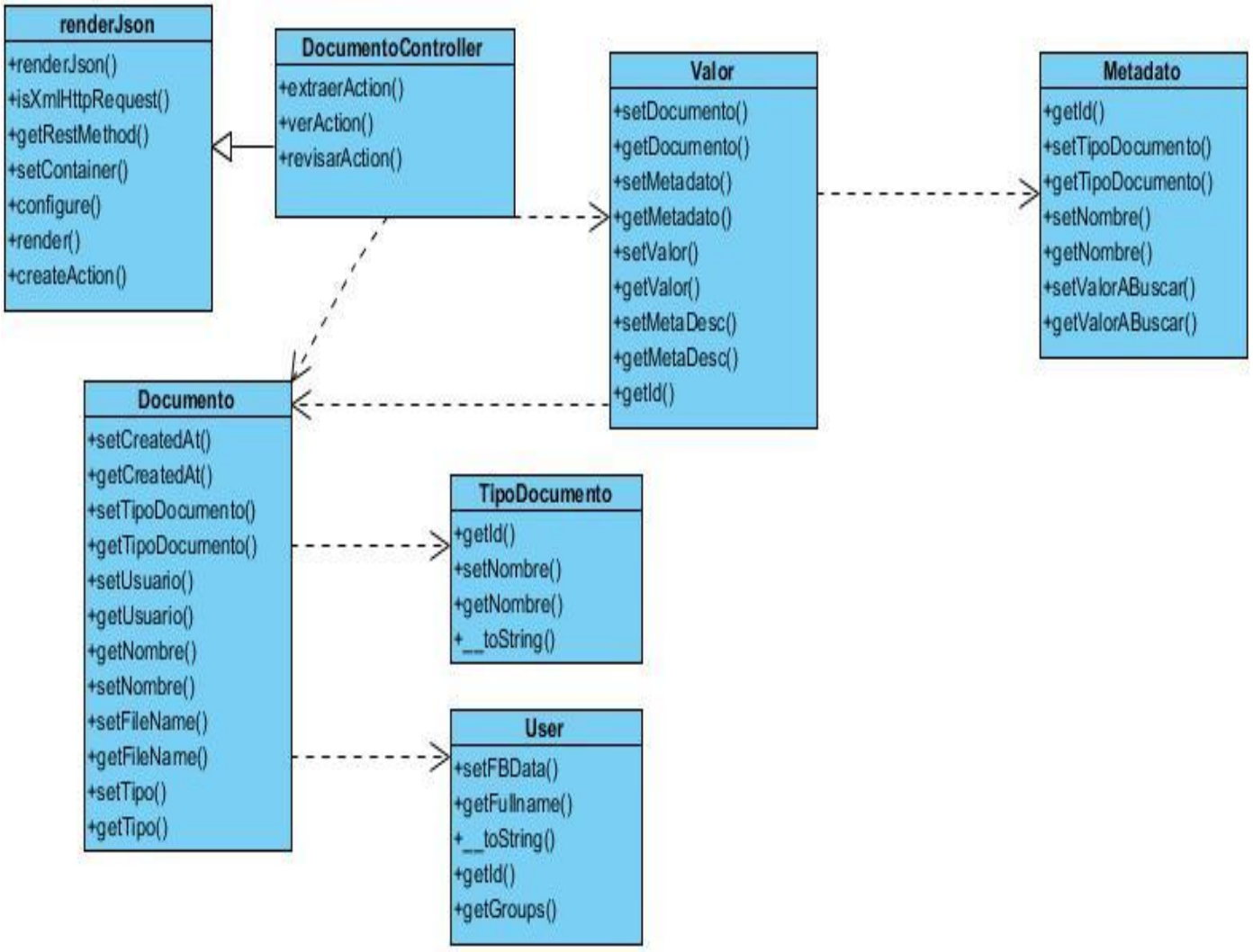


Figura 3. Diagrama de clases perteneciente a la clase DocumentoController

2.4.3. Tarjetas CRC

Las tarjetas CRC (Clase, Responsabilidad y Colaboración) son utilizadas para representar las responsabilidades de las clases y sus interacciones. Estas tarjetas permiten trabajar con una metodología basada en objetos, permitiendo que el equipo de desarrollo completo contribuya en la fase del diseño. El nombre de la clase se coloca a modo de título de la tarjeta, las responsabilidades se colocan a la izquierda y las clases que se implican en cada responsabilidad a la derecha, en la misma línea que su requisito correspondiente. Las tarjetas determinan el comportamiento de cada actividad.

A continuación se muestran las tarjetas CRC DocumentoController.php y DefaultController.php, pues son dos de las clases de mayor importancia en la solución por las responsabilidades que poseen estas, las restantes tarjetas CRC se pueden encontrar en el Anexo 2.

Clase: DocumentoController.php	
Descripción: Se encarga del proceso de extracción de los metadatos de los distintos documentos en formato PDF, además de la revisión de los metadatos	
Atributos	
Nombre	**Descripción**
Responsabilidades	
Nombre	**Colaboración**
Interviene en el proceso de extracción de metadatos	Valor, Documento, TipoDocumento, Metadato
Interviene en el flujo de revisión de los metadatos extraídos	

Tabla 10. Tarjeta CRC DocumentoController.php.

Clase: DefaultController.php	
Descripción: Se encarga de la revisión de los metadatos extraídos de un documento PDF. Además de mostrar un listado de los documentos existentes en la aplicación para el usuario logeado en ese momento	
Atributos	
Nombre	**Descripción**
Responsabilidades	
Nombre	**Colaboración**
Interviene en el flujo de revisión de los metadatos extraídos.	Valor, Documento, TipoDocumento, Metadato
Se encarga de gestionar los usuarios que tienen acceso al proceso de extracción	User

Tabla 11. Tarjeta CRC DefaultController.php

2.4.4. Diseño de la base de datos

Diagrama Entidad-Relación

El modelo de datos describe de forma abstracta como se representan los datos de una aplicación o sistema de información. Este modelo es muy importante ya que define formalmente las estructuras permitidas y las restricciones que se aplican (46). Es válido destacar que este modelo es un artefacto significativo en el diseño de la propuesta de solución, ya que las actividades que se realizan durante el proceso de pruebas quedan registradas en el sistema. Está compuesto por un conjunto de tablas, en primer lugar la tabla nombrada **Documento**, la cual guarda todos los documentos en formato PDF a los que se le quiere aplicar el proceso de extracción de metadatos. La tabla **TipoDocumento** que es la que define el tipo de documento como dice su nombre, por ejemplo: tesis, artículo científico, entre otros. La tabla **Metadato,** los tipos de metadatos definidos por el modelo Dublin Core que serán procesados por la aplicación y por último la tabla **Valor** surge por la relación de muchos a muchos entre las tablas **Documento** y **Metadato** y guarda el valor del metadato en cuestión. A continuación se presenta el Modelo de Entidad-Relación de la propuesta de esta propuesta de solución.

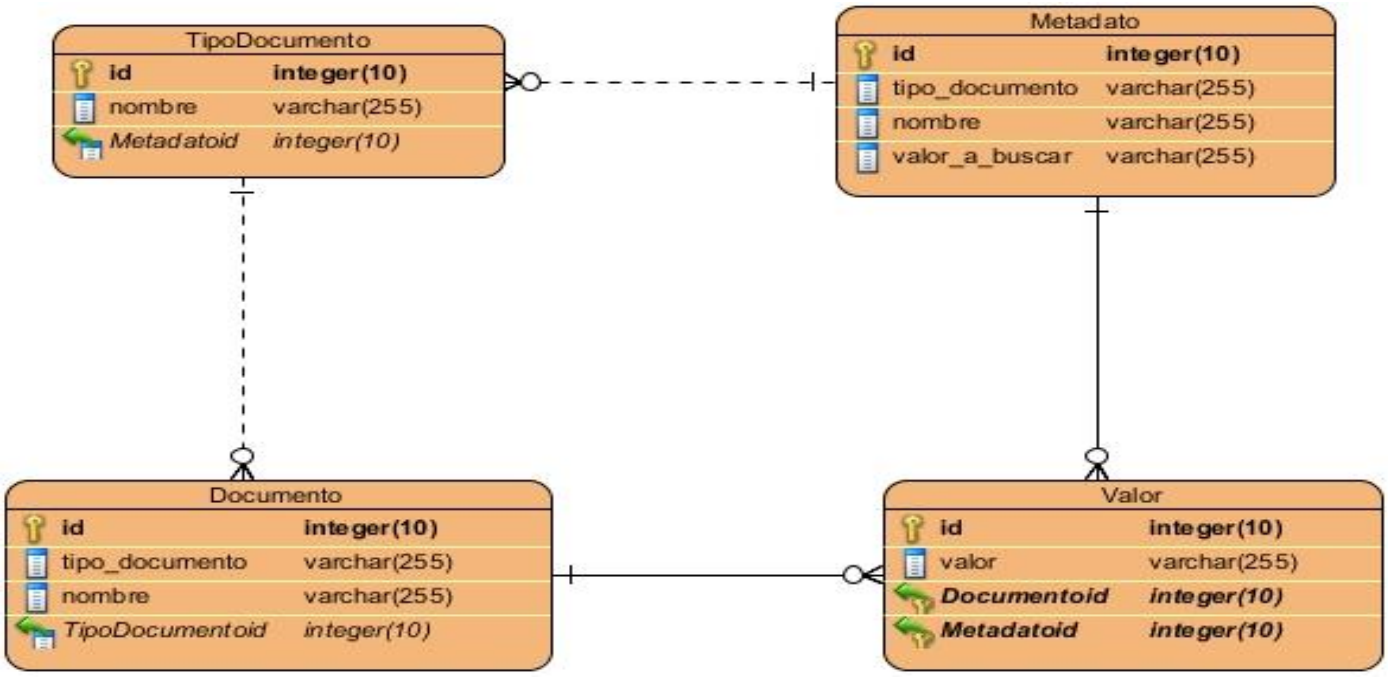


Figura 4. Diagrama Entidad–Relación

2.4.5. Estándares de codificación

XP resalta que la comunicación de los programadores es a través del código, por lo que es necesario que sigan ciertos estándares de programación para lograr un entendimiento entre los programadores y que cualquier persona del equipo de desarrollo pueda modificar cualquier parte del código, además que sea un código entendible por otros programadores que posteriormente puedan apoyarse en ese trabajo y poder desarrollar otras soluciones.

En el caso de la implementación de la herramienta que se desarrolla el estándar de codificación utilizado es el *CamelCase* que no es más que un estilo de escritura que se aplica a frases o palabras compuestas. El nombre se debe a que las mayúsculas a lo largo de una palabra en *CamelCase* se asemejan a las jorobas de un camello. El nombre *CamelCase* se podría traducir como Mayúsculas/Minúsculas Camello. El término case se traduce como "caja tipográfica", que a su vez implica si una letra es mayúscula o minúscula y tiene su origen en la disposición de los tipos móviles en casilleros o cajas. Existen dos tipos de *CamelCase*:

- *UpperCamelCase*: cuando la primera letra de cada una de las palabras es mayúscula. Ejemplo: *EjemploDeUpperCamelCase*
- *lowerCamelCase*: igual que la anterior con la excepción de que la primera letra es minúscula. Ejemplo: *ejemploDeLowerCamelCase*

En el caso de esta herramienta se trabajará con el tipo *UpperCamelCase*.

2.4.6. Patrones de diseño

“Un patrón es una descripción de un problema y la solución a la que se da un nombre, y que se puede aplicar a nuevos contextos; idealmente, proporciona consejos sobre el modo de aplicarlo en varias circunstancias” (47).

Los patrones de diseño pueden usarse durante el diseño del software. Una vez que se ha desarrollado el modelo de análisis, el diseñador puede examinar una representación detallada del problema que debe resolver y las restricciones que impone el problema (48).

Patrones generales de software para asignación de responsabilidades (GRASP)[27]

Los GRASP son patrones para la asignación de responsabilidades, que ayudan a entender el diseño de objetos. Incluso cuando se utilizan metodologías ágiles como XP, es necesario elegir cuidadosamente las

[27] General Responsibility Assignment Software Patterns

clases adecuadas y decidir cómo estas deben interactuar (47). Es por ello que resulta de vital importancia el uso de patrones GRASP. Algunos de los más importantes son (47):

- Experto: la responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase contiene toda la información necesaria para realizar la labor que tiene encomendada. Este patrón lo podemos ver claramente evidenciado en la clase Documento.php de la solución propuesta.
- Creador: este patrón como su nombre lo indica es el que crea, él guía la asignación de responsabilidades relacionadas con la creación de objetos. Este patrón se ve en cualquiera de las dos clases controladoras de la solución, pues en cualquiera de estas se realiza la instancia de un objeto para ser creado.
- Bajo acoplamiento: el sistema debe contener pocas dependencias entre clases (principio básico para la creación un buen diseño). Este patrón se evidencia en la solución propuesta pues como en su descripción dice existe un bajo acoplamiento entre las clases, además así se evidencia en la métrica Relaciones entre Clases utilizada para la validación del diseño.
- Alta cohesión: cada elemento del diseño debe realizar una labor única dentro del sistema. El empleo de este patrón proporciona refactorización en el sistema. Está evidenciado en la solución propuesta pues cada clase realiza solamente las funciones para las que fueron diseñadas y no para funciones ajenas a estas. Es decir, la clase TipoDocumento se encarga de manejar los tipos de documentos existentes en la aplicación y no tiene ninguna relación con la clase Valor que es la encargada de manejar los valores de los metadatos.
- Controlador: la responsabilidad de controlar el flujo de eventos del sistema se debe asignar a clases específicas. Esto facilita la centralización de actividades en el sistema. El siguiente patrón se evidencia en la clase DefaultController, pues esta es la encargada de controlar la mayoría de las operaciones realizadas en la solución propuesta.

