Universidad de las Ciencias Informáticas

Facultad 3

Módulo de Gestión de tipos documentales del Sistema para la obtención de documentos digitales con valor legal.

Autor: Yesenia Tabio Borrego.

Tutores

Ing. Yunior Duque Aguilar.

Msc. Yarina Amoroso Fernández.

Co-Tutor. Reinier Silverio Figueroa.

La Habana, del 2016

DECLARACIÓN DE AUTORÍA

Declaro que soy la única autora de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Autor

Yesenia Tabio Borrego

_____

Tutor

Ing. Yunior Duque Aguilar

_____

Tutor

Msc. Yarina Amoroso Fernández

_____

Co-Tutor

Reinier Silverio Figueroa.

Agradecimientos

Quiero agradecer a mi familia por su apoyo incondicional, en especial a mi mamá que ha sido todo para mí, a mi papá por sus consejos y a mi hermano querido por ser tan cariñoso conmigo. A mis primos Luly, Eli, Reini y Roger, a mis tías Lina y Amarilis. A mis amigos en general, a las del pre que no pudieron venir, a Laira, a Mireya que no pudo estar por problemas personales, a mis amigas las venenosas Yami y Thais por soportarme un beso grande las quiero, a Gilmar por rectificarme tanto en esta vida. A mi novio por ser amigo, hermano y compañero, gracias amor por aguantarme todo este tiempo. A todos los profesores del tribunal, por todas sus recomendaciones, a mis tutores por toda la dedicación que tuvieron conmigo. En fin a todos ustedes por haber venido y compartir este momento tan especial en mi vida, muchas gracias.

Dedicatoria

Dedico el presente trabajo culminado, a mi abuela Emilia que está en el cielo y sé que está orgullosa de mi. A mis abuelos que no pudieron venir los quiero mucho. A mi mamá por ser la persona más especial en mi vida. A mi hermano, a mi papá, a mi familia, a mis amigos de la universidad y de mi barrio. A mí.

RESUMEN

La gestión de objetos digitales con valor legal es un requisito indispensable dentro del Gobierno Electrónico. La conversión de los documentos impresos a objetos digitales requiere de un tratamiento especial para que preserven su valor legal, siendo esto un proceso complejo. Actualmente existen varias herramientas que digitalizan documentos con valor legal, algunas de estas desarrolladas por el Centro de Gobierno Electrónico (CEGEL) de la Universidad de las Ciencias Informáticas (UCI), sin embargo, no se adaptan a los tipos de documentos de cada fondo documental. CEGEL creó el proyecto investigativo “Sistema para la obtención de documentos digitales con valor legal” que tiene como objetivo desarrollar una solución tecnológica que permita agilizar la construcción de sistemas para la generación de objetos digitales con valor legal con la automatización de los procesos de un Centro de Digitalización. Para complementar está herramienta la presente investigación tiene como objetivo desarrollar el módulo de gestión de tipos documentales para el sistema. La solución propuesta permite al software personalizar los tipos documentales de cada fondo documental, basándose en el estudio de la tipología documental existente en Cuba. Esta fue desarrollada con las tecnologías definidas para el sistema al cual tributa, las cuales son tecnologías de código abierto.

Palabras claves: Palabras claves: digitalización, fondo, tipo, documental.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La gestión de la información a través de las Tecnologías de la Información y las Comunicaciones (TIC) ha sido de gran utilidad para entidades a todos los niveles de la sociedad, convirtiéndose en una herramienta para lograr el éxito. A nivel gubernamental las TIC se han convertido en un medio auxiliar para la gestión de información y la comunicación entre los organismos que lo conforman y la ciudadanía, de ahí el surgimiento del Gobierno Electrónico.

El Gobierno Electrónico es una nueva forma de gobierno que basa y fundamenta su aplicación en los resultados esperados, haciendo un uso eficaz de los recursos con que se cuenta, teniendo como objetivo el uso de las TIC para mejorar los servicios y la información ofrecida a los ciudadanos, a las empresas y al propio Gobierno, mejorando y simplificando los procesos de soporte institucional, y facilitando la creación de canales que permitan aumentar la transparencia y la participación ciudadana (Concha, y otros, 2014).

La gestión de objetos digitales con valor legal es un requisito indispensable dentro del Gobierno Electrónico, ya sean documentos generados por sistemas informáticos u obtenidos desde ejemplares impresos. La conversión de los documentos impresos a objetos digitales requiere de un tratamiento especial para que preserven su valor legal, lo cual hace que el proceso sea mucho más complejo que utilizar un dispositivo de digitalización de documentos, más conocidos por scanner.

El Centro de Gobierno Electrónico (CEGEL), perteneciente a la Universidad de las Ciencias Informáticas (UCI) ha desarrollado, en el marco del Convenio Intergubernamental Cuba-Venezuela, sistemas para la obtención de objetos digitales con valor legal para determinados entes gubernamentales y cuyos resultados han sido satisfactorios. Aunque estos gestionan información específica de las organizaciones para los cuales fueron creados, todos tienen implementados los requisitos imprescindibles para lograr la obtención de objetos digitales con valor legal. Es importante destacar que los sistemas actuales están desarrollados a la medida, por lo que la personalización para nuevos entornos es casi imposible, lo que trae como consecuencia que se necesitaría demasiado tiempo y recursos para obtener una solución.

Por eso, a pesar de que el CEGEL cuenta con experticia y soluciones desarrolladas para enfrentar la digitalización de cualquier otro Fondo documental, se hace necesario definir una plataforma informática que permita la personalización de entidades y gestionar la tipología documental en el ámbito nacional, cómo una primera versión de solución genérica.

Este resultado es parte del proyecto investigativo CEGEL “Sistema para la obtención de documentos digitales con valor legal“ que tiene como objetivo desarrollar una solución tecnológica que permita agilizar la construcción de sistemas para la generación de objetos digitales con valor legal con la automatización de los procesos de un Centro de Digitalización, propiciando la disminución del tiempo

de desarrollo y comercialización de los mismos, la modernización e informatización de la gestión de objetos digitales con valor legal, así como la preservación y protección de fondos documentales. La solución está dividida en módulos, los cuales se especializan en los procesos de cada área del Centro de Digitalización. Para garantizar el correcto funcionamiento del sistema es necesario organizar y procesar los documentos para que los módulos puedan interactuar, y se comuniquen los componentes de la solución, de manera automática en función del tipo documental.

Debido a lo expuesto anteriormente el **problema a resolver** queda formulado de la siguiente manera ¿Cómo gestionar los tipos documentales para la digitalización de los fondos documentales?

Por tanto, el **objeto de estudio** de este trabajo lo constituye el patrimonio documental y el **campo de acción** queda enmarcado en los procesos de gestión de la estructura de tipos documentales.

El **objetivo general** es desarrollar el módulo de gestión de tipos documentales del sistema para la obtención de documentos digitales con valor legal.

De aquí se derivan los siguientes **objetivos específicos**:

- Elaborar el marco teórico de la investigación mediante el estudio de los sistemas para la obtención de documentos digitales con valor legal.
- Desarrollar un módulo que permita la gestión de tipos documentales del sistema para la obtención de documentos digitales con valor legal.
- Verificar el correcto funcionamiento de la herramienta y que la solución propuesta dé cumplimiento al objetivo general de la investigación.

Para cumplir los objetivos trazados, se desarrollaron las siguientes **tareas de investigación**:

1. Elaboración del marco teórico del tema de investigación.
2. Estudio de las herramientas para el desarrollo de aplicaciones Desktop (escritorio).
3. Identificación de los requisitos funcionales a tener en cuenta para el desarrollo del módulo de gestión de tipos documentales del sistema para la obtención de documentos digitales con valor legal.
4. Validación de los requisitos funcionales a tener en cuenta para el desarrollo del módulo de gestión de tipos documentales del sistema para la obtención de documentos digitales con valor legal.

5. Identificación de los requisitos no funcionales a tener en cuenta para el desarrollo del módulo de gestión de tipos documentales del sistema para la obtención de documentos digitales con valor legal.

6. Diseño de los prototipos de Interfaz de Usuario (IU).

7. Validación de los prototipos de IU.

8. Realización de la especificación de requisitos de software.

9. Definición de la arquitectura a utilizar para el desarrollo del módulo de gestión de tipos documentales del sistema para la obtención de documentos digitales con valor legal.

10. Realización de los diagramas de clases.

11. Realización del modelo de diseño.

12. Realización del diseño de la base de datos.

13. Implementación del sistema de gestión.

14. Realización de las pruebas de verificar del producto.

Determinándose como **idea a defender** que:

Con el desarrollo del módulo de gestión de tipos documentales del Sistema para la obtención de documentos digitales con valor legal, se garantizará la gestión de los tipos documentales para la digitalización de los fondos documentales.

Métodos de investigación:

Métodos Teóricos:

- Histórico – Lógico: permitió la realización del estudio de la evolución de los sistemas informáticos, para la gestión de documentos que se enmarcan en la digitalización de documentos. Realizando énfasis en las características más importantes que han surgido y que son aplicables en el diseño de la solución propuesta.

- Análisis- Síntesis: este método viabilizó la realización del estudio teórico de la investigación y el análisis de la tipología documental y el proceso de digitalización, permitiendo que se sintetizaran los elementos más importantes de estos para el análisis de los documentos y la bibliografía con el objetivo de obtener información sintetizada y detallada relacionada con el objeto de estudio.

Método Empírico:

- Entrevista: este método se utilizó para esclarecer diversos puntos de vista sobre el tema de la investigación y recopilar datos concretos del cliente, con la finalidad de adquirir información necesaria que posibilitó el avance eficaz de la investigación y el logro de los objetivos propuestos. Se materializó la entrevista informal, individual y no estructurada.

Estructura capitular:

Capítulo 1 Fundamentos Teóricos: en este capítulo se abordan elementos importantes sobre las aplicaciones para la digitalización de documentos, se presentan las definiciones de digitalización de documentos, así como algunos elementos directamente relacionados con esta, fondos documentales y tipos documentales. Se realiza un estudio crítico sobre las herramientas existentes destinadas a la digitalización de documentos a nivel nacional e internacional. Además, se describe la metodología, las tecnologías, las herramientas y el lenguaje a utilizar para la construcción del módulo.

Capítulo 2 Descripción y diseño del módulo: en este capítulo se exponen los principales artefactos que se generan en la actividad de modelado realizada al módulo a desarrollar, como son: la descripción de los requisitos funcionales y no funcionales, definir la arquitectura que va a sustentar al sistema, los diagramas de clases del diseño, y el modelo de datos con el objetivo de tener una mejor comprensión, documentación del sistema y establecer las bases para la implementación del módulo.

Capítulo 3 Validación del módulo: Se describen las diferentes pruebas realizadas al sistema con el objetivo de verificar que el mismo cumple con los requerimientos planteados.

CAPÍTULO 1: Fundamentación teórica

1.1 Introducción

El dominio de algunos conceptos relacionados a los procesos documentales y procesos de digitalización de documentos, es fundamental para lograr una mayor visión sobre el tema de la investigación. También en función de estos procesos es necesario caracterizar varias herramientas en el ámbito nacional e internacional para identificar en qué y cómo pueden ser aplicadas al tipo de información y cuáles características son relevantes para la solución del problema a resolver. A partir de estos presupuestos a continuación se exponen conceptos y se identifican herramientas y metodologías que guían el proceso de la investigación acorde a los objetivos de investigación.

1.2 Definición de términos

Se abordan algunos conceptos básicos que conforman el sustento teórico de la presente investigación. De esta manera se facilitará la comprensión de los elementos que componen la propuesta del diseño de un módulo de gestión de tipos documentales del sistema para la obtención de documentos digitales con valor legal.

1.2.1 Gestión documental

La gestión documental se extiende al ciclo de vida completo de los documentos desde su producción hasta la eliminación final y su envío al archivo para su conservación permanente. Está dirigido a asegurar una documentación adecuada, evitar lo no esencial, simplificar los sistemas de creación y uso del papeleo. Además, mejora la forma de cómo se organizan y se recuperan los documentos, proporciona el cuidado adecuado y almacenamiento de los documentos en los centros de archivo y asegurar la ordenación adecuada de los documentos que no se necesitan por mucho tiempo en la conducción de los asuntos del momento (Mena, 2005).

Se entiende por consiguiente que la gestión documental permite administrar el flujo de documentos de todo tipo en una organización. Además, permite la recuperación de información desde ellos, determinando el tiempo que los documentos deben guardarse, eliminar los que ya no sirven y asegurar la conservación de los documentos más valiosos.

1.2.2 Fondo documental

La idea de un fondo se asocia a la totalidad de la documentación, con independencia de su tipo documental o soporte, producida orgánicamente y/o acumulada y utilizada por una persona física, familia o entidad en el transcurso de sus actividades y funciones como productor; y recibida por una institución o persona. Un fondo puede identificarse con un archivo. Sin embargo, pueden existir varios fondos, procedentes de instituciones diversas, depositadas en una institución de archivo (Archivo Nacional Chile, 2016).

Se entiende por consiguiente que fondo documental es un conjunto de documentos producidos por una persona natural o jurídica en desarrollo de sus funciones o actividades.