Banda de los cuatro (GOF)[28]

Esta clase de patrones recibe este nombre tan particular en honor a los 4 autores del libro Design Patterns, los autores Gamma, Helm, Johnson y Vlissides.

Creacionales:

- Abstract factory (Fábrica abstracta): permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando.

[28] Gang Of Four

Cuando el marco de trabajo necesita crear un nuevo objeto para una petición, busca en la definición de la fábrica el nombre de la clase que se debe utilizar para esta tarea (49).

- Builder (Constructor): la intención es abstraer pasos de construcción de objetos de modo que diferentes implementaciones de estos pasos pueden construir diferentes representaciones de objetos (50).

Estructurales:

- Composite (Objeto compuesto): permite tratar objetos compuestos como si de uno simple se tratase. Sirve para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol. Esto simplifica el tratamiento de los objetos creados, ya que al poseer todos ellos una interfaz común, se tratan todos de la misma manera (50).
- Flyweight (Peso mosca): este patrón es un objeto que minimiza el uso de memoria mediante el intercambio de datos tanto como sea posible con otros objetos similares; es una forma de usar objetos cuando una simple representación reiterada usaría una inaceptable cantidad de memoria. A menudo algunas partes del estado de los objetos puede ser compartida, y es una práctica común para mantenerlos en estructuras de datos externas y pasarlos a objetos moscas temporalmente para ser usados (50).
- Dependency Injection (Inyección de dependencia): permite la eliminación de las dependencias incluidas en el código, por lo que es posible cambiarlas, ya sea en tiempo de ejecución o en tiempo de compilación. Este patrón puede ser usado como una forma simple de cargar componentes dinámicamente o elegir objetos simulados en entornos de prueba frente a objetos reales en entornos de producción (51).
- Facade (Fachada): una fachada es un objeto que proporciona una interfaz simplificada de un bloque de código más grande, tal como una biblioteca de clases. Una fachada permite:
- Usar, entender y probar una biblioteca de software
- Hacer la biblioteca más legible.
- Reducir la dependencia de código (49)

2.5. Fase de desarrollo

El proceso de desarrollo del software es lo que constituye la fase principal de la metodología XP, en esta fase se obtiene una primera versión del producto deseado por el cliente. Este propone una serie de prácticas que sirven para el desarrollo exitoso del mismo. Al inicio se lleva a cabo un chequeo del plan de iteraciones

por si es necesario realizar modificaciones, como parte de este plan se crean tareas de ingeniería para ayudar a organizar la implementación exitosa de las HU.

2.5.1. Diagrama de despliegue

El diagrama de despliegue muestra las relaciones físicas entre los componentes del hardware y los del software, es decir la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes del software (46).

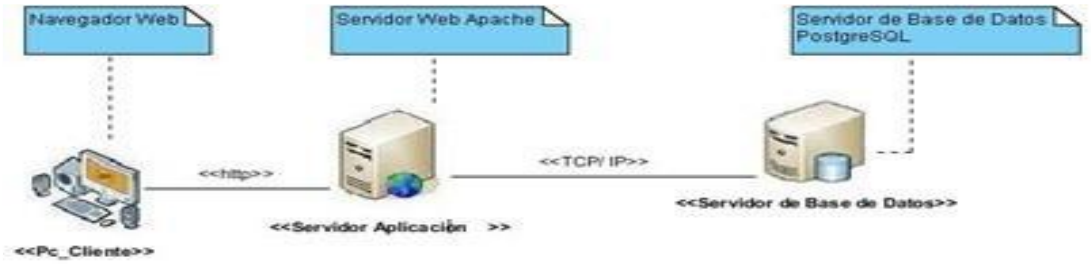


Figura 6. Diagrama de despliegue

Descripción de elementos e interfaces de comunicación

HTTPS: el protocolo de transferencia de hipertexto seguro es un protocolo que define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxy para comunicarse. Es la combinación del protocolo HTTP y protocolos criptográficos. Se utiliza para conseguir conexiones seguras en la www, habitualmente para transacciones de pagos o cada vez que se intercambie información importante en internet. (52)

TCP: el Protocolo de Control de Transmisión permite a dos anfitriones establecer una conexión e intercambiar datos. El TCP garantiza la entrega de datos, es decir, que los datos no se pierdan durante la transmisión y también garantiza que los paquetes sean entregados en el mismo orden en el cual fueron enviados. (53)

Descripción de los nodos

PC_Cliente: accede a la aplicación a través de un computador, donde es ejecutada mediante el navegador Mozilla Firefox v20.0 o superior, sobre cualquier sistema operativo. Se conecta al Servidor_de_Aplicaciones mediante el protocolo HTTPS.

Servidor_Aplicaciones: como servidor de aplicaciones se utiliza el Apache el cual está encargado de atender las solicitudes del nodo PC_Cliente, radicando en el mismo la lógica de negocio del sistema.

Servidor_Base_Datos: como servidor de base de datos se encuentra PostgreSQL v9.1, este responde a las peticiones realizadas por el Servidor_de_Aplicaciones de almacenamiento y recuperación de datos, a través del protocolo TCP.

2.5.2. Tareas de ingeniería por iteraciones

Cada HU está compuesta por una o varias tareas de ingeniería, estas no son más que pasos lógicos a seguir por el programador para realizar la implementación de una HU. A continuación se muestra las tareas a desarrollar para la iteración 1 por cada HU, las tareas de las demás iteraciones se pueden encontrar a partir del Anexo 3:

Historias de usuario	Tareas de ingeniería por historia de usuario
Autenticar usuario	Permitir al usuario introducir el usuario y la contraseña con la que podrá acceder a la aplicación
Gestionar usuario	Permitir que el administrador sea capaz de añadir, eliminar y listar los usuarios que interactúen con la aplicación
Gestionar documentos	Permitir que el usuario pueda añadir, modificar, eliminar, listar uno o varios documentos en formato PDF

Tabla 12. Tareas de ingeniería para la iteración 1

2.5.2.1. Tareas detalladas

A continuación se muestran tres descripciones detalladas, fueron escogidas las tres primeras tareas de ingeniería para la iteración 1, las de las siguientes iteraciones se pueden encontrar en el Anexo 4:

Tarea de ingeniería	
Número de tarea: 1	**Nombre de historia de usuario:** Gestionar usuarios
Nombre de la tarea: Gestionar usuarios	
Tipo de tarea: Desarrollo **(Desarrollo, Corrección, Mejora)**	**Puntos estimados (días):** 2
Fecha inicio: 3/3/2014	**Fecha fin:** 5/3/2014
Programador responsable: Osmín	
Descripción: Permite las gestión de los usuarios por del administrador (añadir, eliminar, modificar y listar usuarios)	

Tabla 13. Tarea de ingeniería detallada 1

Tarea de ingeniería	
Número de tarea: 2	**Nombre de historia de usuario:** Autenticar usuario
Nombre de la tarea: Autenticar usuario	
Tipo de tarea: Desarrollo **(Desarrollo, Corrección, Mejora)**	**Puntos estimados (días):** 1
Fecha inicio: 6/3/2014	**Fecha fin:** 7/3/2014
Programador responsable: Osmín	
Descripción: Permitir al usuario introducir su user y la contraseña con la que podrá acceder a la aplicación	

Tabla 14. Tarea de ingeniería detallada 2

Tarea de ingeniería	
Número de tarea: 3	**Nombre de historia de usuario:** Gestionar documentos
Nombre de la tarea: Gestionar documentos	
Tipo de tarea: Desarrollo **(Desarrollo, Corrección, Mejora)**	**Puntos estimados (días):** 2
Fecha inicio: 10/3/2014	**Fecha fin:** 12/3/2014
Programador responsable: Osmín	
Descripción: Permite que el usuario pueda gestionar (añadir, modificar, eliminar, listar) los documentos en formato PDF que se suben a la aplicación	

Tabla 15. Tarea de ingeniería detallada 3

2.6. Conclusiones parciales

- Se crearon las HU, así como la estimación del esfuerzo necesario para la implementación de las mismas lo que permitió realizar la estimación del tiempo y el plan de entregas de producto.
- Se definió la arquitectura y se generaron las tarjetas CRC donde se reflejan las diferentes clases que intervendrán en la aplicación con sus responsabilidades e interacciones, por lo que se sentaron las bases para comenzar con la fase de desarrollo.
- El uso de patrones de diseño permitió asignar correctamente las responsabilidades a las clases, lo que permitió la creación de un código más fácil de comprender, mantener y extender.

- Se crearon las tareas de ingeniería por cada iteración y las tareas detalladas, así como algunos diagramas para el mejor entendimiento de la propuesta, dejando así lista la solución para pasar a la etapa de pruebas y realizar la validación de la misma.

Capítulo 3: Validación de la solución propuesta

Introducción

En este capítulo se realiza la validación de la solución propuesta. Se realizará primeramente la validación de los requisitos, luego se validará el diseño de la solución propuesta, dándole paso a la verificación y validación para comprobar el correcto funcionamiento del sistema. Por último valida la propuesta de solución mediante la realización de un experimento.

3. Validación de requisitos

3.1 Métricas para la validación de la calidad de los requisitos

Una vez definidos los requisitos del sistema estos deben de ser validados para asegurar que el análisis de los mismos y los resultados obtenidos durante la definición de requisitos son correctos. La validación de requisitos tiene como misión demostrar que la declaración de los requisitos define realmente el sistema que el usuario necesita o que el cliente desea.

La validación de los requisitos examina las especificaciones para asegurar que todos los requisitos del sistema han sido determinados sin ambigüedad, sin inconsistencias, sin omisiones, que los errores detectados hayan sido corregidos y que el resultado del trabajo se ajusta a los estándares para el proceso, el proyecto y el producto (54).

Para la validación de los requisitos de la solución propuesta se decidió la utilización de tres técnicas principales:

- Estabilidad de los requisitos
- Especificidad de los requisitos
- Grado de validación de los requisitos

Estas técnicas fueron aplicadas por el equipo de desarrollo en conjunto con el cliente que solicita esta solución el cual dio su aprobación sobre los resultados obtenidos.

Para medir la estabilidad de los requisitos se aplicó la técnica de la siguiente manera:

Se identificaron 14 requisitos funcionales y ninguno de estos fue sometido a modificaciones, estas serían las variables de entrada para la fórmula que se define a continuación:

$$\boldsymbol{ETR = \left[\frac{RT - RM}{RT}\right] * 100}$$

ETR = [14-0/14]*100 = 100

Donde:

ETR: Estabilidad de los requisitos

RT: Total de requisitos definidos

RM: Cantidad de requisitos modificados

La técnica arrojó como resultado un 100% por lo se llegó a la conclusión de que los requisitos definidos son estables en su total mayoría, ya que la métrica ofrece valores entre 0 y 100, siendo el mejor valor de ETR el más cercano a 100 porque mostrará que no se han realizado cambios sobre los requisitos, estos son estables y por tanto, es confiable trabajar el análisis y el diseño sobre ellos.

Para cuantificar la **especificidad de los requisitos** se aplicó la técnica de la siguiente manera:

$$\boldsymbol{Q_1 = \frac{n_{ui}}{n_r}}$$

$Q_1 = 14/14 = 1$

Donde:

Q_1: Especificidad de los requisitos

N_r: Total de requisitos definidos

N_{ui}: Cantidad de requisitos para los que los revisores[29] tuvieron interpretaciones idénticas

Por lo tanto se puede asegurar que la especificación de los requisitos no posee ambigüedad a que se obtuvo el mayor valor que puede arrojar como resultado esta métrica. Esta característica asegura mayor calidad en el proceso de especificación.

Por último para medir que los requisitos identificados con el cliente fueron correctos se aplicó la técnica de **"Grado de validación"**, de la siguiente manera:

$$\boldsymbol{VR = \frac{n_c}{(n_c + n_{nv})}}$$

VR = 14/(14+0) = 1

Donde:

VR: Grado de validación de los requisitos

N_c: Número de requisitos que se han validado como correctos

N_{nv}: Número de requisitos no validados aún

De acuerdo al resultado obtenido y coincidiendo este con el valor óptimo de esta métrica se asegura que existe un alto nivel de corrección en la definición de los requisitos.

[29] Revisores: cliente junto con el equipo de desarrollo.

3.2 Validación del diseño

El diseño está enfocado a convertir los requisitos del cliente en un modelo que al ser implementado se obtenga el producto deseado. Generalmente constituye el punto de partida para el desarrollo de software una vez especificados los requisitos del sistema. Realizar una validación del mismo para verificar su calidad y flexibilidad garantiza una buena base para la implementación. Con este objetivo se utilizan un conjunto de métricas de software orientadas a determinar, entre otros aspectos, qué características del modelo de diseño se pueden estimar para comprobar que el sistema será fácil de implementar en cuanto a organización.

Lorenz y Kidd dividen las métricas basadas en clases en cuatro categorías: tamaño, herencia, valores internos y valores externos. Las métricas orientadas a tamaños para una clase se centran en cálculos de atributos y de operaciones, para luego promediar los valores para el sistema en su totalidad. Las métricas basadas en herencia se centran en la forma en que se reutilizan las operaciones en la jerarquía de clases. Las métricas para valores internos examinan la cohesión y asuntos relacionados con el código así como las métricas orientadas a valores externos examinan el acoplamiento y reutilización (55).

Para validar el diseño de la presente investigación se decidieron utilizar las métricas Tamaño Operacional de Clases y Relaciones entre Clases.

3.2.1 Tamaño Operacional de Clases (TOC)

Esta métrica está determinada por el número total de operaciones encapsuladas en una clase, es decir, por la cantidad de métodos y atributos asignados a esta. Grandes valores de esta medida muestran que la clase puede tener mucha responsabilidad, lo cual reducirá la reusabilidad de la misma y complicará su implementación. Por otro lado en cuanto menor sea el valor medio para esta métrica es más probable que las clases tengan menos responsabilidad y complejidad y así sea más alto el nivel de reutilización de estas. La métrica TOC evalúa los atributos de Responsabilidad, Complejidad de Implementación y Reutilización de la siguiente manera:

- **Responsabilidad:** consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto de la problemática propuesta. Un aumento del TOC implica un aumento de la responsabilidad asignada a una clase determinada. Si existen valores grandes de TOC significa que una clase tiene demasiada responsabilidad, lo cual reduciría la reutilización de la clase y hará complicada la implementación. De forma contraria sucede si los valores de TOC son pequeños.

- **Complejidad de Implementación:** consiste en el grado de dificultad que tiene implementar un diseño de clases determinado. Un aumento del TOC implica un aumento de la complejidad de implementación de una determinada clase.
- **Reutilización:** consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software. Un aumento del TOC implica una disminución del grado de reutilización de una determinada clase.

La siguiente tabla muestra el rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC.

Atributos	Categorías	Criterios
Responsabilidad	Baja	TOC< = Promedio
	Media	Promedio < TOC < 2* Promedio
	Alta	TOC> 2* Promedio
Complejidad de implementación	Baja	TOC< = Promedio
	Media	Promedio < TOC < 2* Promedio
	Alta	TOC> 2* Promedio
Reutilización	Baja	TOC< = Promedio
	Media	Promedio < TOC < 2* Promedio
	Alta	TOC> 2* Promedio

Tabla 16. Rango de valores para la evaluación técnica de los atributos de la métrica TOC.

Al analizar los resultados obtenidos luego de aplicar el instrumento de medición de la métrica TOC, los cuales se muestran en la figura 8, se puede concluir que el diseño propuesto para el desarrollo de la herramienta para la gestión de metadatos está entre los límites aceptables de calidad. Los atributos de calidad se encuentran en un nivel satisfactorio en la mayoría de las clases; de manera que se puede observar cómo se promueve la reutilización (elemento clave en el proceso de desarrollo de software) y cómo están reducidas la responsabilidad y la complejidad de implementación (48%), lo que promueve el bajo acoplamiento entre las clases y facilitará la implementación y la comprobación al desarrollador.

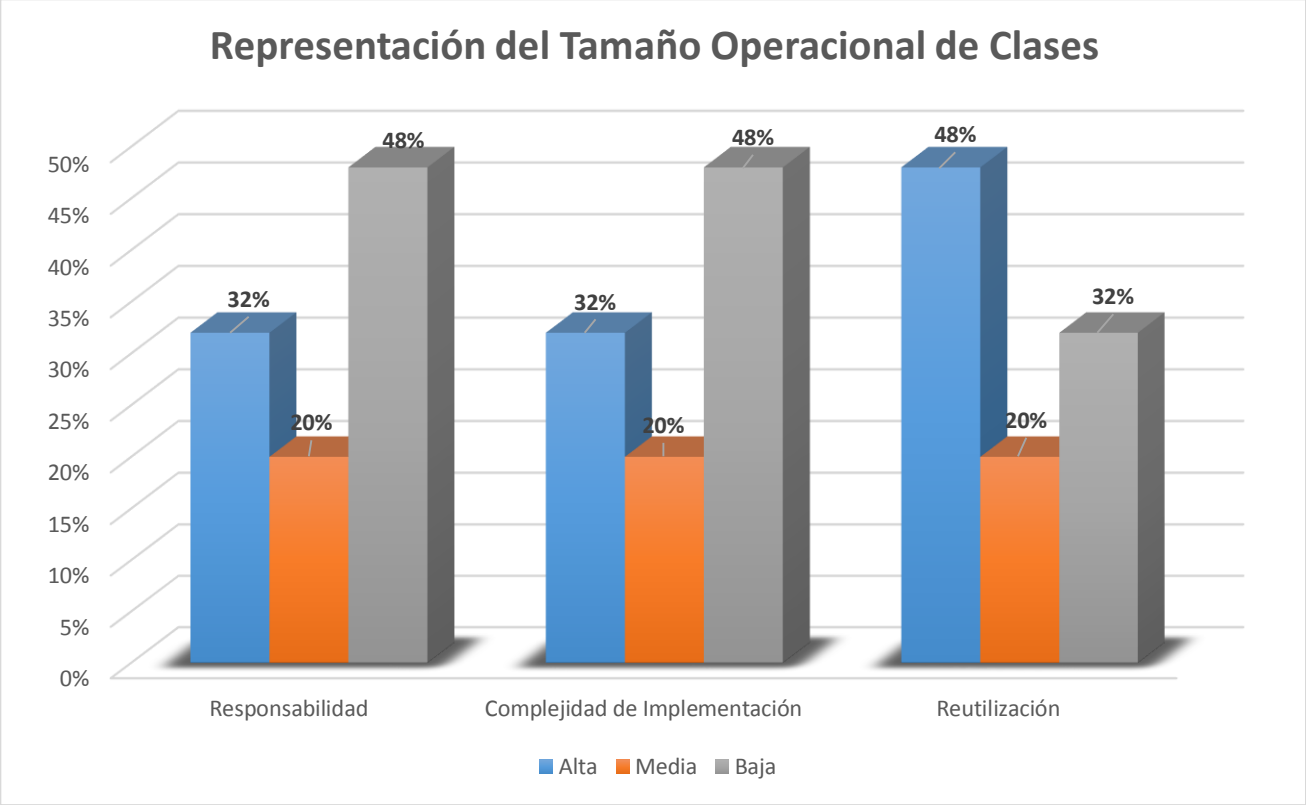


Figura 8. Gráfica de la métrica Tamaño Operacional de Clases

3.2.2 Relaciones Entre Clases (RC)

Esta métrica está dada por el número de relaciones de uso de una clase con otra. Para medir el diseño según RC se evalúan un conjunto de atributos que se relacionan a continuación (54):

- **Reutilización**: consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- **Acoplamiento:** consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.
- **Complejidad del mantenimiento:** consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.
- **Cantidad de pruebas:** consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (componente, módulo, clase, conjunto de clases, etc.) diseñado.

A continuación se muestra una tabla con el rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica RC.

Atributos	Categoría	Criterio
Acoplamiento	Bajo	RC=1
	Medio	RC=2
	Alto	RC>2
Complejidad de mantenimiento	Bajo	RC< =Promedio
	Medio	Promedio < RC < 2* Promedio
	Alto	RC> 2* Promedio
Reutilización	Bajo	RC> 2* Promedio
	Medio	Promedio < RC < 2* Promedio
	Alto	RC<= Promedio
Cantidad de pruebas	Bajo	RC<= Promedio
	Medio	Promedio < RC < 2* Promedio
	Alto	RC> 2* Promedio

Tabla 17. Rango de valores para la evaluación técnica de los atributos de la métrica RC

De los resultados obtenidos en la evaluación de la métrica RC, los cuales se pueden observar en la figura 9, se puede deducir que el diseño de la herramienta tiene una calidad aceptable, teniendo en cuenta que el 69% de las clases presentes en el diseño cuentan con una baja cantidad de relaciones de uso. Se puede observar que las clases promueven un bajo nivel de acoplamiento, así como de la complejidad de mantenimiento; igualmente la cantidad de pruebas a realizar representan un bajo por ciento. En consecuencia el grado de reusabilidad es mayor. En sentido general el resultado de la aplicación de estas métricas demuestra que las clases no se encuentran muy sobrecargadas en responsabilidad, existe además bajo acoplamiento entre las mismas y presentan un alto nivel de reutilización. Indican también que el diseño no es complejo, pues la complejidad de mantenimiento es baja, así como la cantidad de pruebas a realizar. Estos resultados sustentan la calidad del diseño.

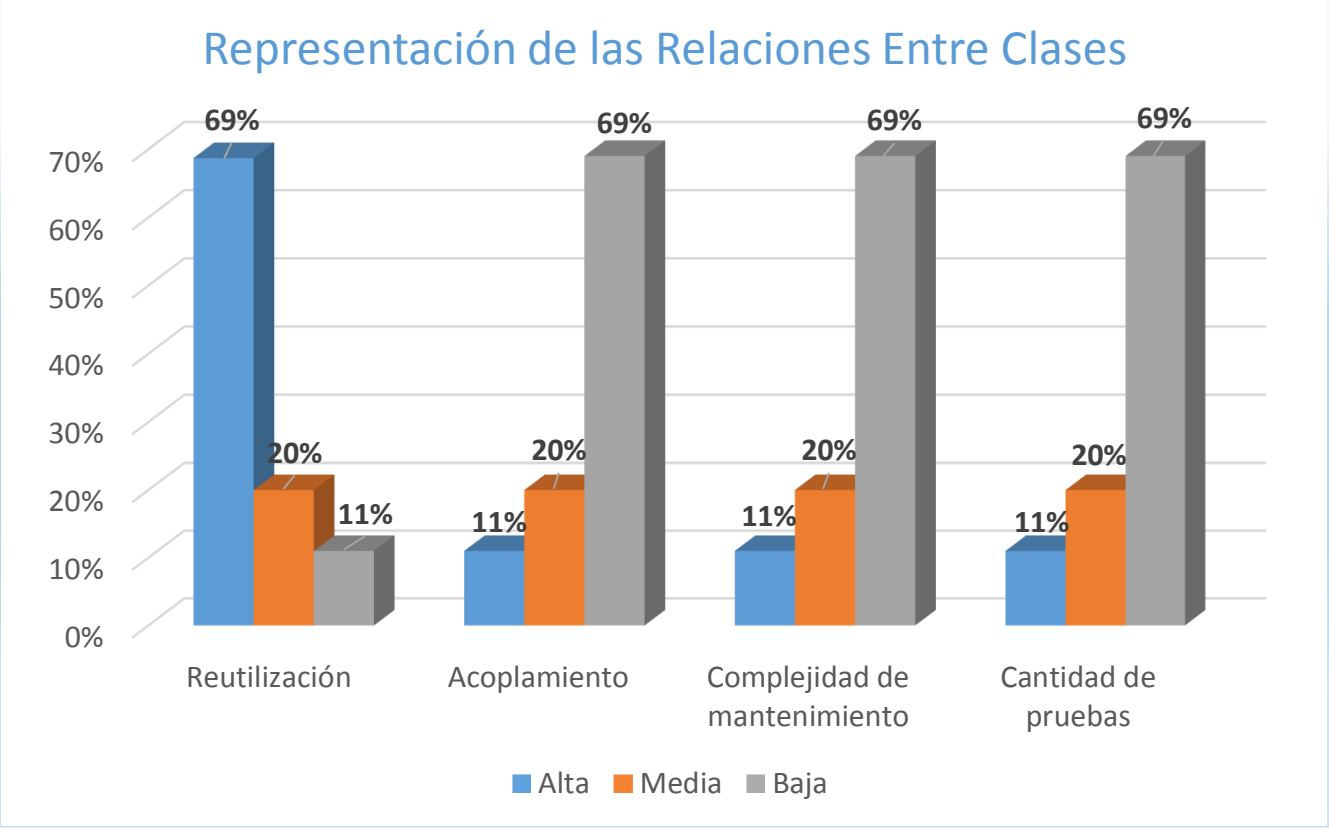


Figura 9. Gráfica de la métrica Relaciones entre Clases

3.3 Fase de pruebas

Pressman plantea que una prueba de software es la ejecución de programas de software con el objetivo de detectar defectos y fallas. Permiten comprobar y mostrar la calidad de un producto de software (48).

Pensar en utilizar la metodología XP sin tener en cuenta las pruebas no es un camino aconsejable. Estas constituyen una parte imprescindible de esta metodología, que no se debe obviar bajo ningún concepto y que debe aplicarse de manera frecuente y temprana.

La metodología XP divide las pruebas en dos grupos: pruebas unitarias, desarrolladas por los programadores, encargadas de verificar el código de forma automática y las pruebas de aceptación, destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente (26).

El objetivo fundamental que tienen las pruebas de software, es verificar los requisitos del sistema, por lo que son los propios requisitos la principal fuente de información a la hora de construir las pruebas del sistema (56).

Para la realización de las pruebas a la solución propuesta se definió una estrategia de pruebas. Las pruebas se realizarán a nivel de unidad y sistema. Los métodos utilizados serán el método de caja blanca para realizar las pruebas al código de la aplicación y el método de caja negra para verificar la interfaz y la correcta implementación de los requisitos funcionales. Estos dos métodos se llevarán a cabo mediante las técnicas de cálculo de la complejidad ciclomática por parte del método de caja blanca y se utilizará la técnica de pruebas de aceptación o funcionales por parte del método de caja negra.

3.3.1 Pruebas unitarias

Las pruebas unitarias o pruebas de caja blanca se basa en realizar pruebas al código del sistema. Para llevar a cabo esta tarea se comprueban los caminos lógicos de la aplicación mediante casos de prueba, que pongan a prueba los algoritmos implementados. Las pruebas unitarias no se le puede realizar a todo el código de la aplicación, ya que el número de caminos lógicos puede llegar a crecer de manera exponencial lo cual imposibilita realizar casos de pruebas para todo estos caminos y muchos menos se podrían procesar todos. Por este motivo las pruebas de caja blanca se realizan a los principales algoritmos o procedimientos (48).

Uno de los métodos o técnicas de prueba unitarias, es la prueba del camino básico. Esta técnica permite obtener una medida de la complejidad de un procedimiento o algoritmo y un conjunto básico de caminos de ejecución de este, los cuales luego se utilizan para obtener los casos de prueba. Esta técnica asegura que durante la prueba se ejecute al menos una vez cada sentencia del código que se está probado (48). Esta será la técnica a utilizar para realizar las pruebas unitarias a la solución propuesta.

De igual manera existen varias métricas de software para realizar pruebas unitarias, entre estas se encuentra la complejidad ciclomática, la cual será utilizada junto a la técnica explicada anteriormente. Esta métrica proporciona una medición cuantitativa de la complejidad lógica de un procedimiento. La complejidad ciclomática cuando se utiliza en el contexto del método de prueba del camino básico, el valor que se calcula como complejidad ciclomática define el número de caminos independientes[30] del conjunto básico[31] de un programa y nos da un límite superior para el número de casos de prueba que se deben realizar para asegurar que cada sentencia de código se ejecuta al menos una vez (48).