1.2.3. Tipología documental

Desde el punto de vista archivístico es el análisis y estudio de los distintos tipos o clases de documentos generados por una función o trabajo determinado de una oficina o departamento. Son documentos de la misma naturaleza, tienen el mismo modo de transmitir la información e idéntica configuración física; es decir, tienen el mismo soporte, formato y forma (Quinzaños, 2006)**.**

Un tipo documental es una unidad documental producida por un organismo en el desarrollo de una competencia concreta, regulada por una norma de procedimiento y cuyo formato, contenido informativo y soporte son homogéneos (por ejemplo, el expediente personal). Por tanto, se puede decir que son las unidades de información contenidas en los documentos.

Estas unidades documentales están clasificadas como simples o complejas:

- **Simples:** están formadas por un solo tipo documental, cuyo contenido mantiene una unidad de información. Ejemplos: el oficio, la carta, el memorando, un libro de registro, un libro de caja, recibo, cualquier otra forma de documento.
- **Complejas:** se conforma por dos o más tipos documentales que se sustentan entre sí y cuyo contenido mantiene una unidad de información. Se le conoce comúnmente como "expediente". Ejemplos: el expediente de un estudiante, trámites para licencias, cualquier otro trámite administrativo.

Por consiguiente, se entiende como tipología documental como los diferentes tipos de documentos que se pueden encontrar.

Importancia de la tipología documental

En la gestión documental el trabajo con los tipos de documentos es sumamente importante debido a que se pueden clasificar los documentos para saber cómo serán tratados y se identifica el tipo de documento que será archivado. Esto proporcionará la eficiencia de la recuperación de la información para la toma de decisiones, así como la protección de la misma, la estabilidad y continuidad administrativa, ya que se lleva a cabo un buen control sobre estos tipos. Conocer los tipos de documentos permite, además ordenar y organizar toda la documentación producida en una empresa. Esto favorecerá a un mayor desenvolvimiento en la organización permitiendo que la documentación con la que se trabaja sea más entendida por las personas que hacen uso de ella.

1.2.4. Digitalización de documentos

La digitalización de documentos se define como la captura de una imagen física, mediante escáner o cámara digital, que una vez convertida en imagen electrónica puede ser almacenada y procesada

por una computadora. Sus resultados están determinados por la resolución (densidad de puntos, o píxeles que tiene una imagen) y por la distribución luminosa en el documento, ya sea en sus niveles de grises o tonalidades de color (Gonzales Mesa, 2006).

Por consiguiente, se entiende que la digitalización de documentos es el proceso de convertir un documento a una imagen que pueda ser reconocida por un computador.

1.2.5. Documento con valor legal

Un documento de archivo puede tener diferentes valores. El valor legal es el que tiene todos los documentos que sirven de testimonio ante la ley. Sirve para documentar las obligaciones legales y proteger los derechos de las personas y las instituciones. Algunos de los documentos con valor legal son: un testamento, una tesis doctoral y el registro de miembros de una asociación (Universidad de Navarra, 2009).

Por consiguiente, se entiende por documento con valor legal como un documento que sirve para darle valor jurídico a un hecho, ya sea un acta, carta o escrito.

1.3 Sistemas existentes

En la actualidad existen herramientas que resuelven el gran problema de digitalizar los documentos, los cuales están enfocados mayormente a las necesidades de empresas e instituciones que trabajan con gran cantidad de papeles. Muchas no incluyen las funcionalidades relacionadas con digitalizar documentos con valor legal y gestionar los tipos documentales. A continuación, se expone el análisis de los sistemas homólogos.

Internacional

Abbyy Fine Reader

Abbyy Fine Reader es una aplicación de Reconocimiento Óptico de Caracteres (OCR por sus siglas en inglés) capaz de construir texto con formato a través de imágenes procedentes de un escáner o de una cámara digital. De esta forma, evita tener que volver a introducir por teclado el contenido de un texto impreso del que no se ha guardado una copia digital. El funcionamiento es sencillo: se escanea el documento, se 'lee' y se guarda en el ordenador **(ABBYY, 2015).**

El programa posee un alto nivel de precisión en el reconocimiento de caracteres y retención de formato, lo que permite disponer en unos pocos minutos de una copia electrónica de papeles, revistas, faxes, periódicos, libros y a partir de aquí poder realizar operaciones de edición y almacenamiento en diversos formatos.

Abbyy Fine Reader es una herramienta muy potente que reconoce el texto, las imágenes o las tablas que hay en un documento. Resulta además muy fácil de usar; dispone de una interfaz intuitiva y de diseño sencillo, con un editor de textos integrado, soporte para guardar imágenes, utilidad de

búsqueda y mucho más. Abbyy Fine Reader es considerado uno de los mejores de su tipo por sus funcionalidades, facilidad de uso y consumo de recursos (ABBYY, 2015).

QuickScan 4.5

QuickScan es una aplicación de procesamiento digital de imágenes de alto rendimiento para Microsoft Windows dotada de varias características, entre ellas:

- Digitalización: QuickScan usa las librerías ISIS (Especificación de imágenes e interfaz de escáner), que lo hacen compatible con más de 300 escáneres de diversos fabricantes y le permiten ajustar los valores de configuración de su escáner a través de la propia aplicación QuickScan. Los perfiles de digitalización le permiten almacenar la configuración de las tareas de digitalización más habituales con el fin de simplificar el proceso.
- Procesamiento de imágenes: El conjunto de filtros de procesamiento de imágenes de QuickScan es capaz de limpiar imágenes sucias, enderezar imágenes torcidas por causa del alineamiento incorrecto del papel durante la digitalización y eliminar los círculos negros originados por perforaciones de las imágenes digitalizadas. Los filtros se pueden usar de forma individual o secuencial y se pueden configurar para obtener un resultado óptimo. Los filtros de procesamiento de imágenes pueden aplicarse a imágenes ya existentes abiertas en QuickScan o bien de forma automática durante la digitalización de un lote.
- Sellado digital: Permite añadir un sello digital a las páginas de un lote de documentos.
- Guardar: Se almacenan las imágenes en diversos formatos de archivo estándar y esquemas de compresión (Corporation, 2006)**.**

Infoviews e-Efficiency

Infoviews e-Efficiency es una solución mexicana enfocada a resolver la administración y control de documentos físicos y digitales de todo tipo de empresas e instituciones públicas. Logra administrar el ciclo completo de vida de la información corporativa a través de sus diferentes etapas, dentro de las que se incluyen la digitalización y la automatización y/o preservación de todos sus documentos e imágenes digitales.

Permite que toda la información de las organizaciones a las que brinda servicios, que se encuentra dispersa en múltiples computadores y servidores (documentos electrónicos), pueda ser consultada mediante un sistema Web disponible desde cualquier computador perteneciente a su red, con el simple hecho de entrar por medio de un navegador y autenticarse.

Algunos de los beneficios de utilizar Infoviews e-Efficiency son:

- Reducción de horas-hombre por búsqueda de documentos.
- Reducción de costos por almacenamiento de documentos en las oficinas.
- Mejor organización de la información.

- Control de documentos físicos y electrónicos.
- Mayor eficiencia en los procesos administrativos internos debido al acceso inmediato a la información (Infoviews, 2011).

Los sistemas internacionales mencionados no satisfacen las necesidades actuales que presenta el módulo. Además, estos sistemas se ejecutan sobre el sistema operativo Windows, son software privados siendo su uso limitado por el pago de licencias y en abril de 2004 el Consejo de Ministros adoptó el Acuerdo 084/2004 donde indicaba al Ministerio de la Informática y las Comunicaciones (MIC) ordenar el proceso paulatino de migración de Cuba a Software Libre.

Nacional

DigiPyrus

El sistema de Digitalización DigiPyrus, desarrollado por CEGEL de la UCI, permite digitalizar los Tomos de los Registros y Notarías de la República Bolivariana de Venezuela, contribuyendo a agilizar el acceso a la información que a diario se tramita en cada una de sus oficinas.

Este sistema surge debido a la demanda de una gestión más eficiente y segura de los trámites que se realizan con los documentos archivados en las Oficinas de los Registros y Notarías Públicas. De ahí, que se haya propuesto presentar una solución que cumpla con el objetivo de dotar a esta entidad de un sistema que contenga las funciones, para realizar el proceso de digitalización de todos los documentos archivados y de esta forma obtener el fondo digital de cada una de estas (ALBERT, 2010).

DigiDAP

DigiDAP forma parte de la Solución Tecnológica Integral para la automatización y modernización de la División de Antecedentes Penales de la República Bolivariana de Venezuela, como el subsistema que informatiza los procesos del Centro de Digitalización para el fondo documental de dicha institución. Su objetivo fundamental es garantizar la obtención de objetos digitales con valor legal a partir de la digitalización del fondo para mejorar los servicios de inscripción y certificación de antecedentes penales a los ciudadanos de la nación venezolana (Font, 2011). Este sistema tiene como limitante que es una solución específica ante una problemática determinada, lo que limita las posibilidades de reutilización de sus componentes ante otras oportunidades de negocio con diferentes fondos documentales.

CDA

CDA es la solución tecnológica desarrollada para el Centro de Digitalización de Alfabéticas para el Servicio Administrativo de Identificación, Migración y Extranjería. Para su desarrollo se utilizó como marco de trabajo el utilizado por la solución DigiPyrus por lo que las características arquitectónicas y tecnologías de desarrollo utilizadas son similares. Su objetivo fundamental es extraer tarjetas

alfabéticas para convertirlas en metadatos. Esta solución está compuesta por varios módulos, algunos de estos son:

- Almacén Temporal: permite la gestión de inventario de los recursos que entran y salen del Centro de Digitalización, así como del control de recibo y despacho de las Tarjetas de Alfabética desde las Oficinas de Identificación hacia cada Centro de Digitalización.
- Preparación: permite la creación de los lotes de documentos en el sistema, se registran los documentos rechazados por malas condiciones para entrar en el proceso de digitalización y se realiza la corrección de los documentos rechazados por mala preparación o códigos de barra ilegible.
- Digitalización: permite la digitalización de los lotes de documentos o de documentos rechazados.
- Firma digital: permite la emisión de la firma digital de los documentos, certificando que la información obtenida es verídica (Sistema de identificación, migración y extranjería, 2011).

Todos los sistemas nacionales mencionados anteriormente, no incluyen la funcionalidad relacionada con la gestión de tipos documentales. Por lo que el presente trabajo propone la incorporación de esta nueva funcionalidad al sistema para la obtención de documentos digitales con valor legal.

En la tabla 1 se realiza una comparación de los sistemas estudiados.

Soluciones	**Licencia**	**Digitalizan documentos con valor legal**	**Gestión de tipo documental**
Abbyy FineReader	Sí	No	No
QuickScan 4.5	Sí	No	No
Infoviews e-Efficiency	Sí	No	No
DigiPyrus	Sí	Sí	No
DigiDAP	Sí	Sí	No

CDA	Sí	Sí	No

Tabla 1. Tabla Comparativa

1.3.1 Valoración crítica

Sobre el análisis de las aplicaciones para la digitalización de documentos existentes a nivel nacional e internacional se concluye, que no resultan soluciones factibles a tener en cuenta, pues estos fueron desarrollados con software propietario, lo que implica gastos muy elevados en licencias y mantenimiento. Algunos digitalizan documentos con valor legal, pero son sistemas hechos a la medida, por lo que la personalización de entidades para nuevos entornos es casi imposible, lo que tomaría demasiado tiempo y recursos. Otras de las deficiencias es que ninguno gestiona tipos documentales, por lo que se hace necesario desarrollar un módulo que dé solución al objetivo general de la investigación.

1.4 Las herramientas y tecnologías utilizadas

1.4.1 Metodología de desarrollo de software

“Las metodologías de desarrollo de software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental en el momento de desarrollar un producto de software” según (Isaías Carrillo Perez, 2008).

Existen varias metodologías de desarrollo de software que pueden ser clasificadas en ágiles y robustas, dejando al equipo de desarrollo de un producto, tomar la decisión de cuál es la que más se ajusta a sus características. Para guiar el desarrollo del módulo, se decidió emplear una metodología ágil, debido a que el mismo estará caracterizado por:

- Presentar un equipo de desarrollo muy pequeño y por ende con pocos roles.
- Énfasis en la entrega frecuente de funcionalidades desarrolladas.
- Optimizar el tiempo y evitar la demora en la generación de artefactos.
- Propiciar el trabajo conjunto del cliente con el equipo de desarrollo.

AUP – UCI

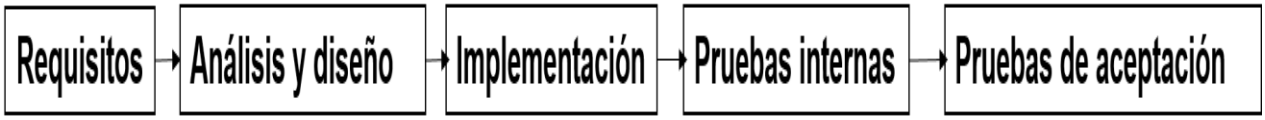
La metodología a emplear es una variación de la metodología AUP (Proceso Unificado Ágil o Agile Unified Process), de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI, en unión del modelo CMMI-DEV v1.3 (Sánchez, 2015), ya que está definida por el proyecto que desarrolla el módulo. Además, genera los artefactos que permiten obtener la documentación necesaria para una mejor comprensión de la herramienta a desarrollar.

Esta metodología cuenta de tres fases: inicio, ejecución y cierre por las que deben transitar durante el desarrollo de las actividades productivas. La presente investigación elige la fase de ejecución,

en la que se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura.

Durante el desarrollo se obtiene los requisitos, se elabora la arquitectura y el diseño, se implementa y se libera el producto.