Para realizar las pruebas unitarias o de caja blanca al código de la herramienta se seleccionaron varios algoritmos que son los fundamentales. A continuación se explica todo el procedimiento de obtener los casos

[30] camino independiente: es cualquier camino del programa que introduce, por lo menos, un nuevo conjunto de sentencias de proceso o una nueva condición.
[31] conjunto básico: Es el conjunto de caminos independientes.

de prueba y poner a pruebas estos a través de un ejemplo, utilizando la técnica prueba del camino básico junto con la métrica complejidad ciclomática.

El algoritmo seleccionado lleva como nombre *revisarMetadatosAction* y tiene como finalidad poder revisar los metadatos extraídos de un documento para posibilitar cualquier arreglo que se tengan que realizar sobre estos. A continuación se muestra en la figura el código del algoritmo antes mencionado:

```
public function revisarMetadatosAction()
{
    $em = $this->getDoctrine()->getManager(); //1
    $params = $this->getRequest()->request->all(); //1

    $key = array_keys($params); //1
    $doc = $em->getRepository('Pdf2XmlBundle:Documento')->findOneById(explode('_', $key[0])[1]); //1

    foreach ($params as $key => $param) { //2
        if (count(explode('_', $key)) == 2) //3
            $valor = new Valor(); //4
        else
            $valor = $em->getRepository('Pdf2XmlBundle:Valor')->findOneById(explode('_', $key)[2]); //5

        $valor->setDocumento($doc); //6

        $meta = $em->getRepository('Pdf2XmlBundle:Metadato')->findOneById(explode('_', $key)[0]); //6
        $valor->setMetadato($meta); //6
        $valor->setMetaDesc($meta->getNombre()); //6

        $valor->setValor($param); //6

        $em->persist($valor); //6
    }

    $doc->setCompleto(Documento::TIPO_C); //7
    $em->persist($doc); //7
    $em->flush(); //7
    return $this->redirect($this->generateUrl('sonata_admin_dashboard')); //7
} //8
```

Figura 10. Código del algoritmo *revisarMetadatosAction*

Para obtener los casos de prueba a partir de este algoritmo se debe construir inicialmente el grafo de flujo correspondiente al código. A continuación se muestra en la figura dicho grafo:

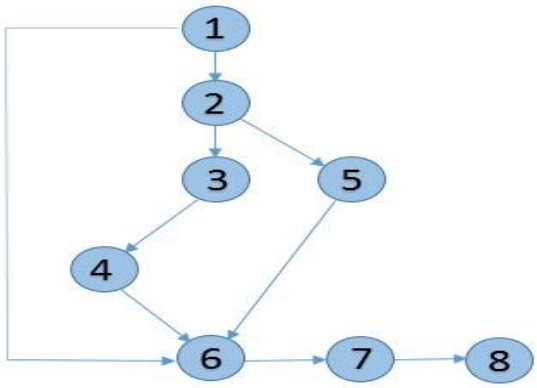


Figura 11. Grafo de flujo.

Luego se determina la complejidad ciclomática **V (G)** del grafo resultante **G**, para ello se utilizan las siguientes fórmulas:

- $\boldsymbol{V(G) = (A - N) + 2}$

Donde **A** son las aristas y **N** son los nodos:

V (G) = (9 – 8) + 2 = 1 + 2 = 3

- $\boldsymbol{V(G) = P + 1}$

Donde **P** es la cantidad de nodos predicados[32] :

V (G) = 2 + 1 = 3

- $\boldsymbol{V(G) = R}$

Donde **R** son las regiones del grafo. Las regiones de un grafo son los espacios en blanco que existan en este, incluyendo la parte exterior del grafo la cual cuenta como una región más:

V (G) = 3

El cálculo efectuado mediante las distintas fórmulas de la complejidad ciclomática ha conseguido el mismo resultado, por lo que se puede asegurar que dicha complejidad es de 3, lo que significa que existen 3 posibles caminos por donde el flujo puede circular, este valor representa el mínimo número total de casos de prueba para el procedimiento tratado. En la siguiente tabla se muestran los tres caminos existentes:

Número	Caminos básicos
1	1 – 7 – 8
2	1 – 2 – 3 – 4 – 6 – 7 – 8

[32] Nodos predicados: nodos de los cuales parten dos o más aristas.

3	1 – 2 – 5 – 6 – 7 – 8

Tabla 18. Caminos básicos

Posteriormente de haber determinado los caminos básicos se procede a ejecutar los casos de pruebas para cada uno de estos. Para definir los casos de prueba es necesario tener en cuenta:

- **Descripción:** se describe el caso de prueba y de forma general se tratan los aspectos fundamentales de los datos de entrada
- **Condición de ejecución:** se especifica cada parámetro para que cumpla una condición deseada y así ver el funcionamiento del procedimiento
- **Entrada:** se muestran los parámetros que serán la entrada al procedimiento
- **Resultado esperado:** se expone el resultado esperado que debe devolver el procedimiento después de efectuado el caso de prueba

Caso de prueba para el camino básico 1:

Descripción: Se selecciona la opción de revisar metadatos, pero no se realiza ningún cambio sobre estos.

Condiciones de ejecución: Que se encuentre registrado el documento al que se le van a revisar los metadatos y que se hayan extraídos sus metadatos con anterioridad.

Entrada: N/P

Resultado esperado: No se realiza ningún cambio sobre los metadatos revisados, y el documento pasa al estado de visualizar metadatos.

Caso de prueba para el camino básico 2:

Descripción: Se selecciona la opción de revisar metadatos por el botón de editar, luego de que el documento ya había pasado al estado de visualizar metadatos, y se realizan cambios sobre estos, luego el documento vuelve a pasar al estado de visualizar metadatos.

Condiciones de ejecución: Que el documento se encuentre registrado, y que este haya pasado anteriormente por los estados de extracción de metadatos y por el propio estado de revisión y ya se encuentra en el estado de visualizar metadatos.

Entrada: N/P

Resultado esperado: Se editan los metadatos deseados por el usuario y el documento pasa nuevamente al estado de visualizar metadatos.

Caso de estudio para el camino básico 3:

Descripción: Se selecciona la opción de revisar metadatos, el usuario realiza los cambios en los metadatos deseados y los guarda, luego pasa el documento al estado de visualizar metadatos.

Condiciones de ejecución: Que se encuentre registrado el documento al que se le desea realizar la revisión y que este ya pasara por el estado de extracción de metadatos.

Entrada: N/P

Resultado esperado: El usuario realiza la revisión de los metadatos modificando los deseados y el documento pasa al estado de visualizar metadatos.

3.3.2 Pruebas de aceptación

Las pruebas de aceptación son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido implementada correctamente. Las pruebas de aceptación se realizan con el objetivo de validar que un sistema cumple con el funcionamiento requerido y esperado por el cliente, permitiendo al mismo determinar su aprobación, teniendo en cuenta la funcionalidad y el rendimiento de la aplicación informática. Este tipo de pruebas es considerado como "pruebas de caja negra", donde cada prueba representa una salida esperada del sistema.

A continuación se muestran dos Casos de Prueba de Aceptación (CPA en lo adelante) que se realizaron para validar el sistema a partir de las HU y los requisitos del sistema, el resto de los CPA se encuentran a partir del Anexo 5.

Casos de Pruebas de Aceptación	
Código: HU5_P1	**Historias de Usuario:** 5
Nombre: Extraer metadatos del documento	
Descripción: Realiza la extracción de los metadatos del documento o los documentos seleccionados	
Condiciones de Ejecución: El usuario debe seleccionar la opción "+ Adicionar Documento" presente el panel izquierdo de la página de inicio de la aplicación El usuario debe seccionar el botón "Crear y regresar al listado" El usuario debe ir a la página de inicio y seleccionar la opción "Extraer Metadatos"	
Resultados esperados: El sistema muestra donde la opción de "Verificar errores de la extracción" donde anteriormente está la de "Extraer Metadatos"	
Evolución de la prueba: Prueba satisfactoria	

Tabla 19. Caso de Prueba de Aceptación HU5_P1

Casos de Pruebas de Aceptación	
Código: HU6_P1	**Historias de Usuario:** 6
Nombre: Actualizar metadatos del documento.	
Descripción: Se introduce o se completan los campos en el que los metadatos extraídos no coincidan con la salida esperada.	

Condiciones de Ejecución: El usuario debe seleccionar la opción “+ Adicionar Documento¨ presente el panel izquierdo de la página de inicio de la aplicación. El usuario debe seccionar el botón “Crear y regresar al listado”. El usuario debe ir a la página de inicio y seleccionar la opción “Extraer Metadatos”. El usuario debe seleccionar la opción “Verificar errores de la extracción”. El usuario debe seleccionar la opción “Editar”.
Resultados esperados: El sistema borra todos los campos de los metadatos extraídos.
Evolución de la prueba: Prueba satisfactoria

Tabla 20. Caso de Prueba de Aceptación HU6_P1

3.3.3 Análisis de los resultados

La aplicación fue sometida a cuatro iteraciones de pruebas de aceptación. En la primera iteración se encontraron un total de 23 no conformidades de las cuales 18 fueron de aplicación y 5 de documentación. Dentro de las no conformidades de aplicación 11 fueron de validación y 7 de interfaz. En cuanto a las de documentación se encontraron 4 de concordancia y 1 de ortografía. Todas estas no conformidades fueron resueltas seguidamente de haber sido detectadas lo que permitió que la aplicación pasara a una segunda iteración.

En esta segunda iteración se encontraron un total de 9 no conformidades. De estas no conformidades encontradas 6 fueron de aplicación, de ellas 4 de validación y 2 de interfaz, el resto fueron de documentación y todas de concordancia. Las no conformidades de la segunda iteración fueron resueltas y la aplicación pudo pasar a la tercera iteración en la cual se encontraron un total de 8 no conformidades. De las cuales 5 fueron de aplicación, 3 de validación y 2 de interfaz, el resto de las no conformidades fueron de documentación, 2 de ellas de concordancia y 1 de ortografía, estas no conformidades fueron resueltas seguidamente.

La aplicación pasó a la cuarta iteración en la cual no se encontró ninguna no conformidad por lo que se puede concluir que los resultados obtenidos en las pruebas de aceptación fueron satisfactorios.

A continuación se muestra una gráfica que ilustra las iteraciones antes mencionadas y las no conformidades encontradas divididas estas en cada uno de sus clasificaciones.

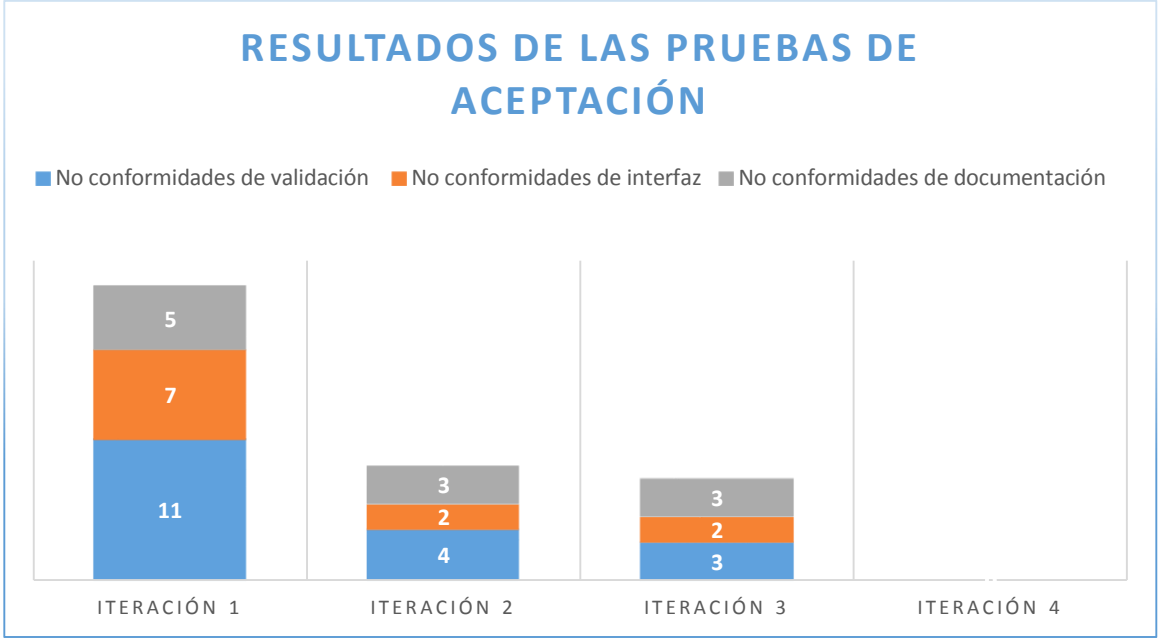


Figura 12. Gráfica resultado de las pruebas de aceptación

3.4 Validación de la solución

Para la validación de la solución propuesta se realizará un cuasi-experimento con el objetivo de medir el comportamiento de la solución implementada en un caso de estudio real. El experimento consiste en la extracción de metadatos mediante la herramienta propuesta a dos grupos de documentos distintos para luego calcular cuatro métricas de evaluación definidas a continuación.

Para la mejor comprensión del contenido de este epígrafe es necesario definir qué es un experimento. El término experimento es un estudio de investigación en el que se manipulan deliberadamente una o más variables independientes (supuestas causas) para analizar las consecuencias de esa manipulación sobre una o más variables dependientes (supuestos efectos) dentro de una situación de control para el investigador (57). La variable independiente que se va a medir en el experimento es la precisión de la herramienta en la extracción de los metadatos a partir de los documentos PDF.