De las siete disciplinas que propone la metodología se eligen cinco, las cuales son:



Para modelar el desarrollo de la solución, se opta por el cuarto escenario, el cual define que son para **proyectos que no modelen negocio solo pueden modelar el sistema con HU (Historia de Usuarios)**, donde se aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Además, se recomienda en proyectos no muy extensos, ya que un HU no debe poseer mucha información.

1.4.2 El Lenguaje Unificado de Modelado UML

El Lenguaje Unificado de Modelado (por sus siglas en inglés UML) es el lenguaje de modelado más conocido y utilizado en la actualidad para el desarrollo de software. UML sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas de software como para la arquitectura de hardware donde se ejecuten.

Permite visualizar, construir y documentar los elementos que forman un sistema de software orientado a objetos. Este lenguaje indica cómo escribir y leer los modelos, no dice como crearlos, pues eso es objetivo de la metodología que se utilice (Larman, 1999).

Se seleccionó UML en su versión 2.1 para modelar los artefactos seleccionados durante el proceso de desarrollo del software.

1.4.3 Herramienta de modelado

Las herramientas CASE comprenden un conjunto de programas de diferentes tipos empleados para ayudar a las actividades del proceso del software como el análisis de requisitos, el modelado de sistemas, la depuración y las pruebas (Sommerville, 2005).

1.4.3.1 Visual Paradigm para UML

Como herramienta CASE se opta por el Visual Paradigm para UML en su versión 8.0, ya que es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML (Visual Paradigm International, 2007).

1.4.4 Lenguaje de Programación

Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente.

1.4.4.1 Java

Java es un lenguaje de programación orientado a objetos ideado por SunMicrosystem. Es un lenguaje de propósito general por lo que con él se podría crear cualquier tipo de aplicación. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria (Fernández, 2005).
Como parte de las tendencias de lenguajes de programación actuales, Java presenta los siguientes beneficios:

- Es independiente de la plataforma de desarrollo.
- Existen dentro de su librería, clases gráficas que permiten crear objetos visuales comunes altamente configurables y con una arquitectura independiente de la plataforma.
- Java permite a los desarrolladores aprovechar la flexibilidad de la Programación Orientada a Objetos en el diseño de sus aplicaciones.
- El manejo de las bases de datos es uniforme, es decir, transparente y simple.
- El sistema de Java es seguro ya que tiene muchas funciones de seguridad integradas, que garantizan la seguridad del código que se está ejecutando.

Para el trabajo de diploma se decide usar Java en su versión 1.8 para la implementación del módulo, ya que es un lenguaje orientado a objetos, distribuido, seguro, multitarea y de gran rendimiento.

1.4.5 Entorno de Desarrollo Integrado (IDE)

NetBeans

Netbeans es un IDE disponible para Windows, Mac, Linux y Solaris. Permite el rápido y fácil desarrollo de aplicaciones Java de escritorio, móviles y aplicaciones web, además proporciona una gran herramienta para JavaFX, PHP, JavaScript y Ajax, Ruby y Ruby en Rails, Groovy y Grails y C/C ++ (Oracle Corporation NetBeans, 2016)**.**

A continuación, algunas ventajas de este IDE:

- Es un IDE multilenguaje y adaptable.
- Es software libre y gratuito.
- Intuitivo y fácil de utilizar.
- Se puede desarrollar todo tipo de aplicaciones.
- Es poderoso y extensible.
- Fácil integración con el servidor de aplicaciones Glassfish.

Para desarrollar el módulo propuesto se escogió NetBeans en su versión 8.0 como entorno de desarrollo integrado debido a las características mencionadas anteriormente.

1.4.6 Sistema de Gestor de Base Datos SGBD – PostgreSQL

Como SGBD se opta por PostgreSQL, ya que es orientado a objetos de software libre, publicado bajo la licencia BSD1. Según (Stinson, 2001) permite que mientras un proceso escriba en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último que se ejecutó. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

Utiliza un modelo cliente-servidor y multiproceso en vez de multihilo para garantizar la estabilidad del sistema. La aplicación de escritorio pgAdmin III se utiliza como interfaz gráfica para hacer más fácil la administración del usuario con la base de datos, además se puede ejecutar en varias plataformas (PostgreSQL, 2013).

1.4.7 Mapeo Objeto-Relacional

En la actualidad, casi todas las aplicaciones están diseñadas para usar la Programación Orientada a Objetos (POO), y las bases de datos más utilizadas son del tipo relacional, es decir, que solo permiten guardar tipos de datos primitivos. Este problema trae consigo que no se pueda guardar directamente en las tablas los objetos que genera la aplicación, por lo que se hace necesario realizar el Mapeo de Objeto-Relacional (ORM), ya que es el encargado de realizar conversiones y simula que la base de datos es orientada a objeto.

1 Berkeley Software Distribution (Distribución de Software Berkeley)

ORM: es una técnica de programación para convertir datos entre el lenguaje de programación orientado a objetos utilizado y el sistema de base de datos relacional utilizado en el desarrollo de una aplicación (Programación en castellano, 2016).

Para el mapeo de la base de datos se escoge Eclipse Link, ya que es un ORM de código abierto y el marco de persistencia de objetos está apoyado en un entorno de Java. Soporta varias fuentes y formatos de datos incluyendo las bases de datos relacionales, no relacionales, los servicios XML (Lenguaje de Marcas Extensible), JSON[2] y se puede utilizar en cualquier plataforma Java.

1.4.8 XEGFORT

XEGFORT es un marco de trabajo que facilita la reutilización de componentes en sistemas de corte empresarial. Es fácil de utilizar y configurar. Algunos de los elementos fundamentales con los que cuenta este marco de trabajo son la seguridad, la configurabilidad y la adaptabilidad, convirtiéndolo en un software altamente reusable, además garantiza la confiabilidad e integridad de la información que se intercambia a través de la red entre los clientes y el servidor de aplicaciones, facilitando la autenticación de los clientes.

Entre las principales funcionalidades con las que cuenta XEGFORT es la fortaleza de las contraseñas de las cuentas de los usuarios, contando con una encriptación de las mismas, evitando el robo de estas, que implicaría comprometer la seguridad del sistema. También se lleva un control del tiempo de inactividad máximo permitido sin que un operador interactúe con la aplicación reduciendo el riesgo de accesos que no estén autorizados a interactuar con el sistema.

Entre otras funcionalidades con las que cuenta son: la gestión de las sesiones de los usuarios, la gestión de usuarios, el control de trazas y excepciones y otras funcionalidades de administración del sistema.

Está diseñado para la creación de aplicaciones con arquitectura Cliente-Servidor utilizando la plataforma JEE (Java Platform, Enterprise Edition). Por esta razón XEGFORT consta de dos partes, una que se encuentra en el lado del cliente facilitando la creación de la capa de presentación y la comunicación con los objetos remotos desplegados en el servidor y una segunda que se encuentra en el lado del servidor donde se encuentra la lógica de negocio (Lenis Matos Rodriguez, 2012).

Para el desarrollo del módulo se opta por el marco de trabajo XEGFORT en su versión 2.0, ya que es el marco de trabajo definido por el proyecto.

1.5 Arquitectura de software

La arquitectura de software alude a la estructura general del software y las formas en que la estructura proporciona una integridad conceptual para un sistema. En su forma más simple, la

2 JSON (JavaScript Object Notation): formato de texto ligero para el intercambio de datos.

arquitectura es la estructura u organización de los componentes del programa (módulos), la manera en que estos componentes interactúan, y la estructura de datos que utilizan los componentes. En sentido más amplio, sin embargo, los componentes pueden generalizarse para representar elementos importantes del sistema y sus interacciones (Pressman, 2002).

A continuación, se fundamentarán los diferentes estilos arquitectónicos presentes en la propuesta de solución:

1.5.1 Arquitectura Cliente-Servidor

Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (Servidor servidor) que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras. La interacción cliente-servidor es el soporte de la mayor parte de la comunicación por redes. Ayuda a comprender las bases sobre las que están construidos los algoritmos distribuidos (Ver Figura 1).

Según Somerville, cuando un sistema se organiza como un conjunto de servicios y servidores asociados, más clientes que acceden y usan los servicios, se está en presencia del modelo arquitectónico cliente-servidor, el cual está compuesto por los siguientes componentes (Sommerville, 2005):

- Un conjunto de servidores que ofrecen servicios a otros subsistemas.
- Un conjunto de clientes que llaman a los servicios ofrecidos por los servidores.
- Una red que permite a los clientes acceder a estos servicios. Esto no es estrictamente necesario, ya que los clientes y los servidores podrían ejecutarse sobre una misma máquina.

Figura 1. Modelo Cliente/Servidor.

1.5.2 Arquitectura en Capas

Los autores Garlan y Shaw (David Garlan, 1994) definen el estilo en capas como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente inferior y se sirve de las prestaciones que le brinda la inmediatamente superior.

Los principales beneficios del estilo de arquitectura basado en capas son (Microsoft | Achitecture, 2010):

- Abstracción: las capas permiten cambios que se realicen en un nivel abstracto.
- Aislamiento: la arquitectura de capas permite aislarlos cambios en tecnologías a ciertas capas para reducir el impacto en el sistema total.
- Rendimiento: distribuir las capas entre múltiples sistemas (físicos) puede incrementar la escalabilidad, la tolerancia a fallos y el rendimiento.
- Mejoras en pruebas: la capacidad de realizar pruebas se beneficia de tener interfaces bien definidas para cada capa, así como de la habilidad para cambiar a diferentes implementaciones de las interfaces de cada capa.

1.6 Patrones de diseño

Los patrones de diseño son herramientas que proveen facilidades para crear un software reutilizable de buena calidad. Cada patrón describe un problema que ocurre repetidamente en nuestro entorno, y describe el núcleo de la solución a ese problema, de tal forma que ésta pueda ser usada un millón de veces, sin hacer el mismo trabajo dos veces (Asenjo González, 2003).

Algunos de los patrones utilizados en la solución que se brinda en este trabajo de diploma para el diseño del módulo son:

Patrones DAO

El patrón Objeto de Acceso a Datos (DAO por sus siglas en inglés) suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos. Consiste básicamente en una clase que es la que interactúa con la base de datos. En una aplicación, hay tantos DAOs como tablas en la base de datos. Los métodos de esta clase dependen de la aplicación y de lo que queramos hacer (Richardmx, 2008).

Patrones DTO

Los DTO (Data Transfer Object) o también denominados VO (Value Object) son utilizados por DAO para transportar los datos desde la base de datos hacia la capa de lógica de negocio y viceversa (Richardmx, 2008).

Patrones GRASP

Los Patrones Generales de Software para la Asignación de Responsabilidades (GRASP por sus siglas en inglés) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (Pressman, 2002).

- **Experto:** el patrón experto en información se utiliza con frecuencia en la asignación de responsabilidades; es un principio de guía básico que se utiliza continuamente en el diseño de objetos. Este permite conservar el encapsulamiento, ya que los objetos se valen de su propia información para llevar a cabo las tareas. El comportamiento se distribuye entre clases que cuentan con la información requerida, alentando con ello definiciones de clase sencillas y más cohesivas que son más fáciles de comprender y mantener. Así se brinda soporte a una alta cohesión (Buono, 2012).
- **Creador:** el patrón guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización (Larman, 1999) y (Visconti, y otros, 2004).
- **Controlador:** el patrón controlador es un objeto que no pertenece a la interfaz de usuario, es un manejador artificial de todos los eventos del sistema que define el método de su operación. A menudo genera un bajo acoplamiento y permite un mayor potencial de los componentes reutilizables (Visconti, y otros, 2004). En (Larman, 1999) se plantea que los objetos externos de conexión y la capa de presentación no deberían tener la responsabilidad de llevar a cabo los eventos del sistema.
- **Bajo acoplamiento:** consiste en tener las clases lo menos relacionadas entre sí, para que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión en el resto de las clases. Esta característica permite potenciar la reutilización y disminuye la dependencia entre las clases (USM.cl, 2015).
- **Alta cohesión:** en la perspectiva del diseño orientado a objetos, la cohesión (o, más exactamente, la cohesión funcional) es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme (Larman, 1999) y (Visconti, y otros, 2004).

Patrones GoF

Se encargan de solucionar problemas de composición de clases y objetos, creación de instancias, integración y responsabilidades entre clases y objetos, así como los algoritmos que encapsulan para el diseño básico del sistema.

- **Fachada:** es la clase definida que ofrece una interfaz común con un conjunto heterogéneo de interfaces. Las interfaces heterogéneas pueden ser un conjunto de funciones, un esquema, un grupo de otras clases o un subsistema (Pressman, 2002).
- **Instancia única (Singleton):** Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. En la propuesta se evidencia en todas las clases controladoras que son instancias únicas (Pressman, 2002).

1.7 Técnicas de validación de requisitos

Con el objetivo de demostrar que los requerimientos previamente definidos cumplen con las expectativas del cliente, se aplicaron las técnicas de validación de requisitos:

- **Revisión de requisitos:** se realizan reuniones para localizar errores en el documento. Donde se agregaron requisitos y se modificaron otros.
- **Generación de casos de prueba de aceptación*:*** para validar los requisitos funcionales de la solución, se diseñan casos de pruebas de aceptación para cada una de las Historias de Usuario.
- **Prototipos:** algunas propuestas se basan en obtener de la definición de requisitos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema con el usuario. Esta técnica tiene el problema de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final (Pressman, 2010).

1.8 Métrica de requisitos

Las métricas son un buen medio para entender, monitorizar, controlar, predecir y probar el desarrollo de software y los proyectos de mantenimiento (Briand, 1996) y pueden ser utilizadas por profesionales e investigadores para tomar mejores decisiones (Pfleeger, 1997). A continuación, se presenta la métrica a tener en cuenta para la validación de los requisitos del módulo.

Métrica de estabilidad

El objetivo de esta métrica es medir la estabilidad de los requerimientos para asegurar su adecuación antes de pasar al próximo flujo de trabajo. Se considera que los requerimientos son estables cuando no existen adiciones o supresiones en ellos que impliquen modificaciones en las funcionalidades principales de la aplicación. La estabilidad de los requerimientos se calcula como (Alcantara Rabí, y otros, 2010):

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

- ERT: valor de la estabilidad de los requerimientos.
- RT: total de los requerimientos definidos.

- RM: número de requerimientos modificados, que se obtienen como la sumatoria de los requerimientos insertados, modificados y eliminados.

Esta métrica ofrece valores entre 0 y 100. El mejor valor de ETR es el más cercano a 100 ya que mostrará que no se están realizando cambios sobre los requisitos, son estables y por tanto es confiable trabajar el diseño sobre ellos.

1.9 Verificación del diseño

Para evaluar la calidad del diseño se opta por el uso de las métricas orientada a clases: Tamaño operacional de clases (TOC) y Relaciones entre clases (RC), y la métrica de Árbol de Profundidad de Herencia (APH), para medir la profundidad de herencia del módulo.

TOC: está dada por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad (Pressman, 2002):

- **Responsabilidad:** un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
- **Complejidad de implementación:** un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
- **Reutilización:** un aumento del TOC implica una disminución del grado de reutilización de la clase.

A continuación, se muestra en la siguiente tabla la aplicación de la métrica TOC, donde CM se refiere a la cantidad de métodos:

Atributo	Categoría	Criterio
Responsabilidad	Baja	CM < =Promedio
	Media	Promedio<= CM <=2* Promedio
	Alta	CM > 2* Promedio
Complejidad de implementación	Baja	CM <=Promedio
	Media	Promedio<= CM <=2* Promedio
	Alta	CM > 2* Promedio
Reutilización	Baja	CM > 2* Promedio
	Media	Promedio<= CM <=2* Promedio
	Alta	CM <=Promedio

Tabla 2. Rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC.

Pasos que se llevaron a cabo para aplicar la métrica:

1. Cálculo de la cantidad de procedimientos. La cantidad de métodos se toma del tamaño general de una clase que se determina sumando todas las operaciones que posee.
2. Calcular el promedio de los métodos.
3. Teniendo en cuenta los valores antes obtenidos se determina la incidencia de los atributos de calidad en cada una de las clases, según los criterios expuestos en la tabla 2.

RC: está dado por el número de relaciones de uso de una clase con otra, o sea el número de dependencias que una clase tiene con otra y evalúa los siguientes atributos de calidad (Pressman, 2002):

- **Acoplamiento:** un aumento de las RC implica un aumento del Acoplamiento de la clase.
- **Complejidad de mantenimiento:** un aumento de las RC implica un aumento de la complejidad del mantenimiento de la clase.
- **Reutilización:** un aumento de las RC implica una disminución en el grado de reutilización de la clase.
- **Cantidad de pruebas:** un aumento de las RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

A continuación, se muestra en la siguiente tabla la aplicación de la métrica RC, donde CRU se refiere a la cantidad de relaciones de uso.

Atributo	**Categoría**	**Criterio**
Acoplamiento	Ninguno	CRU = 0
	Bajo	CRU = 1
	Medio	CRU = 2
	Alto	CRU >2
Complejidad del mantenimiento	Baja	CRU <=Promedio
	Media	Promedio<=CRU<=2* Promedio
	Alta	CRU > 2* Promedio
Cantidad de pruebas	Baja	CRU <=Promedio
	Media	Promedio<=CRU<=2* Promedio
	Alta	CRU > 2* Promedio
Atributo	**Categoría**	**Criterio**

Tabla 3. Rangos de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica RC.

Pasos que se llevaron a cabo para aplicar la métrica:

1. Determinar la cantidad de relaciones de uso (CRU) que poseen las clases a medir.
2. Calcular el promedio de las CRU.
3. Teniendo en cuenta los valores antes obtenidos se determina la incidencia de los atributos de calidad en cada una de las clases, según los criterios expuestos en la tabla 3.

APH: mide el máximo nivel en la jerarquía de herencia de clases; la raíz del árbol de herencia no hereda de ninguna clase y está en el nivel cero del árbol de herencia. De acuerdo a Chidamber y Kemerer esta métrica se hizo con el propósito de medir la complejidad de la clase, la complejidad del diseño y el potencial de reutilización, ya que mientras más profunda es la clase, mayor será el número de métodos de clase que debe heredar. Lorenz y Kidd surgieren un umbral de 6 niveles para las clases individuales para indicar herencia excesiva para proyectos en C++. En lenguajes como Java donde los objetos siempre heredan de la clase Object, se agrega un nivel (Harrison, 2001).

Pasos para aplicar la métrica:

1. Hallar la APH individual de cada clase.
2. Calcular el promedio del APH.
3. Comprobar si el promedio del APH < 7 (sería el nivel del umbral); en caso de ser así, se puede concluir que hay una buena reutilización de herencia.

1.10 Conclusiones del capítulo

El estudio de los conceptos más esenciales asociados a la investigación permitió un mejor entendimiento sobre el desarrollo del mismo y comprender el negocio para modelarlo. El análisis de los sistemas homólogos, referente a la digitalización de documentos, permitió identificar la necesidad de desarrollar un módulo que permita gestionar los tipos documentales del Sistema para la obtención de documentos digitales con valor legal. El análisis de la metodología de desarrollo AUP-UCI, brindó elementos significativos para la implementación del módulo, tales como los artefactos que son necesarios para el desarrollo de la solución. Se utilizarán las herramientas y tecnologías que fueron asignadas por el proyecto, para el desarrollo del mismo.

CAPÍTULO 2: Características y diseño del módulo

2.1 Introducción

En este capítulo se realiza una propuesta de solución del módulo para darle cumplimiento al objetivo general, teniendo en cuenta las fases que presenta la metodología ágil de desarrollo de software AUP-UCI. Se describen los principales artefactos generados en dicha actividad, con el objetivo de lograr una mayor documentación y comprensión del módulo a desarrollar, facilitando el proceso de implementación de la misma. Además, se definen los patrones de diseño, la arquitectura y el modelo de datos.

2.2 Especificación de los requisitos de software

La especificación de los requisitos de software fue una tarea realizada por los analistas del proyecto en la fase de levantamiento de requisitos.

Se corresponde con la descripción completa del comportamiento del sistema que se va a desarrollar, ayuda a los ingenieros de software a entender mejor el problema. Incluye un conjunto de tareas que ayudan a comprender cuál será el impacto del software sobre el negocio, qué es lo que quiere el cliente y como interactuarán los usuarios finales con el software (Pressman, 2002).

2.2.1 Requisitos funcionales (RF)

En el contexto de esta investigación, se entiende por requisitos funcionales a las condiciones o capacidades que el sistema debe cumplir. A continuación, se enumeran los requisitos funcionales identificados en el desarrollo de la solución:

RF1 – Adicionar fondo documental.

RF2– Modificar fondo documental.

RF3 – Buscar fondo documental.

RF4 – Eliminar fondo documental.

RF5 – Adicionar tipo documental.

RF6– Modificar tipo documental.

RF7– Listar tipo documental.

RF8– Eliminar tipo documental.

RF9 – Crear atributo.

RF10 – Modificar atributo.

RF11 – Eliminar atributo.

RF12– Buscar atributo.

RF13 – Listar tipo atributo.

2.2.1.1 Resultado de la métrica de requisitos

Los requisitos del software son la base de las medidas de calidad. En la disciplina de requisitos se tuvo en cuenta la métrica para medir su estabilidad.

Aplicación de la métrica Estabilidad de requisitos

Teniendo en cuenta que se identificaron un total de 13 requisitos funcionales, de los cuales tres resultaron modificados (tres por eliminación) se calcula:

ETR = [(13 - 3) / 13 * 100]= 76.92

Como resultado se obtuvo un valor de 76,92. Dicha cifra demuestra que no se han realizado cambios significativos sobre los requisitos, son estables y, por tanto, es confiable el análisis y diseño sobre ellos.

2.2.4 Requisitos no funcionales (RNF)

Los requisitos no funcionales definen propiedades y restricciones del sistema, estos requisitos requieren de mayor atención en algunas ocasiones que los requisitos funcionales ya que son normalmente a los que debe apuntar la arquitectura y si estos no son cumplidos, el software puede no funcionar o el cliente simplemente no aceptar el producto.

RNF1. Usabilidad

El sistema debe ser usado por personas que tengan conocimientos básicos de informática.

RNF2. Portabilidad

Las herramientas podrán ser usadas bajo cualquier sistema operativo de Windows NT en adelante o cualquier distribución de Linux.

RNF3. Apariencia o Interfaz Externa

El sistema debe ofrecer una interfaz amigable y fácil de operar. Se debe mantener la línea de diseño establecida, con la uniformidad y representatividad de la solución. Todas las interfaces deben estar en idioma español.

RNF4. Software

La herramienta deberá funcionar sobre el sistema operativo GNU/Linux, en su versión 11.04 o superior. Para la base de datos se requiere tener instalado el PostgreSQL en su versión 9.4, JDK en su versión 1.8.6.9, Glassfish Server en su versión 4.1 y el NetBeans como IDE de desarrollo en su versión 8.1, para el funcionamiento de la aplicación.

2.3 Historias de Usuario (HU)

Las HU son utilizadas como herramienta para dar a conocer los requerimientos del sistema al equipo de desarrollo. Son pequeños textos en los que el cliente describe una actividad que realizará el sistema; la redacción de los mismos se realiza bajo la terminología del cliente, no del desarrollador, de forma que sea clara y sencilla, sin profundizar en detalles. También son utilizadas para estimar el tiempo que el equipo de desarrollo tomará para realizar las entregas. En una entrega se puede desarrollar una o varias historias de usuario, esto depende del tiempo que demore la implementación de cada una de las mismas (Luis Calabria, 2003).

De los requisitos funcionales que se especificaron se realizaron sus historias de usuarios. A continuación, se muestra un ejemplo de la historia de usuario del requisito: *Adicionar tipo documental,* el resto de ellas están en el expediente del proyecto.

Número: 5	**Nombre del requisito:** Adicionar tipo documental.
Programador: Yesenia Tabio Borrego.	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 4h
Riesgo en Desarrollo: Alto	**Tiempo Real:** 5 días
Descripción: Permite al sistema adicionar los diferentes tipos documentales. El sistema muestra una interfaz para que el usuario registre un tipo documental, en la interfaz se muestran los campos para el nombre del tipo documental, cantidad de página que tiene el tipo documental, una lista de los fondos documentales y los atributos que pueden asociarse a este. ✓ Nombre tipo documental: campo tipo texto, N caracteres alfanumérico. ✓ Cantidad de página: campo de tipo entero. ✓ Fondo documental: campo de tipo Lista. ✓ Atributos: campo de tipo Lista.	
Observaciones: La historia de usuario depende de las HU 1 y 9. El usuario no puede seleccionar el botón aceptar sin a ver llenado todos los campos, ya que todos los campos son obligatorios.	
Prototipo elemental de interfaz gráfica de usuario:	

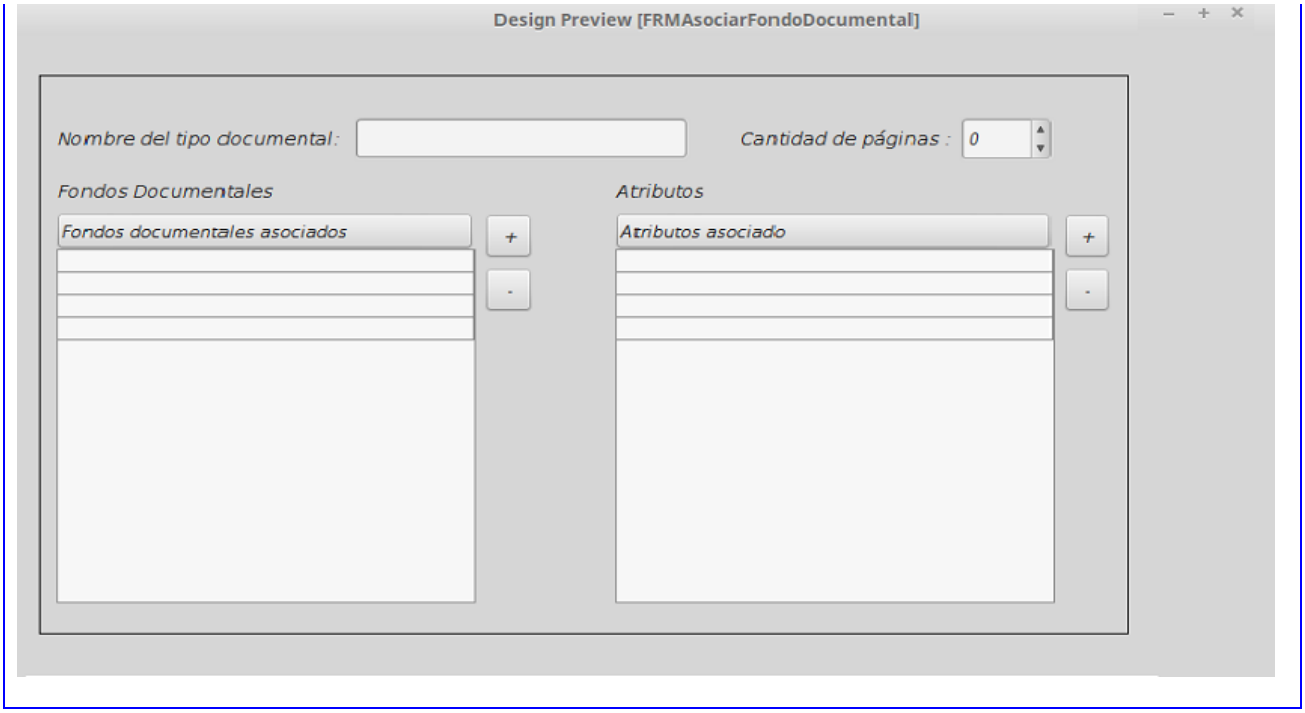


Tabla 4. Historia de Usuario *Adicionar tipo documental.*

2.3.1 Planificación de la entrega

El objetivo de esta fase es el de llegar a un acuerdo entre los clientes y los programadores respecto a cuáles serán las HU a ser implementadas durante cada iteración y establecer cuál va a ser el contenido de la primera entrega. Los desarrolladores estiman cuanto tiempo y esfuerzo requiere cada HU y se determina el cronograma. La duración del calendario para la primera entrega no suele superar los dos meses (Calabria, y otros, 2003).