Una definición más simple pero de igual manera correcta es: un estudio que involucra la manipulación intencional de una acción para analizar sus posibles efectos (57). Los experimentos están divididos en 3 grupos, pre-experimentos, causi-experimentos y experimentos puros.

3.4.1 Métricas para la evaluación

Para la realización de la validación de la solución desarrollada es necesario definir algunas métricas que evalúen los resultados obtenidos y demuestren su viabilidad. Según la revisión de las bibliografías consultadas las métricas más utilizadas son **Precisión, Exactitud, *Recall*, *F-Measure*.**

Estas métricas son utilizadas comúnmente en problemas relacionados con la recuperación de información (58) para evaluar la calidad de las consultas realizadas a un conjunto de datos determinados. También pueden ser aplicadas a problemas de clasificación supervisada (59) para determinar cuan bien ha sido predicha una clasificación realizada por un determinado clasificador. Debido a que la gestión de metadatos desde corpus en formato PDF se reduce a extraer información de estos se puede modelar como un problema de recuperación de información. Lo anterior permite seleccionar dichas métricas como elementos de validación para la investigación realizada.

3.4.1.1 Precisión

El termino Precisión (P, del inglés Precision) es la proporción de los casos predichos positivos que fueron correctos. En el caso particular de la gestión de metadatos se refiere a la proporción de los metadatos que fueron extraídos correctamente a continuación se muestra la ecuación donde se define dicha proporción.

$$\boldsymbol{P = \frac{VP}{FP + VP}}$$

Donde VP se refiere a la cantidad de metadatos extraídos correctamente de manera parcial[33] o total[34] (Verdaderos Positivos) y FP se refiere a la cantidad de metadatos que no fueron encontrados por la herramienta (Falsos Negativos).

3.4.1.2 Exactitud

El termino Exactitud (E, del inglés *Accuracy*) es la proporción de la cantidad total de predicciones que fueron correctas. En el caso del problema que estamos tratando en cuestión es la cantidad total de metadatos que fueron extraídos correctamente. A continuación se muestra la ecuación donde se define dicha proporción.

$$\boldsymbol{E = \frac{VP + VN}{P + N}}$$

Donde VP se refiere a la cantidad de metadatos extraídos correctamente (Verdaderos Positivos), P se refiere a la cantidad de metadatos que fueron extraídos (Positivos), N se refiere a la cantidad de metadatos que no fueron extraídos y VN a la cantidad de metadatos extraídos incorrectamente (Verdaderos Negativos).

[33] El metadato fue extraído por la herramienta, pero el usuario debe realizarle cambios mínimos manualmente.
[34] El metadato fue extraído correctamente por la herramienta y no es necesario realizarle ningún cambio.

3.4.1.3 *Recall*

El término *Recall* (R) es una medida de completitud y representa el porciento de predicciones correctas que fueron etiquetadas como tal. En el problema en cuestión es el porciento de metadatos extraídos. a continuación se muestra la ecuación que define dicha métrica.

$$\boldsymbol{R = \frac{VP}{VP + FN}}$$

Donde VP se refiere a la cantidad de metadatos extraídos correctamente (Verdaderos Positivos) y FN se refiere a la cantidad de metadatos extraídos incorrectamente (Falsos Negativos).

3.4.1.4 *F-Measure*

El término *F-Measure* (F) se refiere a una combinación de las métricas **Precisión** y ***Recall***, asignándoles igual peso a ambas. Seguidamente se presenta la ecuación donde se define dicha combinación.

$$\boldsymbol{F = \frac{2 * P * R}{P + R}}$$

Donde P representa el valor obtenido en la métrica **Precisión** y R el valor obtenido en la métrica ***Recall***.

3.4.2 Resultados experimentales

Se definió un experimento para probar la validez de la herramienta propuesta, en el mismo se utiliza la aplicación informática propuesta y las métricas definidas en el epígrafe anterior.

3.4.3 Entorno de ejecución

Para desarrollar el experimento se utilizó una computadora (PC) con las siguientes características:

Hardware:

- Procesador: core i5 2.30 GZ
- Memoria ram: 4gb

Software:

- Sistema operativo: Opensuse 13.1
- Navegador: Mozilla Firefox 24.0
- Servidor web: Apache2
- Gestor de base de datos: PostgretSQL 9.2

3.4.3.1 Diseño experimental

Para el causi-experimento realizado se definieron dos grupos de documentos diferenciados por los tipos de documentos existentes en cada grupo, estos son “Tesis de diploma” y “Artículos científicos”. Cada grupo tiene un total de 10 documentos en formato PDF y para cada tipo de documento se definieron un conjunto

de metadatos. Para los documentos de tipo “Tesis de diploma” se definieron los metadatos: Título, Autor, Tutor y Resumen; en el caso de los documentos de tipo “Artículos científicos” se definieron los metadatos: Título, Autor y Resumen. A continuación se muestra una tabla con el diseño experimental realizado.

R G_1	X_1	O_1
R G_2	X_1	O_2

Tabla 21. Diseño experimental propuesto

La simbología utilizada en la tabla anterior es la siguiente:

- **R**: Asignación de los documentos en cada uno de los grupos en dependencia a la clasificación antes mencionada.
- $\mathbf{G_x}$: Grupo de participantes, en el caso del experimento en cuestión los grupos de documentos. Cada grupo de documentos tiene un total de 10 documentos.
- $\mathbf{X_1}$: Tratamiento o estímulo, en este caso la herramienta para la gestión de metadatos.
- $\mathbf{O_x}$: Observación realizada luego de la aplicación del estímulo.

Seguidamente se aplica la solución propuesta a cada uno de los grupos y se calculan cada una de las métricas definidas en las secciones anteriores, facilitando la utilización del mismo diseño experimental para cada cálculo de las métricas. Las observaciones realizadas representan la medición de las métricas definidas para cada grupo correspondiente. Todas estas mediciones se basan en la comparación de los resultados obtenidos por la herramienta informática propuesta. El objetivo de dichas observaciones es medir el grado de efectividad de dicha herramienta para extraer los metadatos definidos.

3.4.3.2 Análisis de los resultados

Para el análisis de los resultados normalmente se comparan los datos obtenidos en el experimento con los valores reales de los sistemas con características afines que fueron estudiados con anterioridad en el estado del arte. Esto evidenció que no existía ningún otro sistema con características similares a la solución en cuestión, por lo que se mostrarán a continuación los resultados de la evaluación de cada una de las métricas calculadas y se explicará el significado de cada uno de ellos.

A continuación se muestra una tabla propuesta por Kaufmann en la que se determinan los porcientos de aceptabilidad de dichas métricas para cuando se produce el caso que se explicó anteriormente. Los valores de aceptabilidad de las métricas son dados en correspondencia con el valor obtenido por la media total de los dos grupos conformados para la realización del experimento decidido (59).

Métrica	Alta	Media	Baja
Precisión	X > 58%	30% < X < 58%	X < 30%
Exactitud	X > 65%	25% < X < 65%	X < 25%
Recall	X > 55%	25% < X < 55%	X < 25%
F-Measure	X > 60%	35% < X < 60%	X < 35%

Tabla 22. Rango de los valores de aceptabilidad de las métricas para la evaluación de la validación

De acuerdo al diseño experimental propuesto se realizaron varias pruebas encaminadas al cálculo de las métricas definidas con anterioridad para los dos grupos conformados. Dichas pruebas se realizaron para detectar factores que influyen en los resultados obtenidos.

Las primeras pruebas estuvieron encaminadas al cálculo de la métrica **Precisión** en cada grupo de documentos conformado. Se puede apreciar en los resultados mostrados que en el grupo de "Tesis de diploma" la métrica obtiene valores más altos que en el otro grupo conformado. Esto se debe a que en este grupo se encontraron correctamente un mayor número de metadatos de los que fueron definidos con anterioridad. Aun así se puede llegar a la conclusión que los porcientos de precisión de la herramienta son altos ya que la media total es de un 62%. A continuación se muestra la gráfica que refleja dichos resultados.

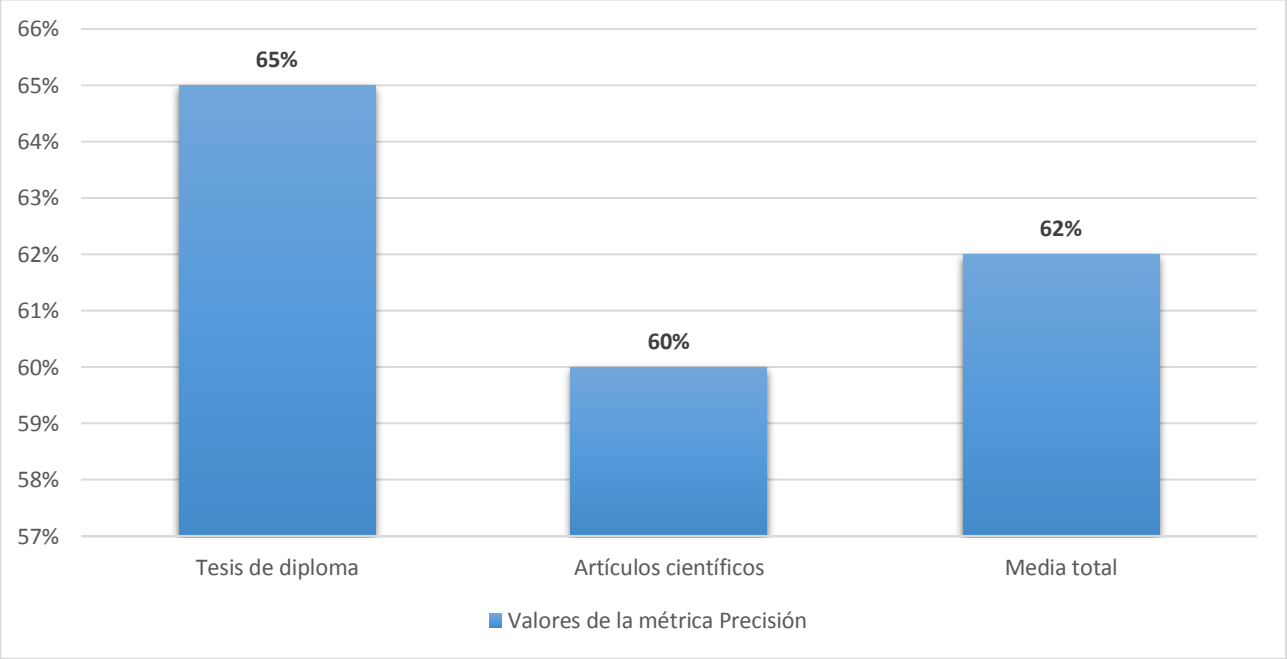


Figura 13. Gráfica de la métrica Precisión

Luego de analizados los valores obtenidos para la métrica **Precisión** se pasó al cálculo de la métrica **Exactitud.** Al igual que en la métrica anterior se puede apreciar que la métrica **Exactitud** obtiene valores más altos en el grupo de documentos de tipo “Tesis de diploma”. Aun así en ambos casos los valores de dicha métrica son altos. Teniendo en cuenta ambos valores se puede calcular la media de dicha métrica para los dos grupos analizados siendo esta de un 70%, este valor se puede clasificar como alto teniendo en cuenta la tabla de rangos propuesta con anterioridad. A continuación se muestra una gráfica que ilustra dichos resultados.

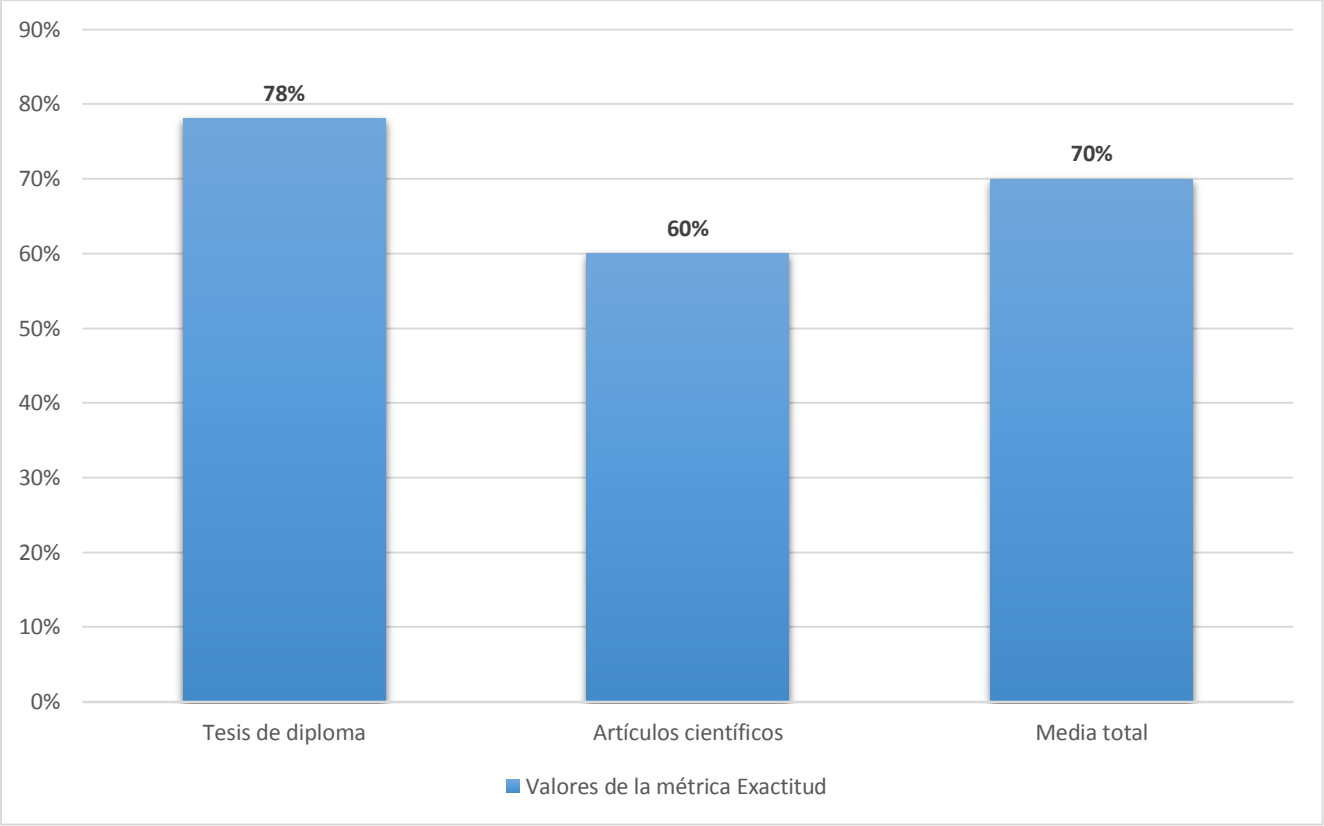


Figura 14. Gráfica de la métrica Exactitud

La tercera métrica que se calculó fue ***Recall***. Dicha métrica fue la que arrojó los valores más bajos, nuevamente mayores valores en el caso de los documentos de tipo “Tesis de diploma”, siendo este de un 55%, en el otro caso el resultado obtenido fue de un 39%, para una media de un 49% siendo este considerado un valor medio según la tabla de rangos antes propuesta. A continuación se muestra la gráfica con dichos resultados.