No	Historias de usuario	Estimación (semana)
1	Adicionar fondo documental.	1
2	Modificar fondo documental.	½
3	Buscar fondo documental.	½
4	Eliminar fondo documental.	½
5	Adicionar tipo documental.	1
6	Modificar tipo documental.	½
7	Listar tipo documental.	½
8	Eliminar tipo documental.	½
9	Crear atributo.	½
10	Modificar atributo.	½

11	Buscar atributo.	½
12	Eliminar atributo.	½
13	Listar tipo atributo.	½

Tabla 5. Plan de duración de las iteraciones.

2.3.2 Iteraciones

Se realiza un calendario dividido en un número iteraciones, de tal manera que cada iteración tome de una a tres semanas de implementación. En la primera iteración se crea un sistema que abarca los aspectos más importantes de la arquitectura global, esto se logra seleccionando las HU que hagan referencia a la construcción de la estructura de todo el sistema. El cliente decide que HU van a ser implementadas para cada iteración.

Plan de iteraciones

En el plan de iteraciones se especifican las HU a implementar en cada iteración del módulo, estableciéndose tres iteraciones para la realización del módulo en coordinación con el cliente:

- Iteración 1: tiene como objetivo la implementación de las HU: 1, 5, y 9.
- Iteración 2: tiene como objetivo la implementación de las HU: 2, 3, 6, 7, 10, 11 y 13.
- Iteración 3: tiene como objetivo la implementación de las HU: 4, 8 y 12.

Plan de entregas

No	Historia de usuario	Estimación (semana)	Fecha (inicio-fin)
1	Adicionar fondo documental. Adicionar tipo documental. Crear atributo.	Dos y media	15/05/2016-31/05/2016
2	Modificar fondo documental. Buscar fondo documental. Modificar tipo documental. Listar tipo documental. Modificar atributo. Buscar atributo. Listar tipo atributo.	Tres y media	31/05/2016-24/06/2016

3	Eliminar fondo documental. Eliminar tipo documental. Eliminar atributo.	Una y media	24/06/2016-4/07/2016

Tabla 6. Plan de entregas.

2.4 Análisis y diseño del módulo

2.4.1 Arquitectura del módulo

Atendiendo al análisis realizado en la fundamentación teórica de la investigación (capítulo1), se definió para el módulo una combinación del estilo Cliente/Servidor, distribuida en n capas (Ver Figura 2).

Haciendo uso del estilo cliente/servidor existe un cliente que presenta llamadas a un servidor de aplicaciones mediante el protocolo RMI-IIOP y este le retorna el resultado de la llamada para que se encargue de la presentación de los datos en una interfaz gráfica de usuario.

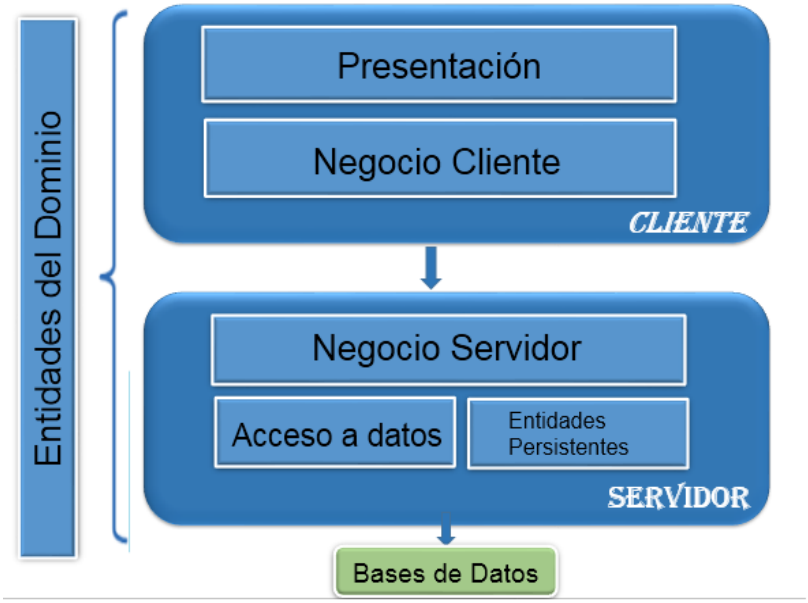


Figura 2. Distribución de las capas de la arquitectura.

Estas capas son Presentación, Negocio Cliente, Negocio Servidor y Acceso a Datos.

La **capa de presentación:** es la que interactúa con el usuario, mostrando los datos, que obtiene en la capa negocio cliente. Además, controla los eventos que se generan en la misma. Solo se relaciona con la capa negocio cliente, ya que cada capa solo se comunica con otra vecina a ella.

La **capa de negocio cliente:** gestiona la lógica de negocio relacionada en la parte del cliente, y el acceso a las funcionalidades de los componentes del servidor. La lógica de negocio está

implementada sobre un concepto que agrega a la arquitectura, que es el concepto de gestor de negocio del cliente, que no es más que la agrupación de funcionalidades que se refiere a una misma entidad en un gestor con su nombre.

La **capa de negocio servidor:** es responsable de presentar una fachada a todos los componentes que se encuentran en el servidor a modo de gestores de negocio, físicamente alojada en un servidor de aplicaciones al cual se accede de forma remota. Los gestores de negocio servidor presenta una interfaz y su implementación contiene la lógica de negocio de las funcionalidades. Además, esta capa hace llamada a la capa de acceso a datos y se lo retorna a la capa de negocio cliente.

La **capa de acceso a datos:** recibe los datos de la capa de negocio servidor y los puede consultar, persistir, actualizar y eliminar en la base de datos, mediante mecanismos que ofrecen las especificaciones utilizadas, que se realizan en conjunto con las entidades persistentes.

Las **entidades persistentes:** representan una relación entre la base de datos y los objetos de la programación, donde cada entidad se corresponde con una tabla.

La **base de datos:** se encuentra ubicada físicamente en el SGBD, en ella se encuentran los datos almacenados listos para ser consultados y actualizados, también se encuentran los mecanismos para facilitar el trabajo con los datos como las funciones y secuencias.

Las **entidades del dominio:** está implícita a través de todas las capas de la arquitectura.

A continuación, se muestra la estructura por paquetes de la HU *Adicionar Tipo Documental*, siguiendo la arquitectura empleada:

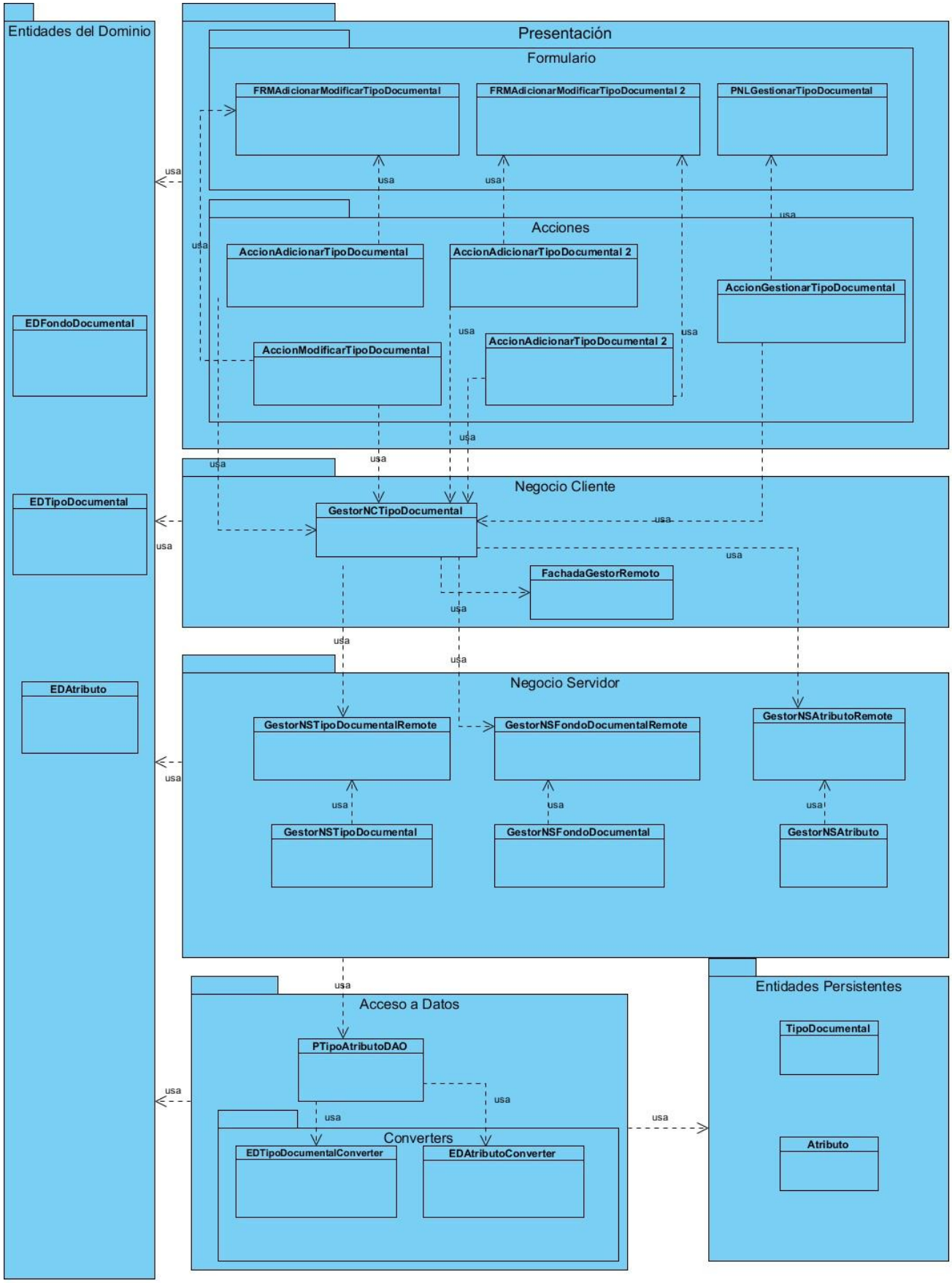


Figura 3. Distribución de paquete de la HU Adicionar tipo documental.

2.4.2 Patrones del diseño

Para diseñar el módulo se emplearon un conjunto de patrones de diseño, los cuales son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software. A continuación, se describen los patrones empleados:

Patrones DAO

Consiste básicamente en las clases que interactúan con la base de datos. El uso de este patrón se evidencia en la clase PTipoDocumentalDAO.

Patrones DTO

Son utilizados por las clases DAO, para transportar los datos desde la base de datos hasta los gestores de negocio del servidor y viceversa. Este patrón se evidencia en la clase EDTipoDocumental.

Patrones de diseño GRASP

- **Experto:** otorga la responsabilidad de adicionar un tipo documental a la clase GestorMarcoTrabajo, que es la experta de instanciar y mostrar los formularios y las acciones en el área de trabajo. Se representa en el anexo.
- **Controlador:** la clase AccionTipoDocumental le asigna la responsabilidad de controlar y coordinar el esfuerzo de las demás clases, para responder a la petición del usuario. Se representa en el anexo.
- **Creador:** asigna la responsabilidad de crear un objeto de la clase EDTipoDocumentalConverter a la clase PTipoDocumentalDAO, para transportar datos desde el cliente al servidor. Se representa en el anexo.
- **Alta Cohesión:** la principal característica de este patrón es asignar responsabilidades de modo que la cohesión siga siendo alta. La información que almacena una clase debe de ser coherente y debe estar en la medida de lo posible relacionada con la clase. El patrón se evidencia en cada de una de las clases del módulo, de tal forma que se elimina la sobrecarga de responsabilidades.
- **Bajo Acoplamiento:** su principal característica es mantener las clases más independientes entre sí y con la menor cantidad de relaciones; la cual posibilita que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización y disminuyendo la dependencia entre las clases. El patrón se evidencia en cada una de las clases diseñadas.

Patrones GoF

- **Fachada:** se evidencia en la clase FachadaGestoresRemotos, que brinda el método getGestorRemoto() que unifica y facilita el trabajo necesario para obtener un componente del servidor. Se representa en el anexo.

- **Instancia única (Singleton):** presenta un mecanismo para limitar el número de instancias de una clase. Su uso se aprecia en la clase GestorMarcoTrabajo, proporcionando una instancia única a cada instancia del cliente que se ejecuta. Se representa en el anexo.

2.4.3 Estándares de codificación

El estándar de codificación empleado en el desarrollo del módulo fue definido por el equipo de desarrollo. A continuación, se muestran algunas pautas de dicho estándar.

- Todas las nomenclaturas a utilizar se definirán en idioma español.
- Los nombres de los paquetes y las clases serán con mayúscula, en caso de ser un nombre compuesto las siguientes palabras se escribirán de igual forma.
- Los nombres de los métodos serán con minúscula, en caso de ser un nombre compuesto las siguientes palabras se escribirán con mayúscula.
- Las clases gestoras de negocio comienza con el prefijo Gestor y luego el nombre de la clase (GestorNSTipoDocumental.java).
- Las clases de acceso a datos comienzan con el prefijo P, el nombre de la clase y luego con el prefijo DAO (PTipoAtributoDAO.java).

2.4.4 Diseño de la Base de Datos

La estructura de la base de datos del módulo de Gestión de tipos documentales se muestra mediante el modelo de datos. Este modelo fue elaborado utilizando la técnica de modelado de datos Entidad-Relación, mostrando las entidades de datos, sus atributos asociados y las relaciones entre las entidades.

La base de datos está compuesta por las tablas fondo_documental, tipo_documental, atributo_tipo_documental, atributo y tipo_atributo. La tabla fondo_documental almacena los nombres de todos los fondos documentales. La tabla tipo_documental guarda los tipos documentales asociados a un fondo documental. La tabla atributo_tipo_documental almacena el valor de la relación que hay entre un tipo documental y el atributo asociado a ese tipo documental. La tabla atributo almacena todos los atributos que tiene un tipo documental. La tabla tipo_atributo almacena los diferentes tipos de atributos (integer, character, boolean).

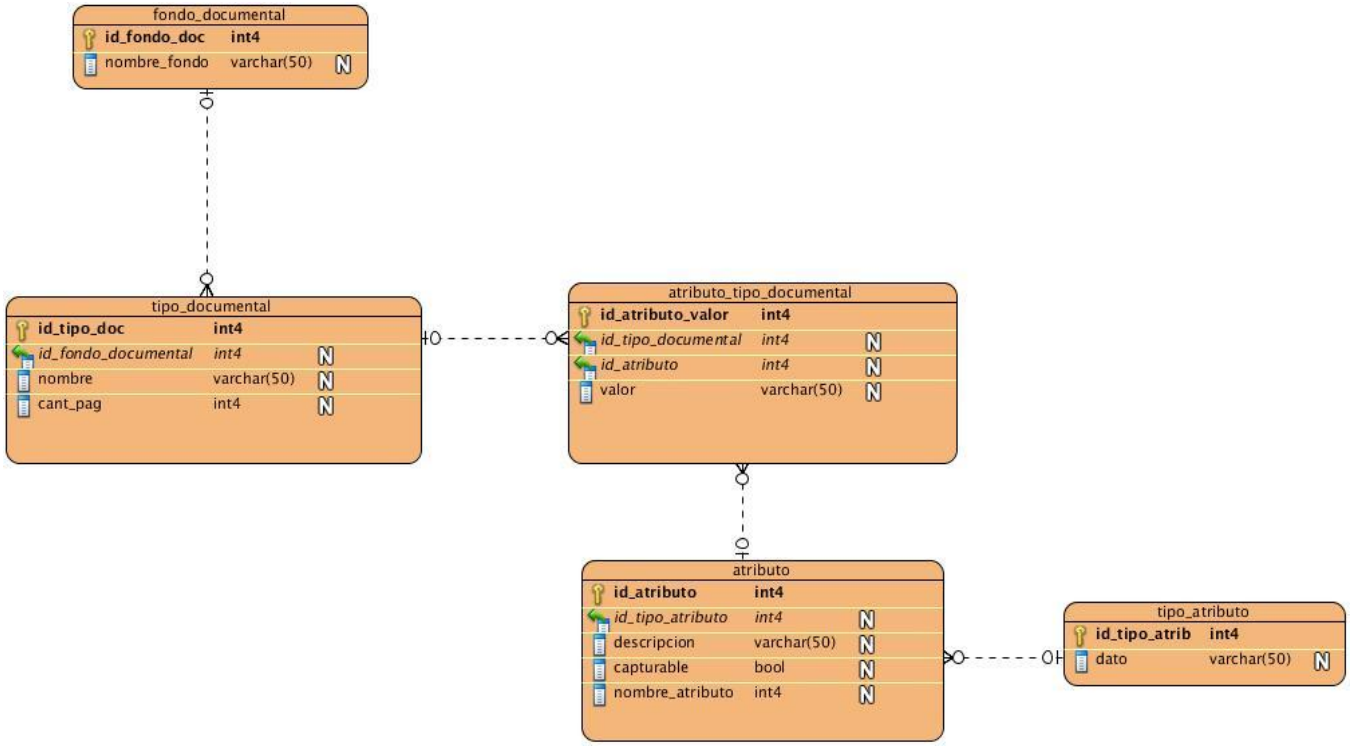


Figura 4. Modelo de datos.

2.5 Verificación del diseño

Para la evaluación de la calidad del diseño propuesto se hace uso de la métrica Tamaño operacional de clase (TOC) propuestas por Lorenz y Kidd, la métrica Árbol de Profundidad de Herencia (APH) y la métrica Acoplamiento entre clases (CBO) propuesta por Chidamber y Kemerer.

2.5.1 Tamaño Operacional de clases TOC

TOC: permite medir los atributos de calidad responsabilidad, complejidad de implementación y reutilización de las clases del diseño. La responsabilidad y la complejidad son inversamente proporcionales a la reutilización, por lo que, a mayor responsabilidad y complejidad de implementación de una clase, menor será el nivel de reutilización. A continuación, se muestran las clases del módulo evaluadas en los atributos de calidad mencionados.

Clase	Cantidad de Métodos	Responsabilidad	Complejidad	Reutilización
AccionAdicionarAtributo	3	Baja	Baja	Alta

AccionGestionarAtributo	2	Baja	Baja	Alta
AccionModificarAtributo	2	Baja	Baja	Alta
AccionAdicionarFondoDocumental	6	Media	Media	Media
AccionGestionarFondoDocumental	2	Baja	Baja	Alta
AccionModificarFondoDocumental	2	Baja	Baja	Alta
AccionAdicionarTipoDocumental	2	Baja	Baja	Alta
AccionAsociarAtributoTipoDocumental	2	Baja	Baja	Alta
AccionAsociarFondoDocumentalTipoDocumental l	2	Baja	Baja	Alta
AccionGestionarTipoDocumental	2	Baja	Baja	Alta
AccionModificarTipoDocumental	2	Baja	Baja	Alta
GestorNCAtributo	4	Media	Media	Media
GestorNCFondoDocumental	5	Media	Media	Media
GestorNCTipoDocumental	2	Baja	Baja	Alta
EDAtributoConverter	4	Media	Media	Media
EDFondoDocumentalConverter	4	Media	Media	Media
EDTipoDocumentalConverter	4	Media	Media	Media
EDTipoAtributolConverter	4	Media	Media	Media
PAtributoDAO	4	Media	Media	Media
PFondoDocumentalDAO	4	Media	Media	Media
PTipoAtributoDAO	1	Baja	Baja	Alta
PTipoDocumentalDAO	4	Media	Media	Media
GestorNSAtributo	3	Baja	Baja	Alta
GestorNSFondoDocumental	4	Media	Media	Media
GestorNSTipoDocumental	3	Baja	Baja	Alta
GestorNSTipoAtributo	1	Baja	Baja	Alta

Tabla 7. Evaluación de las clases del módulo mediante la métrica TOC.

2.5.1.1 Resultados de la aplicación de la métrica TOC

En la Figura 5 se puede observar el resultado de la evaluación de los atributos de la métrica TOC.

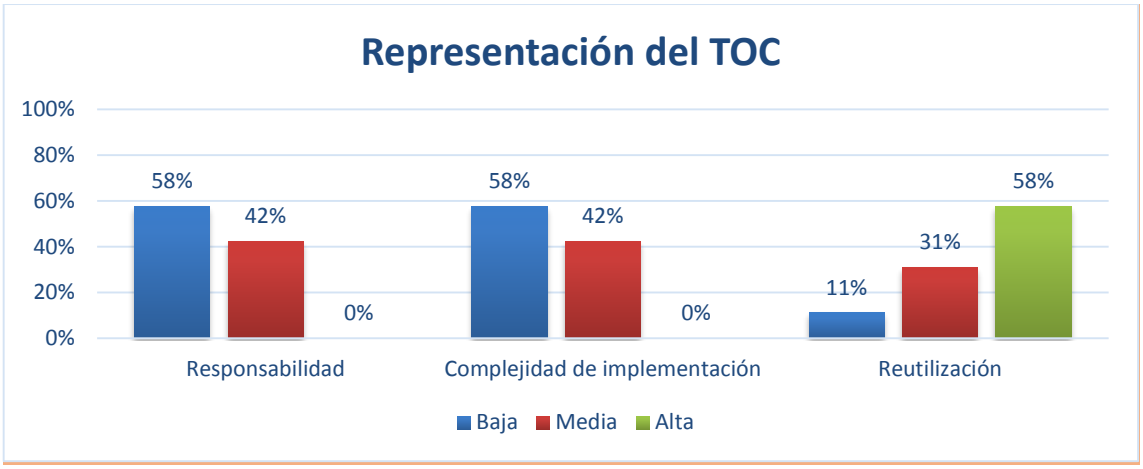


Figura 5. Representación de los resultados de la métrica TOC.

Cuando existe un TOC alto se afectan los parámetros de calidad definidos por esta métrica. Se reduce la reutilización de las clases, la implementación se hace más compleja, las pruebas son difíciles de realizar y aumenta la responsabilidad de las clases.

Haciendo un análisis de los resultados obtenidos en la evaluación de la herramienta de medición de la métrica TOC, se puede concluir que el diseño del módulo tiene una calidad aceptable teniendo en cuenta que el 58 % de las clases incluidas en este módulo, poseen una alta reutilización. Además, este mismo por ciento de clases poseen evaluaciones positivas en los atributos de calidad (Responsabilidad y Complejidad de Implementación) en el diseño propuesto, evidenciando que las clases no tienen tanta responsabilidad, no son tan complejas, y poseen un elevado grado de reutilización. Por lo que se concluye que los resultados obtenidos según esta métrica son positivos.

2.5.2 Relaciones entre clases RC

RC: está dada por el número de relaciones de uso de una clase con otra. Permite evaluar los atributos de calidad: acoplamiento, complejidad de mantenimiento, reutilización y cantidad de pruebas de unidad necesarias de cada clase, teniendo en cuenta las relaciones existentes entre ellas. A continuación, se muestran las clases del sistema, evaluadas en los atributos de calidad mencionados.

Clase	Cantidad de relaciones de uso	Acoplamiento	Complejidad del mantenimiento	Cantidad de pruebas	Reutilización
AccionAdicionarAtributo	2	Medio	Baja	Alta	Baja
AccionGestionarAtributo	3	Media	Media	Media	Media

AccionModificarAtributo	3	Media	Media	Media	Media
AccionAdicionarFondoDocumental	4	Media	Media	Media	Media
AccionGestionarFondoDocumental	3	Media	Media	Media	Media
AccionModificarFondoDocumental	3	Media	Media	Media	Media
AccionAdicionarTipoDocumental	5	Media	Media	Media	Media
AccionAsociarAtributoTipoDocumental	3	Media	Media	Media	Media
AccionAsociarFondoDocumentalTipoDocumental	3	Media	Media	Media	Media
AccionGestionarTipoDocumental	5	Media	Media	Media	Media
AccionModificarTipoDocumental	5	Media	Media	Media	Media
GestorNCAtributo	3	Media	Media	Media	Media
GestorNCFondoDocumental	3	Media	Media	Media	Media
GestorNCTipoDocumental	4	Media	Media	Media	Media
PAtributoDAO	2	Baja	Alta	Baja	Baja
PFondoDocumentalDAO	2	Baja	Alta	Baja	Baja
PTipoAtributoDAO	1	Baja	Alta	Baja	Baja
PTipoDocumentalDAO	2	Baja	Alta	Baja	Baja
GestorNSAtributo	1	Baja	Alta	Baja	Baja
GestorNSFondoDocumental	1	Baja	Alta	Baja	Baja
GestorNSTipoAtributo	1	Baja	Alta	Baja	Baja
GestorNSTipoDocumental	1	Baja	Alta	Baja	Baja

Tabla 8. Instrumento de evaluación de las métricas RC.

2.5.2.1 Resultados de la aplicación de la métrica RC

En la Figura 6 se puede observar el resultado de la evaluación de los atributos de la métrica RC.