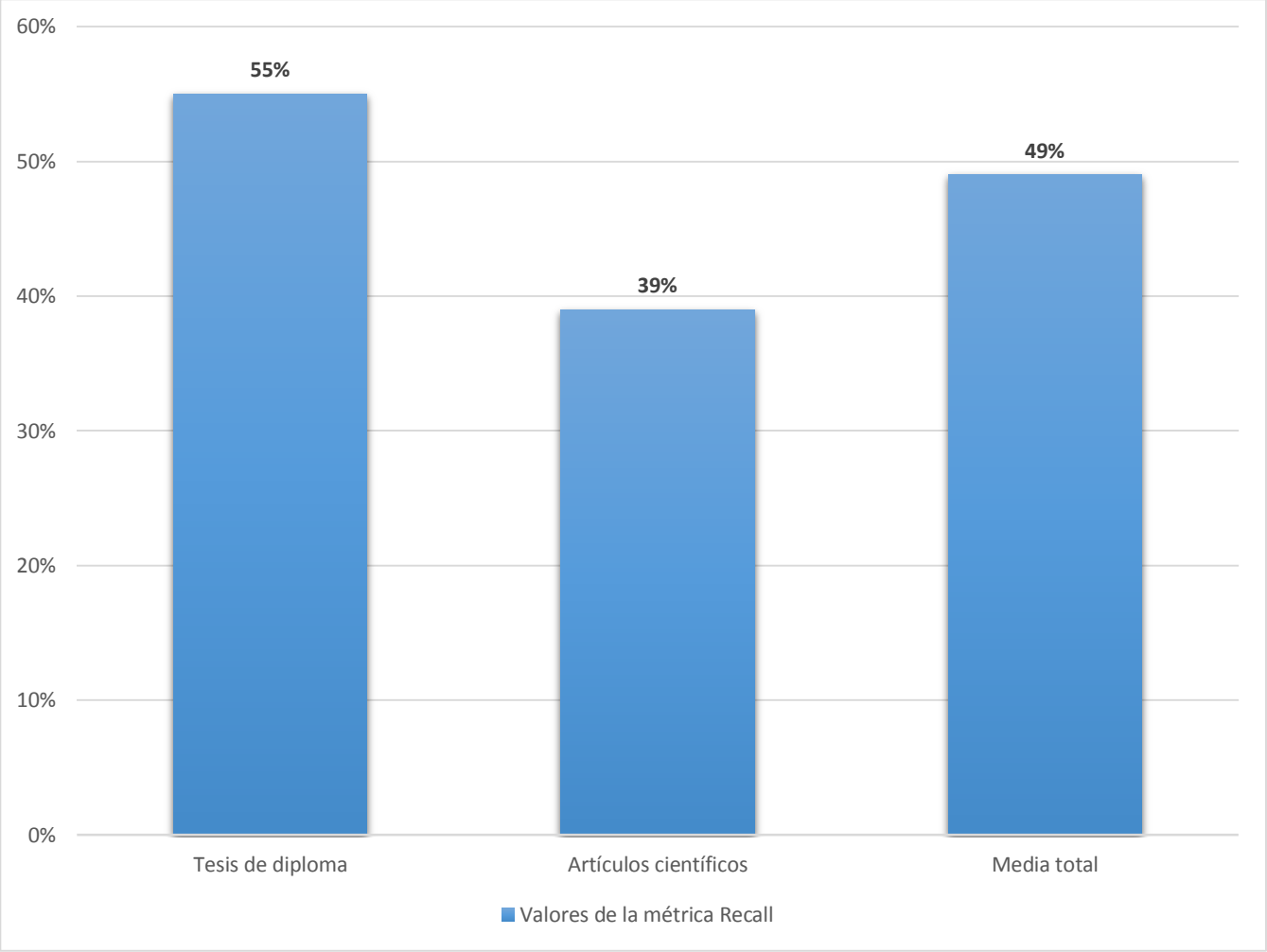


Figura 15. Valores de la métrica *Recall*

La última de las métricas definidas y que se calculó fue la ***F-Measure***. Los resultados obtenidos por esta métrica son de igual manera altos, siendo nuevamente mayores en el grupo de "Tesis de diploma" obteniéndose un resultado de un 65% en comparación con un 48% en el grupo de "Artículos científicos". A continuación se muestra en gráfica los resultados antes mencionados.

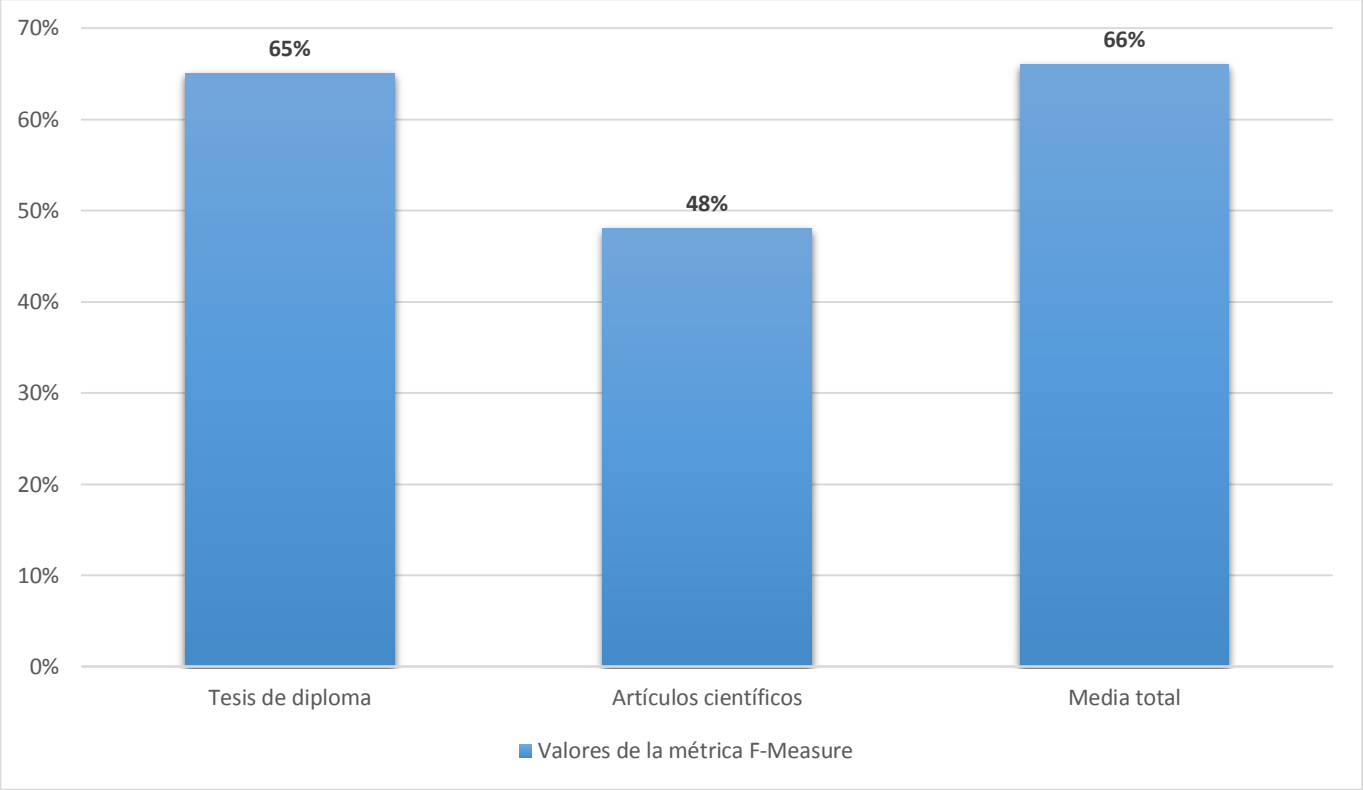


Figura 16. Valores de la métrica *F-Measure*

Con el análisis de los resultados anteriores se puede comprobar en qué condiciones la herramienta propuesta se comporta de una mejor manera, así como también los peores casos de ejecución. En tres de las cuatro métricas calculadas se obtuvieron resultados altos, mientras que en la otra se obtuvieron resultados medios, lo que demuestra el correcto funcionamiento de la solución obtenida. Esto también se evidencia con el Aval de aceptación, la cual se encuentra en el anexo 6, emitido por el cliente que solicitó dicha investigación. De esta forma se puede concluir que la solución propuesta cumplió satisfactoriamente la etapa de validación y está lista para ser usada en un entorno real.

3.5 Conclusiones parciales

- La validación de los requisitos mediante el uso de métricas permitió asegurar que el análisis de los mismos y los resultados obtenidos durante su definición son correctos.
- La validación del modelo de diseño mediante el uso de las métricas Tamaño Operacional de Clases y Relaciones entre Clases permitió determinar que la mayor parte de las clases son reutilizables, poco dependientes entre sí y con responsabilidades bien delimitadas.

- Las pruebas unitarias realizadas a varios algoritmos del sistema posibilitaron la detección de errores, obteniéndose finalmente resultados satisfactorios una vez corregidos estos.
- Las pruebas de aceptación posibilitaron la corrección de los errores detectados durante este proceso obteniéndose finalmente luego de varias iteraciones, el correcto funcionamiento de la aplicación desarrollada para la gestión de metadatos bibliográficos desde documentos PDF.
- Se realizó la validación de la aplicación en conjunto con el cliente asegurando la adecuada correspondencia entre las funcionalidades desarrolladas y los requisitos iniciales. Lo cual demostró la satisfacción del cliente, que quedó plasmada en la carta de aceptación redactada por este.

Conclusiones

Mediante la realización de este trabajo se llegó a las siguientes conclusiones:

- Luego de realizar un análisis de las herramientas informáticas existentes para la gestión de metadatos se llegó a la conclusión que ninguna satisface completamente las necesidades existentes en el proyecto Observatorio de Gobierno Electrónico.
- Los artefactos propuestos por la metodología escogida, los cuales se generaron durante la investigación sirvieron de base para la correcta implementación de la propuesta de solución.
- La propuesta de solución fue implementada en el lenguaje de programación utilizando el paradigma de la POO a la cual se le realizaron las pruebas de software que propone la metodología escogida para regir el proceso de desarrollo obteniendo buenos resultados.
- Se validó la solución propuesta mediante pruebas de software y el cumplimiento de los objetivos de la investigación, lo que permitió comprobar el correcto funcionamiento de la herramienta obtenida.

Recomendaciones

- Realizar un estudio más profundo que permita incorporar nuevas funcionalidades a la herramienta.
- Realizar un estudio más profundo en cuanto a las herramientas existentes que realizan la conversión de documentos PDF a XML en aras de encontrar una mejor opción para realizar esta tarea.
- Luego de poner el sistema en explotación por parte de los especialistas que lo utilicen tomar experiencias y retroalimentarse para el mejoramiento futuro de la herramienta.
- Estudiar la posibilidad de incorporarle rasgos de la inteligencia artificial a dicha herramienta.