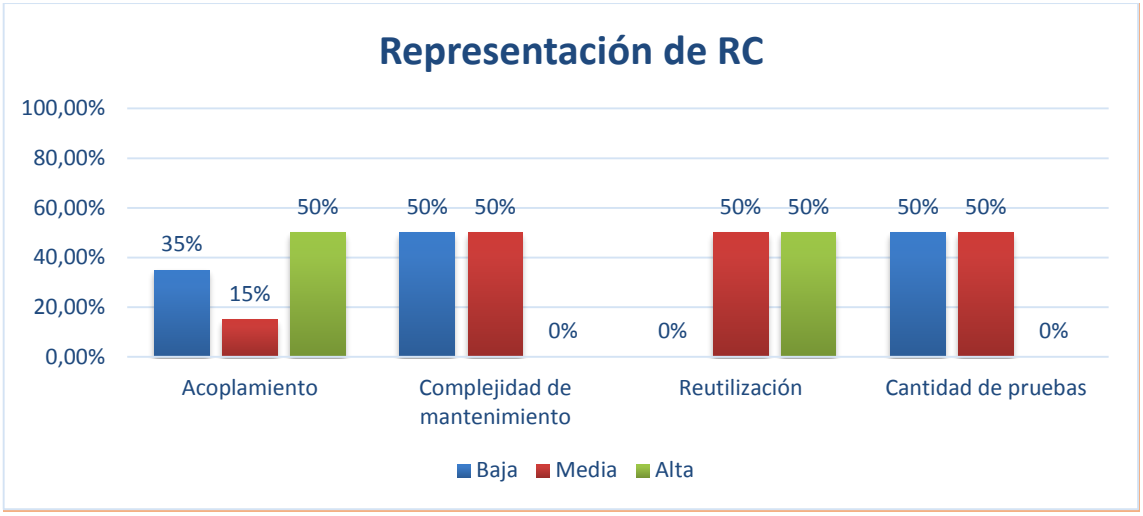


Figura 6. Representación de los resultados de la métrica RC.

Haciendo un análisis de los resultados obtenidos en la evaluación de la herramienta de medición de la métrica RC, se puede concluir, que el diseño realizado tiene una buena calidad teniendo en cuenta que el 50% de las clases poseen acoplamiento alto, lo cual no afecta el diseño, teniendo en cuenta las características del negocio. También la aplicación de la métrica RC arrojó resultados satisfactorios en el atributo complejidad de mantenimiento con un 50% de las clases con índices bajo, lo que facilita las tareas de corrección, modificación y mantenimiento de las clases. El número o el grado de esfuerzo para realizar las pruebas de calidad del producto diseñado es bajo dado que la métrica aplicada arroja un 50% de baja cantidad de pruebas fomentando así un alto índice de reutilización con un 50%.

2.5.3 Árbol de Profundidad de Herencia APH

APH: mide el máximo nivel en la jerarquía de herencia de clases. Permite medir la complejidad de la clase, la complejidad del diseño y el potencial de reutilización de la herencia. A continuación, se muestra el APH individual de cada clase.

Clase	Valor APH
AccionAdicionarAtributo	1
AccionGestionarAtributo	1
AccionModificarAtributo	2
AccionAdicionarFondoDocumental	1
AccionGestionarFondoDocumental	1
AccionModificarFondoDocumental	2
AccionAdicionarTipoDocumental	1

AccionGestionarTipoDocumental	1
AccionModificarTipoDocumental	2
AccionAsociarAtributoTipoDocumental	1
AccionAsociarFondoDocumentalTipoDocumental	1
GestorNSAtributo	1
GestorNSFondoDocumental	1
GestorNSTipoDocumental	1
GestorNSTipoAtributo	1

Tabla 9. Instrumento de evaluación de las métricas APH.

El resultado de la métrica APH es 1.2.

2.5.3.1 Resultados de la aplicación de la métrica APH

Haciendo un análisis del resultado obtenido en la evaluación de la métrica APH sobre el diseño de la solución, se puede concluir, que el diseño presenta un buen nivel de reutilización a través de la herencia y la complejidad de implementación es aceptable.

2.6 Conclusiones del capítulo

El empleo de la metodología AUP – UCI, permitió realizar un trabajo organizado y estructurado, generando los artefactos de la fase de ejecución. Además, se identificaron los requisitos del módulo, obteniéndose un total de 13 requisitos funcionales a tener en cuenta para el desarrollo del módulo. El uso de la arquitectura n-capas y el empleo de patrones de diseño, garantiza obtener una solución de software con poca dependencia entre las clases. La aplicación de métricas de diseño, demostró que las clases del diseño poseen alto acoplamiento, pero no afecta el diseño del negocio, que existe además una baja responsabilidad y complejidad de implementación y una alta reutilización en el diseño propuesto.

CAPÍTULO 3 Validación del módulo

En este capítulo se realizan las pruebas definidas por la metodología seleccionada, para comprobar el correcto funcionamiento de las interfaces.

3.1 Estrategia de Prueba

La Estrategia de Prueba de software integra un conjunto de actividades que describen los pasos que hay que llevar a cabo en un proceso de prueba: la planificación, el diseño de casos de prueba, la ejecución y los resultados, tomando en consideración cuánto esfuerzo y recursos se van a requerir, con el fin de obtener como resultado una correcta construcción del software. La estrategia de pruebas se hace con el objetivo de que el producto de software que se encuentre en desarrollo, reúna con todos los requisitos planteados por el cliente mediante la lógica del negocio (Pressman, 2002).

3.1.1 Prueba

La prueba es un conjunto de actividades que pueden planearse por adelantado y realizarse de manera sistemática. Por esta razón, durante el proceso de software, debe definirse una plantilla para la prueba del software: un conjunto de pasos que incluyen métodos de prueba y técnicas de diseño de casos de prueba específicos (Pressman, 2002).

La prueba del software es un tema más amplio que usualmente se conoce como verificación y validación. La verificación se refiere al conjunto de tareas que garantizan que el software implementa correctamente una función específica. La validación es un conjunto diferente de tareas que aseguran que el software que se construye sigue los requerimientos del cliente.

3.1.1.2 Niveles de Pruebas

Las pruebas son aplicadas con diferentes objetivos y en diferentes escenarios. Por lo que se agrupan por niveles como (Pressman, 2002):

- Nivel de unidad: es el nombre que reciben los procedimientos de pruebas locales a un módulo del sistema. Por definición dichas pruebas cubren la funcionalidad propia del módulo tanto con los métodos de pruebas caja blanca como de caja negra. Combinando ambos métodos se obtiene una mayor fiabilidad del producto final.
- Nivel de integración: es para asegurar que los componentes en el modelo de implementación funcionen correctamente una vez integrados, de manera ascendente y descendente, para ejecutar un caso de uso.
- Nivel del sistema: verifican que se alcance la funcionalidad y el rendimiento del sistema total.
- Nivel de aceptación: son las pruebas finales que se hace antes del despliegue. Su objetivo es verificar que el software esté listo y que puede ser usado por los usuarios finales para cumplir con las funciones y tareas para las cuales fue construido.

Una de las disciplinas de la metodología AUP – UCI, son las pruebas, que tienen como objetivo validar el correcto funcionamiento de la solución. Las pruebas que se utilizaran son las pruebas unitarias, las funcionales y las de aceptación.

3.2 Verificación

3.2.1 Pruebas unitarias

Al desarrollar un nuevo software la primera etapa a considerar es la de las pruebas unitarias, también llamadas pruebas modulares ya que permiten determinar si un módulo de código está listo y correctamente terminado (Oré, 2009). Como parte de las pruebas unitarias, se aplicaron al módulo el método de caja blanca.

Los métodos de pruebas de caja blanca garantizan que se ejerciten todos los caminos independientes de cada módulo, así como la ejecución de todos los bucles y las estructuras de datos internas (Pressman, 2002). Para emplear los mismos se empleó la técnica de camino básico. Se tomó como ejemplo el método onBtnAceptar(), de la clase AccionAdicionarAtributo, como base para realizar el procedimiento anteriormente descrito. El método seleccionado es uno de los más complejos, ya que es una de las principales funcionalidades que responden el objetivo general. En la Figura 10 se muestra este método.

```
@Override
public void onBtnAceptar() {
    List<String> esValido = esValido(); //1
    if (esValido.size() > 0) {//2
        if (esValido.size() == 1) {//3
            GestorMensajes.MensajeAviso(esValido.get(0));//4
        } else if (esValido.size() > 1) {//5
            String mensaje = "";//6
            mensaje = mensaje.concat("<ul>");
            for (int i = 0; i < esValido.size(); i++) {//7
                mensaje = mensaje.concat("<li>" + esValido.get(i) + "</li>");//8
            }
            mensaje = mensaje.concat("</ul>");//9
            GestorMensajes.MensajeAvisoExtendido(mensaje);//10
        }
    } else {//11
        EDAtributo atrib = new EDAtributo();
        atrib.setNombreAtributo(getFormulario().txNombreAtributo.getText());
        atrib.setDescipcion(getFormulario().txaDescripcion.getText());
        atrib.setTipo((EDTipoAtributo) getFormulario().cmbTipoAtributo.getSelectedItem());//12
        atrib.setCapturable(getFormulario().jchCapturble.isSelected());
        try {
            gestor.AdicionarAtributo(atrib);//13
        } catch (Exception ex) {//14
            GestorExcepcion.reportarExcepcion(ex);//15
        }
    }
}//16
```

Figura 7. Método onBtnAceptar().

A continuación, se muestra los pasos para realizar la técnica de camino básico:

1. **Confeccionar el grafo de flujo:** usando el método de la figura 10, se realizó la representación del grafo del flujo.

- Nodos: son círculos que representan una o más sentencias procedimentales.
- Aristas: son flechas que representan el flujo de control y son análogas a las flechas del diagrama de flujo.
- Regiones: son las áreas delimitadas por aristas y nodos.

En la figura 11 se muestra el grafo obtenido.

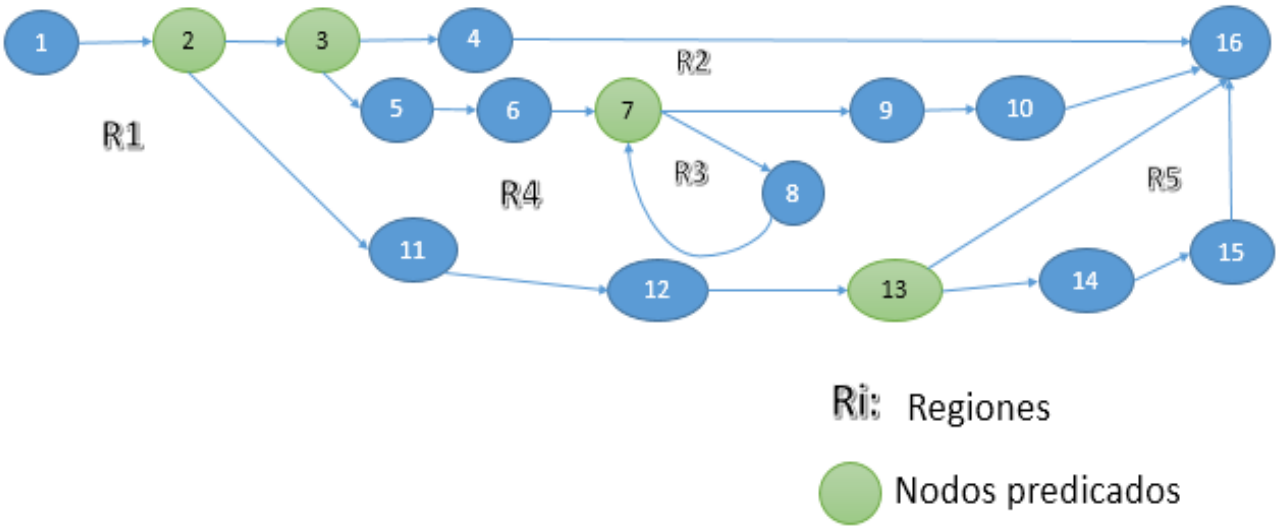


Figura 8. Grafo de camino básico del método onBtnAceptar().

Después de haber realizado el grafo, se calcula la complejidad ciclomática por tres fórmulas distintas, las cuales deben dar el mismo resultado para comprobar que el cálculo sea correcto

2. **Calcular la complejidad ciclomática:** proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado define el número de caminos independientes del conjunto básico de un programa, la complejidad ciclomática se calculó con las siguientes fórmulas:

   - V (G)= A-N+2, donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.

     V (G) = 21-16+2=5

   - V (G) = Regiones

     V (G) = 5

- V (G) = P+ 1, donde P son los nodos predicados.

  V (G) =4+1=5

3. **Determinar un conjunto básico de caminos linealmente independientes:** el valor de V (G) da el número de caminos linealmente independientes de la estructura de control del programa, por lo que se definen los siguientes cinco caminos:

   Camino básico #1: 1-2-3-4-16
   Camino básico #2: 1-2-3-5-6-7-8-7-9-10-16
   Camino básico #3: 1-2-11-12-13-16
   Camino básico #4: 1-2-11-12-13-14-15-16
   Camino básico #5: 1-2-3-5-6-7-9-10-16

4. **Obtención de casos de prueba (CP):** luego de haberle aplicado el método del camino básico a la funcionalidad anterior se comprobó que cada sentencia es ejecutada al menos una vez. El valor calculado como complejidad ciclomática definió el número de caminos independientes del conjunto básico (Pressman, 2002), lo que facilitó conocer el número de pruebas que se deben realizar.