Referencias

1. **Real Academia de la Lengua Española.** *Diccionario de la Lengua Esppañola.* Madrid : s.n., 2001.
2. **Zapater, José Javier Samper.** *Ontologías para servicios web semáticos de información de tráfico: Descripción y herramientas de explotación. Departamento de Informática, Universidad de Valencia.* Valencia : Universidad de Valencia, 2005.
3. **Méndez, Eva y Senso, Jose A.** Introducción a los metadatos. [En línea] 2004. [Citado el: 24 de Noviembre de 2013.] http://www.sedic.es/autoformacion/metadatos/programa.htm.
4. **Pérez, Chantal.** Estudio de Lenguística del Español. Explotación de los córpora textuales informátizados para la creación de bases de datos terminológicas basadas en el conocimiento. [En línea] Universidad Autónoma de Barcelona. [Citado el: 14 de Febrero de 2014.] http://elies.rediris.es-elies18/23.html.
5. **Catalunya, Fundación para la Universidad Oberta de.** EduKanda. Recursos formativos en red. [En línea] [Citado el: 19 de Febrero de 2014.] http://www.edukanda.es/mediatecaweb/data/zip/614/PID_00154130/web/main/m1/v2_2_1.html..
6. **Testa, Patricia y Ceriotto, Paula .** *Descripción de Objetos Digitales:Metadatos.* s.l. : Universidad Nacional de Cuyo Centro Universitario.
7. **vicomtech.** vicomtech. [En línea] [Citado el: 24 de Febrero de 2014.] http://www.vicomtech.org/t4/e11/procesamiento-del-lenguaje-natural.
8. **BioMed Central Ltd unless otherwise stated.** Source Code For Biology aand Medicine. [En línea] 2014. [Citado el: 24 de Abril de 2014.] http://www.scfbm.org/content/7/1/7/abstrac. 1751-0473.
9. **findbestopensource.com.** Best Open Source. [En línea] 2010. [Citado el: 24 de Abril de 2014.] http://www.findbestopensource.com/product-lapdftext.
10. **Mendeley.** Mendeley Desktop: Herramienta para gestionar metadatos. [En línea] Mendeley. [Citado el: 24 de Noviembre de 2014.] http://www.infobiblio.es/tutorial-de-mendeley.
11. **Biblioteca Nacional de Nueva Zelanda.** Metadata Extraction Tool. [En línea] 2003. [Citado el: 12 de Diciembre de 2013.] http://www.natlib.govt.nz/services/get-advice/digital-libraries/metadata-extraction-tool.
12. **Informática 64.** Informática 64. [En línea] [Citado el: 17 de Noviembre de 2013.] http://www.informatica64.com/foca.aspx.
13. *Ciencias de la Información.* **Cedeño, Dunia Maria Colomé, Rodríguez, Mirurgia Ávila y Sentí, Vivian Estrada.** s.l. : Revista Científica del Instituto de la Información Científica y Tecnológica.
14. *SyGMe: Sistema para la Gestión de Metadatos Geográficos.* **Padrón, Liset García y Sosa, Alejandro Leandro.** La Habana : s.n., 2013.
15. *HERRAMIENTA PARA LA RECOLECCIÓN DE METADATOSBIBLIOGRÁFICOS MEDIANTE EL PROTOCOLO OAI-PMH.* **Delgado, Yusniel Hidalgo, y otros.** La Habana : s.n., 2013. ISBN:978-959-7213-02-4.
16. **Madrid, Universidad Carlos III .** Metadatos Recuperación y Acceso a la Información. [En línea] 25 de Febrero de 2014. http://www.metadatos-xmlrdf.com/metadatos/dublin-core.
17. **Flower, Martin y Scot, Kendall.** *UML gota a gota.* s.l. : Pearson Education, 1999.
18. **Stevens, Perdita y Pooley, Rob.** *Utilización de UML en Ingeniería del Software con Objetos y Componentes.* s.l. : Addison-Wesley Publishing Company, 2002.
19. **Booch, Grady.** *The Unified Modeling Language User Guide.* s.l. : Pearson Education, 2005.

20. **Visual Paradigm.** Visual Paradigm for UML 8.0 Released. [En línea] 16 de Agosto de 2010. [Citado el: 19 de Noviembre de 2013.] http://www.visual-paradigm.com/aboutus/newsreleases/vpuml80.jsp.
21. —. Round-trip Code Engineering. [En línea] [Citado el: 19 de Noviembre de 2013.] http://www.visual-paradigm.com/product/vpuml/features/roundtripcodeengine.jsp#cpproundtrip.
22. **WIELENGA, GEERTJAN.** Top 10 New Features in NetBeans IDE 7.3. [En línea] [Citado el: 22 de Noviembre de 2013.] https://blogs.oracle.com/geertjan/entry/top_10_new_features_in.
23. **Piattini, Mario.** *Análisis y diseño detallado de aplicaciones informáticas de gestión.* Madrid : RA-MA, 2007. 9788478977765.
24. **Beck, Kent.** *Extreme Programming Explained.* 1999. 0201616416.
25. —. *Una explicación de la Programación extrema: aceptar el cambio.* 2002. 8478290559.
26. **Beck, Kent y Flower, Martin.** *Planning Extreme Programming.* 2000. 0201710919.
27. **Chromium Developer Documentation.** User Experience (Chromium Developer Documentation). [En línea] [Citado el: 20 de Noviembre de 2013.] http://dev.chromium.org/user-experience..
28. **GOODGER, BEN.** Welcome to Chromium. [En línea] 2 de Septiembre de 2008. [Citado el: 21 de Noviembre de 2013.] http://blog.chromium.org/2008/09/welcome-to-chromium_02.html..
29. **FETTE, IAN.** Google Chrome, Chromium, and Google. [En línea] 1 de Agosto de 2008. [Citado el: 21 de Noviembre de 2013.] http://blog.chromium.org/2008/10/google-chrome-chromium-and-google.html.
30. **Chromium.** The Chromium Projects. [En línea] [Citado el: 22 de Noviembre de 2013.] http://www.chromium.org/Home..
31. **Murphey, Rebecca.** *Fundamentos de JQuery.*
32. **Cochran, David.** *Twitter Bootstrap Web Development.* s.l. : Packt Publishing, 2012. ISBN 978-1849518826..
33. **The Apache Software Foundation.** Licenses. [En línea] [Citado el: 24 de Noviembre de 2013.] http://www.apache.org/licenses/.
34. —. What is the Apache HTTP Server Project? [En línea] [Citado el: 24 de Noviembre de 2013.] http://httpd.apache.org/ABOUT_APACHE.html.
35. —. Configuration Files. [En línea] [Citado el: 25 de Noviembre de 2013.] http://httpd.apache.org/docs/current/configuring.html.
36. **PHP.** What is PHP? [En línea] [Citado el: 4 de Diciembre de 2013.] http://php.net/.
37. **Potencer, Fabien y Weaver, Ryan.** *Symfony 2, el libro oficial.* s.l. : Creative Commons Atribución - Compartir igual (CC BY-SA) 3.0, 2013.
38. **Doctrine.** Object Relational Mapper. [En línea] [Citado el: 3 de Diciembre de 2013.] http://www.doctrine-project.org/projects/orm.html.
39. —. Doctrine Query Language. [En línea] [Citado el: 5 de Diciembre de 2013.] http://docs.doctrine-project.org/en/2.1/reference/dql-doctrine-query-language.html.
40. **PostgresSQL.** PostgreSQL: About. [En línea] [Citado el: 5 de Diciembre de 2013.] http://www.postgresql.org/about/.
41. **Somerville, Ian.** *Ingeniería del software.* Madrid : Pearson Education S.A., 2005. ISBN:84-7829-074-5.
42. **Toro, Amador Durán y Jiménez, Beatriz Bernárdez.** *Metodología para la Elicitación de Requisitos de Sistemas Software.* Sevilla : s.n., 2000.
43. **Piattini, M. G., y otros.** *Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión.* 1996.

44. **Raghavan, S., Zelesnik, G. y Ford, G.** *Lecture Notes on Requirements Elicitation.* s.l. : Software Engineering Institute, Carnegie Mellon University, 1994. Educational Materials CMU/SEI–94–EM–10.
45. **Joskowicz, José.** *Reglas y Prácticas en eXtreme Programming.* España : s.n., 2008.
46. **Larman, Craig y Hall, Prentice.** *El modelo de diseño. UML y Patrones 2ª Edición.* 2003.
47. **Larman, Craig.** *UML y Patrones Introducción al análisis y diseño orientado a objetos y al proceso unificado.* México : PRENTICE HAL, 1999.
48. **Pressman, Roger.** *Ingeniería de software, un enfoque práctico.* s.l. : Quinta edición: McGraw-Hill, 2002. 8448132149.
49. **Freeman, Eric, y otros.** *Head First Design Patterns.* s.l. : O'Reilly, 2004. ISBN 978-0-596-00712-6..
50. **Gamma, Erich, y otros.** *Design Patterns: Elements of Reusable Object-Oriented Software.* s.l. : Addison-Wesley Publishing Company, 1995. ISBN ISBN 0-201-63361-2..
51. **symfony.es.** Inyección de dependencias en Symfony 2. [En línea] 2 de Abril de 2009. [Citado el: 13 de Diciembre de 2013.] http://www.symfony.es/noticias/2009/04/02/inyeccion-de-dependencias-en-symfony-2/..
52. **Suárez, Armijos y Naufredo, Javier.** *Estudio de las características, funcionamiento, ventajas y técnicas utilizadas en los optimizadores WAN.* s.l. : Universidad Politécnica Salesiana, 2012.
53. **Kioskea.net.** ***Protocolo TCP.*** **[En línea] [Citado el: 20 de abril de 2014.] http://es.kioskea.net/contents/281-protocolo-tcp..**
54. Fornaris, Maite Sánchez y Rabí, Dayana Alcantara. ***Propuesta de una guía de métricas para evaluar el desarrollo de los Sistemas de Información Geográfica.*** **2010.**
55. Lorenz, Mark y Kidd, Jeff. ***Object-Oriented Software Metrics.*** **Nueva Jersey : Englewood Cliffs, 1994.**
56. Gutiérrez JJ, Escalona MJ, Mejías M, Torres J. ***Pruebas del Sistema en Programación Extrema.*** **s.l. : Departamento de Lenguajes y Sistemas Informáticos. Universidad de Sevilla, 2006.**
57. Correa C. Rojas M. Grau, R. ***Metodología de la Investigación. 2da edición.*** **Ibágue, Colombia : EL POIRA, Editores, 2004.**
58. McDonald, John Tait (Eds.) Sharon. ***Advances in Information Retrieval.***
59. Kaufmann, Morgan. ***Data Mining. Concepts and Techniques.***
60. Piñeiro, Antonio de la Rosa y Senso, José A. ***El concepto de Metadato. Algo más que descripcioón de recursos electrónicos.*** **s.l. : Universidad de Granada, 2003.**
61. ***OAI-PMH: Protocolo para la transmisión de contenidos en Internet.*** **Barrueco, José Manuel. Barcelona, España : Biblioteca de Ciencias Sociales. Universidad de Valencia.**
62. Figueroa, Roberth G., Solís, Camilo J. y Cabrera, Armando A. ***Metodologías Tradicionales vs. Metodologías Ágiles.*** **s.l. : Universidad Técnica Particular de Loja, Escuela de Ciencias Informáticas, 2009.**
63. JUnit. JUnit.org. [En línea] 2012. [Citado el: 17 de Noviembre de 2013.] http://www.junit.org.
64. Boehm, Barry W. ***GUIDELINES FOR VERIFYING AND VALIDATING SOFTWARE REQUIREMENTS AND DESIGN SPECIFICATIONS.*** **Redondo Beach, CA, USA : s.n., 1979.**
65. CIBERTEC. ***Pruebas de Software.*** **España : s.n.**
66. Somerville, Ian. ***Ingeniería de Software.*** **s.l. : Pearson Educación, 2002.**

Anexos

Anexo 1: Historias de usuario

Historia de usuario	
Número: 1	**Nombre:** Autenticar usuario
Usuario: usuario	
Estimación: 1	**Iteración:** 1ra
Prioridad de negocio: Baja	**Riesgo en desarrollo:** Medio
Descripción: El usuario podrá autenticarse para acceder a algunas opciones del sistema. Datos para autenticarse: Usuario (obligatorio) Contraseña (obligatoria)	
Observaciones: Previamente debe haberse realizado la HU 2.	

Tabla 1. Historia de usuario número 1: Autenticar usuario.

Historia de usuario	
Número: 2	**Nombre:** Gestionar usuario
Usuario: administrador	
Estimación: 1	**Iteración:** 1ra
Prioridad de negocio: Alta	**Riesgo en desarrollo:** Medio
Descripción: El administrador podrá añadir, eliminar, listar y modificar los usuarios que van a interactuar con la aplicación.	
Observaciones:	

Tabla 2. Historia de usuario 2: Gestionar usuario.

Historia de usuario	
Número: 3	**Nombre:** Gestionar documentos
Usuario: usuario	
Estimación: 1	**Iteración:** 1ra
Prioridad de negocio: Media	**Riesgo en desarrollo:** Medio
Descripción: El usuario podrá añadir, eliminar y listar documentos en formato PDF.	
Observaciones:	

Tabla 3. Historia de usuario 3: Gestionar documentos.

Historia de usuario	
Número: 6	**Nombre:** Actualizar metadatos del documento
Usuario: usuario	
Estimación: 2	**Iteración:** 2da
Prioridad de negocio: Alta	**Riesgo en desarrollo:** Medio
Descripción: Se introduce o se completa los campos en el que los metadatos extraídos no coincidan con la salida esperada.	
Observaciones: Previamente debe haberse realizado la HU 4.	

Tabla 4. Historia de usuario 6: Actualizar metadatos del documento.

Historia de usuario	
Número: 7	**Nombre:** Guardar metadatos del documento
Usuario: usuario	
Estimación: 2	**Iteración:** 2da
Prioridad de negocio: Media	**Riesgo en desarrollo:** Bajo
Descripción: Se guardan los metadatos extraídos.	
Observaciones:	

Tabla 5. Historia de usuario 7: Guardar metadatos del documento.

Historia de usuario	
Número: 8	**Nombre:** Exportar metadatos a Dublin Core
Usuario: usuario	
Estimación: 2	**Iteración:** 3ra
Prioridad de negocio: Alta	**Riesgo en desarrollo:** Medio
Descripción: Lista todos los documentos generados por documento.	
Observaciones:	

Tabla 6. Historia de usuario 8: Exportar metadatos a Dublin Core.

Anexo 2: Tarjetas CRC

Anexo 3: Tareas de ingeniería por iteraciones

Historias de usuario	Tareas de ingeniería por historia de usuario
Mostrar el estado del documento.	Permitir que el usuario vea si el documento tuvo algún error durante la extracción de los metadatos de este.
Extraer metadatos del documento	Permitir que el usuario pueda extraer los metadatos de uno o varios documentos a la vez. Permitir que metadatos se desean extraer del documento.
Guardar metadatos en el documento	Permitir que los metadatos extraídos puedan ser guardados.
Actualizar metadatos del documento	Permitir que el usuario pueda ingresar manualmente los metadatos que no pudieron ser extraídos.

Tabla 1. Tareas de ingeniería para la iteración 2.

Historias de usuario	Tareas de ingeniería por historia de usuario
Exportar metadatos del Dublin Core	Permitir que los metadatos extraídos sean exportados a Dublin Core.

Tabla 2. Tareas de ingeniería para la iteración 3.