3.2.1.1 Casos de pruebas para el método de prueba de caja blanca.

Caso de prueba: Camino básico 3	
Objetivo de la prueba	Comprobar si se adiciona un atributo.
Datos de entrada	Nombre del atributo, descripción, tipo de atributo y si es capturable.
Procedimientos	Escribir el nombre del atributo, escribir una breve descripción, seleccionar el tipo de atributo y marcar si el atributo es capturable.
Salida esperada	Se adiciona el atributo.

Tabla 10. Casos de prueba del camino 3.

Caso de prueba: Camino básico 1	
Objetivo de la prueba	Comprobar si se muestra el mensaje “El campo nombre del atributo es obligatorio”.
Datos de entrada	Descripción, tipo de atributo y si es capturable.

Procedimientos	Escribir una breve descripción, seleccionar el tipo de atributo y marcar si el atributo es capturable.
Salida esperada	Mostrar mensaje.

Tabla 11. Casos de prueba del camino 1.

Una vez ejecutados todos los casos de pruebas obtenidos a través de la aplicación de la técnica camino básico, se concluye que los mismos fueron probados satisfactoriamente demostrando que el código generado no presenta ciclos infinitos y no existe código innecesario en el sistema desarrollado.

3.2.2 Pruebas funcionales

Las pruebas funcionales como su propio nombre lo indican, prueban una funcionalidad completa, donde pueden estar implicadas una o varias clases y hasta la propia interfaz de usuario.

Los métodos de pruebas de caja negra son pruebas funcionales que se realizan al software, permitiendo al ingeniero de software derivar conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Además, permiten detectar funcionamiento incorrecto o incompleto, errores en la interfaz, errores en la estructura de datos externa, problemas de rendimientos y errores de inicialización y terminación (Pressman, 2002).

Dentro de la prueba de caja negra se incluyen las técnicas de pruebas que serán descritas a continuación:

- **Partición de Equivalencia**: divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- **Análisis de Valores Límites**: prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- **Grafos de Causa-Efecto**: permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

En este trabajo fue seleccionada la técnica de partición de equivalencia, diseñando casos de prueba para cada uno de los requisitos funcionales del módulo. A continuación, se verá un ejemplo del caso de prueba para el requisito funcional Crear atributo, el resto se encuentran en el expediente de caso de prueba del proyecto.

Escenario	Descripción	Descripción	Capturable	Nombre del atributo	Tipo de atributo	Respuesta del sistema	Flujo central
EC 1.1 Crear atributo correctamente	Se introducen los datos correctos para crear un atributo nuevo en el sistema.	V Número único de una página de un documento.	V Si	V Folio	V CheckBox	Se crea en el sistema un nuevo tipo de atributo	Clic en el botón Adicionar
		V Número único de una página de un documento.	V No	V Folio	CheckBox		
EC 1.n Crear atributo incorrectamente	Se introducen los datos incorrectos para crear un atributo nuevo en el sistema.	I	V	V	V	El sistema muestra un mensaje diciendo que hay campos con valores incorrectos.	
		-/899Número único de una página de un documento.	Si	Folio	CheckBox		
		V	I	V	V	Se crea en el sistema un nuevo tipo de atributo	
		Número único de una página de un documento.	N/A	Folio	CheckBox		
		V	V	I	V	El sistema muestra un mensaje diciendo que hay campos con valores incorrectos.	
		Número único de una página de un documento.	No	///////*****	CheckBox		
		V	V	V	I	Se crea en el sistema un nuevo tipo de atributo	
		Número único de una página de un documento.	No	Folio	N/A		
EC 1.n Crear atributo de forma incompleto	Se introducen los datos vacíos para crear un atributo nuevo en el sistema.	I	V	V	V	Se muestra un campo diciendo que hay campos obligatorios Vacíos.	
			No	Folio	CheckBox		
		V	I	V	V	Se crea en el sistema un nuevo tipo de atributo	
		Número único de una página de un documento.	N/A	Folio	ChekBox		
		V	V	I	V		

		Número único de una página de un documento.	Si		CheckBox	Se muestra un campo diciendo que hay campos obligatorios Vacíos.
		V	V	V	I	Se muestra un campo diciendo que hay campos obligatorios Vacíos.
		Número único de una página de un documento.	Si	Folio	Selecciones	

Tabla 12. Caso de prueba de la HU “Crear atributo”.

Con el objetivo de comprobar que las funcionalidades de la herramienta se realizan correctamente y responden a las necesidades del cliente, aplicando los CP antes descritos, la herramienta será revisada por el grupo de Calidad CEGEL.

3.3 Validación

3.3.1 Prueba de aceptación

La prueba de aceptación es generalmente desarrollada y ejecutada por el cliente o un especialista de la aplicación y es conducida a determinar cómo el sistema satisface sus criterios de aceptación, validando los requisitos que han sido levantados para el desarrollo, incluyendo la documentación y procesos de negocio. Está considerada como la fase final del proceso, para crear un producto confiable y apropiado para su uso (Pressman, 2010).

Para la aplicación de estas pruebas se confeccionó un caso de prueba de aceptación por cada HU. A continuación, se muestra un caso de prueba de la HU “Adicionar tipo documental”.

Caso de prueba de aceptación		
Código de caso de prueba: 1	**Nombre historia de usuario:** Adicionar tipo documental.	
Nombre de la persona que realiza el caso de prueba: Yesenia Tabio Borrego.		
Descripción de la prueba: el usuario puede adicionar un tipo documental en el sistema.		
Condiciones de ejecución: el usuario debe estar autenticado en el sistema.		
Entrada/Pasos de ejecución		**Resultados esperados:**
Acción:	**Entrada:**	

El sistema muestra un listado de tipos documentales. El usuario debe seleccionar el botón adicionar. El sistema le muestra una ventana flotante, donde se mostrarán los campos a llenar. El usuario escribe los datos del tipo documental, una vez escritos los datos el usuario debe seleccionar el botón aceptar.	Texto	Se adiciona un tipo documental en el listado de los mismos.
Evaluación de prueba:		

Tabla 13. Caso de prueba de aceptación.

Con el objetivo de revisar la herramienta, teniendo en cuenta el CP definido, se realizará un encuentro con el Ing. Yunior Duque Aguilar.

3.3 Conclusiones del capítulo

Al aplicar las pruebas unitarias utilizando el método de prueba de caja blanca, se obtuvo que no existe código innecesario. Cuando sean aplicadas las pruebas funcionales y de aceptación se certificarán la obtención de un software funcional que responde a los requisitos del cliente, avalando por cada caso con las actas de liberación y aceptación emitidas.

CONCLUSIONES GENERALES

Las soluciones existentes consultadas no satisfacen los requerimientos del problema de la investigación, no obstante, aportaron algunos elementos tenidos en cuenta en la implementación. Mediante la implementación de las funcionalidades identificadas se obtuvo un módulo que organiza y procesa los documentos para garantizar la comunicación de los módulos y componentes del sistema para la obtención de documentos digitales con valor legal. Las pruebas realizadas permitieron comprobar el adecuado diseño del módulo y el cumplimento de los requisitos acordados.

Bibliografía

ABBYY. 2015. Softonic. [En línea] 2015. [Citado el: 14 de Marzo de 2016.] http://abbyy-finereader.softonic.com/.

ALBERT, S.A. 2010. *Manual de apoyo para recepción y devolución de documentos del centro de Digitalización.* 2010. RN2-TD-DE-019..

Alcantara Rabí, Dayanis Elvia y Hernández Luque, Eylín. 2010. Propuesta de una guía de métricas para evaluar el desarrollo de los Sistemas de Información Geográfica. *Revista vinculado.* [En línea] 2010. [Citado el: 26 de Junio de 2016.] http://vinculando.org/articulos/sociedad_america_latina/propuesta_guia_de_medidas_para_evaluacion_sistemas_informacion.html.

Archivo Nacional Chile. 2016. Archivo Nacional Chile. [En línea] 2016. [Citado el: 28 de Mayo de 2016.] http://www.archivonacional.cl/616/w3-article-10983.html.

Asenjo González, Diego Andrés. 2003. *Patrones de Diseño.* Habana : s.n., 2003.

Briand, L., S. Morasca. 1996. *Property-Based Software Engineering Measurement.* s.l. : IEEE Transactions on Software Engineering, 1996. 22(1): 68-86..

Buono, Enrique. 2012. Patrones fundamentales en el Diseño Orientado a Objetos (DOO). *Red Colaborativa Postgrado UCV.* [En línea] 2012. [Citado el: 24 de Junio de 2016.] http://kuainasi.ciens.ucv.ve/red_educativa/blogs/42.

Calabria, Luis y Píriz, Pablo. 2003. *Metodología XP.* Universidad ORT Uruguay : Editorial: Universidad ORT, 2003.

Concha, Gastón y Naser, Alejandra. 2014. *Repositorio Digital.* [En línea] 2014. [Citado el: 15 de Junio de 2016.] http://repositorio.cepal.org/bitstream/handle/11362/3969/S2012004.pdf?sequence=1#page=11.

Corporation, EMC. 2006. QuickScan 4.5. 2006.

David Garlan, M.S. 1994. *An Introduction to Software Architecture.* 1994. CMU/SEI-94-TR-21, ESC-TR-94-21.

BIBLIOGRAFÍA

Fernández, Oscar Belmonte. 2005. Introducción al lenguaje de programación Java: Una guía básica. [En línea] 2005. [Citado el: 11 de Mayo de 2016.] http://www3.uji.es/~belfern/pdidoc/IX26/Documentos/introJava.pdf.

Font, Sandra González Valdés y René Suárez. 2011. *Diseño e implementación de los módulos de Preparación de Documentos, Digitalización de Documentos y Asociación de Metadatos del Centro de Digitalización para la División de Antecedentes Penales.* 2011.

Gonzales Mesa, Elda. 2006. *La digitalización de documentos, ¿amiga o enemiga?* s.l. : Bibliotecas. Anales de Investigación, 2006.

Harrison, Daniel Rodriguez y Rachel. 2001. An Overview of Object-Oriented Design Metrics. [En línea] 2001. [Citado el: 28 de Junio de 2016.] http://www.cc.uah.es/drg/b/RodHarRama00.English.pdf.

Infoviews. 2011. *Administración Digital de Contenidos.* s.l. : Infoviews,S.A. DE C.V, 2011.

Isaías Carrillo Perez, Rodrigo Perez Gonzales, David Rodriguez. 2008. Metodologia de desarrollo de software. [En línea] 2008. [Citado el: 5 de Mayo de 2016.] http://es.slideshare.net/alexandervilcapazachavez/metodologias-de-desarrollo1.

Larman, Craig. 1999. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* Mexico : PRENTICE-HALL, 1999.

—. 1999. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* Mexico : PRENTICE-HALL, 1999.

Lenis Matos Rodriguez, Daniel Sarmiento Sastriques. 2012. *Propuesta de un mecanismo de seguridad para el marco de trabajo Kairos.* 2012.

Luis Calabria, Pablo Piriz. 2003. *Metodología XP.* 2003.

Mena, Mayra. 2005. *Gestión documental y organización de archivos.* s.l. : Felix Valera, 2005.

Microsoft | Achitecture. 2010. *Guía de Arquitectura N-Capas orientada al Dominio con .NET 4.0.* 2010.

Oracle Corporation NetBeans. 2016. NetBeans. [En línea] 2016. [Citado el: 11 de Mayo de 2016.] https://netbeans.org/community/releases/67/.

Oré, Alexander. 2009. Pruebas unitarias. 2009.

BIBLIOGRAFÍA

Pfleeger, S. L. 1997. *Assessing Software Measurement.* s.l. : IEEE Software, 1997. March/April: 25-26.

Pressman, Roger S. 2010. Ingeniería del Software. 2010. 7.

Pressman, Roger S. 2002. *Ingenería del software: Un enfoque práctico.* 2002.

Programación en castellano. 2016. Conceptos básicos de ORM (Object Relational Mapping). [En línea] 2016. [Citado el: 14 de Junio de 2016.] http://programacion.net/articulo/conceptos_basicos_de_orm_object_relational_mapping_349.

Quinzaños, Pilar Rivas. 2006. *La tipología documental y las edades de los documentos en los archivos de empresa.* Madrid : s.n., 2006.

Richardmx. 2008. Java Mexico. [En línea] 2008. [Citado el: 14 de Junio de 2016.] http://www.javamexico.org/blogs/richardmx/que_es_data_access_object.

Sánchez, Tamara Rodríguez. 2015. *Metodología de desarrollo para la Actividad productiva de la UCI.* 2015.

Sistema de identificación, migración y extranjería. 2011. *Sistema para la Digitalización de Alfabética.* 2011.

Sommerville. 2005. *Ingenería del software.* s.l. : Madrid: Pearson., 2005.

Stinson, Barry. 2001. *PostgreSQL: Essential Reference.* s.l. : s.l. : Sams Publishing, 2001. ISBN: 0752064711216.

Universidad de Navarra. 2009. Universidad de Navarra. [En línea] 2009. [Citado el: 10 de Marzo de 2016.] http://www.unav.edu/servicio/archivo/elvalorjuridico.

USM.cl. 2015. UNIVERSIDAD TÉCNICA FEDERICO SANTA MARIA. [En línea] Sitio web administrado por la Dirección General de Comunicaciones, 2015. [Citado el: 24 de Junio de 2016.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08.

Visconti, Marcello y Astudillo, Hernán. 2004. *Fundamento de Ingeniería de Software.* Chile : USM, 2004.

Visual Paradigm International. 2007. Free Download Manager. [En línea] 5 de Marzo de 2007. [Citado el: 11 de Mayo de 2016.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.