Anexo 4: Tareas detalladas

Tarea de ingeniería	
Número de tarea: 4	**Nombre de historia de usuario:** Actualizar metadatos del documento
Nombre de la tarea: Actualizar documentos	
Tipo de tarea: Desarrollo **(Desarrollo, Corrección, Mejora)**	**Puntos estimados (días):** 3
Fecha inicio: 13/3/2014	**Fecha fin:** 17/3/2014
Programador responsable: Rubén	
Descripción: Permitir que el usuario pueda ingresar manualmente los metadatos que no pudieron ser extraídos.	

Tabla 1. Tarea de ingeniería detallada 4.

Tarea de ingeniería	
Número de tarea: 5	**Nombre de historia de usuario:** Guardar metadatos del documento
Nombre de la tarea: Guardar metadatos	
Tipo de tarea: Desarrollo	**Puntos estimados (días):** 1

(Desarrollo, Corrección, Mejora)	
Fecha inicio: 18/3/2014	**Fecha fin:** 19/3/2014
Programador responsable: Rubén	
Descripción: Permite que los metadatos extraídos puedan ser guardados.	

Tabla 2. Tarea de ingeniería detallada 5.

Tarea de ingeniería	
Número de tarea: 6	**Nombre de historia de usuario:** Mostrar estado del documento
Nombre de la tarea: Mostrar estado del documento	
Tipo de tarea: Desarrollo **(Desarrollo, Corrección, Mejora)**	**Puntos estimados (días):** 2
Fecha inicio: 20/3/2014	**Fecha fin:** 24/3/2014
Programador responsable: Rubén	
Descripción: Permite que el usuario ver si el documento tuvo algún error durante la extracción de los metadatos de este.	

Tabla 3. Tarea de ingeniería detallada 6.

Tarea de ingeniería	
Número de tarea: 7	**Nombre de historia de usuario:** Extraer metadatos del documento
Nombre de la tarea: Seleccionar metadatos a extraer	
Tipo de tarea: Desarrollo **(Desarrollo, Corrección, Mejora)**	**Puntos estimados (días):** 1
Fecha inicio: 25/3/2014	**Fecha fin:** 26/3/2014
Programador responsable: Rubén	
Descripción: Permite seleccionar que metadatos se desean extraer del documento.	

Tabla 4. Tarea de ingeniería detallada 7.

Tarea de ingeniería	
Número de tarea: 8	**Nombre de historia de usuario:** Extraer metadatos del documento

Nombre de la tarea: Extraer metadatos	
Tipo de tarea: Desarrollo **(Desarrollo, Corrección, Mejora)**	**Puntos estimados (días):** 7
Fecha inicio: 27/3/2014	**Fecha fin:** 4/4/2014
Programador responsable: Rubén	
Descripción: Permite extraer los metadatos de uno o varios documentos	

Tabla 5. Tarea de ingeniería detallada 8.

Tarea de ingeniería	
Número de tarea: 9	**Nombre de historia de usuario:** Exportar los metadatos en Dublin Core
Nombre de la tarea: Exportar metadatos	
Tipo de tarea: Desarrollo **(Desarrollo, Corrección, Mejora)**	**Puntos estimados (días):** 2
Fecha inicio: 7/4/2014	**Fecha fin:** 9/5/2014
Programador responsable: Osmín	
Descripción: Permite que los metadatos extraídos sean exportados a Dublin Core(los campos de son título, autores, resumen, tipo de documento, tutores).	

Tabla 6. Tarea de ingeniería detallada 9.

Anexo 5: Casos de prueba de aceptación

Casos de Pruebas de Aceptación	
Código: HU1_P1	**Historias de Usuario:** 1
Nombre: Autenticar usuario	
Descripción: El usuario podrá autenticarse para acceder a algunas opciones del sistema. Datos para autenticarse: Usuario (obligatorio) Contraseña (obligatoria)	
Condiciones de Ejecución: El usuario debe entrar su usuario y su contraseña. El usuario podrá decidir si desea que su contraseña sea recordada.	
Resultados esperados: El sistema muestra una interfaz con el panel de administración a su izquierda a la derecha un área donde deberían estar los últimos documentos publicados, además de que en la esquina superior derecha aparecerá el nombre del usuario logeado con las opciones a que puede acceder según sus privilegios.	
Evolución de la prueba: Prueba satisfactoria	

Tabla 1. Casos de prueba de aceptación HU1_P1.

Casos de Pruebas de Aceptación	
Código: HU2_P1	**Historias de Usuario:** 2
Nombre: Gestionar usuarios	
Descripción: El administrador podrá agregar un usuario.	
Condiciones de Ejecución: El usuario debe entrar Menú Principal. El usuario debe entrar en la etiqueta usuario. El usuario debe entrar en la etiqueta Agregar nuevo. El usuario debe llenar los campos obligatorios y marcar la casilla de habilitado para garantizar su acceso a la aplicación.	
Resultados esperados: El sistema lanza un mensaje de confirmando que se ha añadido correctamente el usuario.	
Evolución de la prueba: Prueba satisfactoria	

Tabla 2. Caso de prueba de aceptación HU2_P1.

Casos de Pruebas de Aceptación	
Código: HU2_P2	**Historias de Usuario:** 2
Nombre: Gestionar usuarios	
Descripción: El administrador podrá eliminar un usuario.	
Condiciones de Ejecución:	

El usuario debe entrar Menú Principal. El usuario debe entrar en la etiqueta usuario. El usuario selecciona la casilla correspondiente al usuario que desea eliminar. El usuario presiona el botón OK El usuario vera se le mostrara una ventana nueva para confirmar la operación y deberá seleccionar el botón “SI ,ejecutar”
Resultados esperados: El sistema deberá lanzar un mensaje que diga “Los elementos seleccionados fueron eliminados satisfactoriamente” y un listado con los usuarios que quedan.
Evolución de la prueba: Prueba satisfactoria

Tabla 3. Caso de prueba de aceptación HU2_P2.

Casos de Pruebas de Aceptación	
Código: HU2_P3	**Historias de Usuario:** 2
Nombre: Gestionar usuarios	
Descripción: El administrador podrá editar un usuario.	
Condiciones de Ejecución: El usuario debe entrar Menú Principal. El usuario debe entrar en la etiqueta usuario. El usuario debe seleccionar el nombre del usuario que desea editar. El usuario selecciona los campos que desea actualizar. El usuario selecciona el botón actualizar o actualizar y cerrar.	
Resultados esperados: El sistema deberá lanzar un mensaje de confirmación que diga “Elemento actualizado satisfactoriamente.”	
Evolución de la prueba: Prueba satisfactoria	

Tabla 4. Caso de prueba de aceptación HU2_P3.

Casos de Pruebas de Aceptación	
Código: HU2_P4	**Historias de Usuario:** 2
Nombre: Gestionar usuarios	
Descripción: El administrador podrá listar los usuarios del sistema.	
Condiciones de Ejecución: El usuario debe entrar Menú Principal. El usuario debe entrar en la etiqueta usuario.	
Resultados esperados: El sistema muestra una lista con los usuarios que tienen acceso al sistema.	
Evolución de la prueba: Prueba satisfactoria	

Tabla 5. Caso de prueba de aceptación HU2_P4.

Casos de Pruebas de Aceptación	
Código: HU3_P1	**Historias de Usuario:** 3
Nombre: Gestionar documentos	
Descripción: El usuario podrá añadir el documento en formato PDF al que se le va a realizar el proceso de extracción.	

Condiciones de Ejecución: El usuario debe seleccionar la opción “agregar nuevo” correspondiente a la etiqueta documento. El usuario debe llenar los campos obligatorios. El usuario debe seccionar el botón “crear y regresar al listado”.
Resultados esperados: El sistema mostrara el mensaje “Elemento creado satisfactoriamente” y mostrara una lista de los documentos subidos.
Evolución de la prueba: Prueba satisfactoria

Tabla 6. Caso de prueba de aceptación HU3_P1.

Casos de Pruebas de Aceptación	
Código: HU3_P2	**Historias de Usuario:** 3
Nombre: Gestionar documentos	
Descripción: El usuario podrá modificar cualquier documento subido a la aplicación.	
Condiciones de Ejecución: El usuario debe seleccionar la opción “listar” correspondiente a la etiqueta documento. El usuario debe seleccionar la opción de “Editar” correspondiente al documento que desea modificar. El usuario modificara los campos que le sean convenientes. El usuario seleccionara el botón “Actualizar”.	
Resultados esperados: El sistema lanzara un mensaje “Elemento actualizado satisfactoriamente”.	
Evolución de la prueba: Prueba satisfactoria	

Tabla 7. Caso de prueba de aceptación HU3_P2.

Casos de Pruebas de Aceptación	
Código: HU3_P3	**Historias de Usuario:** 3
Nombre: Gestionar documentos	
Descripción: El usuario podrá borrar cualquier documento subido a la aplicación.	
Condiciones de Ejecución: El usuario debe seleccionar la opción “listar” correspondiente a la etiqueta documento. El usuario debe seleccionar la opción “Borrar” correspondiente al documento que desea borrar. El usuario seleccionara el botón “Si, borrar” para confirmar la acción de borrado.	
Resultados esperados: El sistema mostrar un mensaje “Elemento eliminado Satisfactoriamente” y mostrara una lista con los documentos que queda.	
Evolución de la prueba: Prueba satisfactoria	

Tabla 8. Caso de prueba de aceptación HU3_P3.

Casos de Pruebas de Aceptación	
Código: HU4_P1	**Historias de Usuario:** 4
Nombre: Mostrar estado del documento.	
Descripción: Muestra si el estado en que se encuentran los metadatos extraídos, si hay algún campo vacío o no capturo la información que se esperaba.	
Condiciones de Ejecución:	

El usuario debe seleccionar la opción “+ Adicionar Documento¨ presente el panel izquierdo de la página de inicio de la aplicación. El usuario debe seccionar el botón “crear y regresar al listado”. El usuario debe ir a la página de inicio y seleccionar la opción “Extraer Metadatos”. El usuario debe seleccionar la opción “Verificar errores de la extracción”.
Resultados esperados: El sistema muestra unas etiquetas con los campos de los metadatos que se seleccionaron para la extracción.
Evolución de la prueba: Prueba satisfactoria

Tabla 9. Caso de prueba de aceptación HU4_P1.

Casos de Pruebas de Aceptación	
Código: HU5_P1	**Historias de Usuario:** 5
Nombre: Extrae metadatos del documento.	
Descripción: Realiza la extracción de los metadatos del documento o los documentos seleccionados.	
Condiciones de Ejecución: El usuario debe seleccionar la opción “+ Adicionar Documento¨ presente el panel izquierdo de la página de inicio de la aplicación. El usuario debe seccionar el botón “crear y regresar al listado”. El usuario debe ir a la página de inicio y seleccionar la opción “Extraer Metadatos”.	
Resultados esperados: El sistema muestra donde la opción de “Verificar errores de la extracción” donde anteriormente está la de “Extraer Metadatos”.	
Evolución de la prueba: Prueba satisfactoria	

Tabla 10. Caso de prueba de aceptación HU5_P1.

Casos de Pruebas de Aceptación	
Código: HU6_P1	**Historias de Usuario:** 6
Nombre: Actualizar metadatos del documento.	
Descripción: Se introduce o se completa los campos en el que los metadatos extraídos no coincidan con la salida esperada.	
Condiciones de Ejecución: El usuario debe seleccionar la opción “+ Adicionar Documento¨ presente el panel izquierdo de la página de inicio de la aplicación. El usuario debe seccionar el botón “crear y regresar al listado”. El usuario debe ir a la página de inicio y seleccionar la opción “Extraer Metadatos”. El usuario debe seleccionar la opción “Verificar errores de la extracción”. El usuario debe seleccionar la opción “Editar”.	
Resultados esperados: El sistema borra todos los campos de los metadatos extraídos.	
Evolución de la prueba: Prueba satisfactoria	

Tabla 11. Caso de prueba de aceptación HU6_P1.

Casos de Pruebas de Aceptación	
Código: HU7_P1	**Historias de Usuario:** 7
Nombre: Guardar metadatos del documento.	

Descripción: Se guardan los metadatos extraídos.
Condiciones de Ejecución: El usuario debe seleccionar la opción “+ Adicionar Documento¨ presente el panel izquierdo de la página de inicio de la aplicación. El usuario debe seccionar el botón “crear y regresar al listado”. El usuario debe ir a la página de inicio y seleccionar la opción “Extraer Metadatos”. El usuario debe seleccionar la opción “Verificar errores de la extracción”. El usuario cambiara los campos que crea conveniente y dará en el botón “Guardar”.
Resultados esperados: El sistema muestra la interfaz de inicio del sistema.
Evolución de la prueba: Prueba satisfactoria

Tabla 12. Caso de prueba de aceptación HU7_P1.

Anexo 6. Aval de aceptación

AVAL DE ACEPTACIÓN

La Sociedad Cubana de Derecho e Informática declara:

Ante nosotros se ha presentado el proyecto de investigación-desarrollo “Gestión semiautomática de Metadatos en un Corpus de textos semi-estructurados o estructurados”.

El producto constituye un valioso aporte a la materialización de la línea temática: Gestión Documental y Toma de Decisiones en las organizaciones jurídicas así como al acceso y difusión del Derecho todo lo cual forma parte de los objetivos del proceso de informatización del país y en su integración con similares esfuerzo a nivel regional.

En correspondencia la Junta Directiva Nacional expresa su satisfacción respecto al resultado investigativo que satisface las necesidades del cliente y reconoce el esfuerzo de los alumnos y tutores de la Tesis en cuestión.

Para que así conste, se expide el presente Aval de Aceptación. Dado en La Habana a los 27 días del mes de junio del 2014. “Año 56 de la Revolución”.

Junta Directiva Nacional
Sociedad Cubana de Derecho e
Informática de la Unión de Juristas de Cuba