UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD *3*

Gestor documental para el Sistema de Planificación de Actividades SIPAC.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autor: Yosmar Puig Sánchez

Tutor: Ing. Rodolfo Rodríguez Molinet

Co-Tutor: Ing.Mairelys Fernández González

Ing. Anabel Reyes Rosa

Ciudad de la Habana, Junio de 2013.

Es dudoso que el género Humano logre crear un enigma que el mismo ingenio humano no resuelva.

Edgar Allan Poe.

Los hombres y pueblos en decadencia viven acordándose de dónde vienen; los hombres geniales y pueblos fuertes sólo necesitan saber a dónde van.

José Ingenieros

Declaración de autoría:

Declaramos que somos los únicos autores del trabajo “Gestor documental para el Sistema de Planificación de Actividades SIPAC” y autorizamos a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio. Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Yosmar Puig Sánchez	Ing. Rodolfo Rodríguez Molinet
Autor	Tutor
Ing. Mairelys Fernández González	Ing. Anabel Reyes Rosa
Co-tutor	Co-tutor

Agradecimientos

Agradecer ante todo a la Revolución,

A la Universidad de las Ciencias Informáticas y a Fidel;

ellos son los principales responsables de formarnos como ingenieros.

A mis padres, por educarme e inculcarme buenos principios,

por todo el apoyo que siempre he podido encontrar en ellos

y por el sacrificio y esfuerzo que han tenido que hacer durante

todos estos años en función de mi formación como profesional.

A mi hermana quien ha sido mi fuente de inspiración,

por su apoyo y su constante preocupación,

A todos los familiares y amigos que han estado siempre para mí.

A mis Tutoras Mairelys y Anabel

quienes constituyeron una pieza fundamental en este proceso,

por su apoyo, ayuda y amistad

A todos mis compañeros de aula con los que compartí durante estos 5 años

y a los profesores que me impartieron clases

y me ayudaron a formarme como futuro profesional

Dedicatoria

Dedico este trabajo a mis seres más queridos, mi familia,
por ser mí sostén, ya que siempre han estado presentes;
pues sin su apoyo y confianza no habría podido alcanzar esta meta
A mami por su fe inquebrantable, por dármelo todo,
su amor, su cariño y su ternura
Por intentar transmitirme toda su sabiduría.
A papi por el amor y el cariño que me ha dado en estos 23 años,
por ser mí ejemplo, mi vida y mi orgullo.

Resumen

En Cuba se han realizado en los últimos años un conjunto de transformaciones con el fin de lograr mejoras en el sistema de planificación de las entidades. Dicha planificación se elabora bajo el concepto de Planificación de Actividades que surge de la unión de la Dirección por Objetivos y la Planeación Estratégica. Debido a que el manejo de la información que se genera durante su ejecución resulta complejo y engorroso en determinados escenarios, es necesaria la utilización de sistemas que informaticen dichas tareas. No obstante, las herramientas que se utilizan en el país para realizar la planificación no tienen en cuenta la realización del manejo de planes, objetivos y actividades como define el modelo cubano, por lo que fue necesario desarrollar el Sistema para la Planificación de Actividades (SIPAC). Sin embargo, en dicho sistema surge la necesidad de tener constancia de sucesos ocurridos y de documentar acciones y archivos para facilitar el seguimiento e integridad de la información asociada a los elementos planificados. Después de realizar una investigación tanto a nivel nacional como internacional se evidencia la inexistencia de una solución informática que pueda ser integrada a SIPAC; por lo que se propone como solución realizar el análisis, diseño e implementación de la funcionalidad Gestor documental para el Sistema de Planificación de Actividades.

Palabras claves:

Planificación de Actividades, planificación, planes, objetivos, actividades, seguimiento.

Índice

Índice

Índice de figuras

Índice de tablas

INTRODUCCIÓN.

Con el paso de los años, Cuba, ha venido realizando mejoras al sistema de planificación de las entidades con el fin de lograr mayor productividad en las empresas. A partir de la aplicación del perfeccionamiento empresarial, la dirección del país trazó una estrategia económica con el propósito de garantizar el mejoramiento y adecuación de la planificación estratégica y eficacia organizativa.

Para lograr lo planteado anteriormente, se definió el modelo "Planificación de Actividades" descrito en la Instrucción no.1 del Presidente de los Consejos de Estado y de Ministros para la planificación de los objetivos y actividades en los órganos, organismos de la administración central de estado, entidades nacionales y administraciones locales del poder popular. Este modelo define las directrices de la planificación estratégica y operativa de actividades, posibilitando que las actividades individuales y organizacionales se ajusten a las necesidades propias de las instituciones cubanas.

La Planificación de Actividades está determinada por la realización de los procesos: registro, control y seguimiento de las actividades tanto individuales como a nivel de entidad. El manejo de la información que se genera durante su ejecución resulta complejo y engorroso en determinados escenarios y hoy se hace necesaria la utilización de sistemas que informaticen dichas tareas. [1]

En algunas entidades cubanas, el modelo de planificación antes mencionado se gestiona a partir del sistema P-TRAB[1], pero debido a su arquitectura (desarrollado sobre MSDOS[2]) y a la concepción del modelo de Planificación de Actividades mediante el que fue concebido, este no posibilita desarrollar el subproceso correspondiente al registro, control y seguimiento de las actividades. En el país, algunas entidades utilizan como alternativa herramientas del paquete de programas Microsoft Office como son Outlook, Project y Excel, pero estas no tienen en cuenta el manejo de planes, objetivos y actividades como define el modelo cubano de planificación.

Atendiendo a las necesidades de informatización asociadas a la Planificación de Actividades en las entidades nacionales, y debido a que los sistemas utilizados para este propósito no las

[1] **P-TRAB**: sistema informático cubano que automatiza desde finales de la década del 90 el proceso de Planificación por Actividades.
[2] **S.O. MSDOS: Microsoft** Disk Operating System, sistema operativo de disco de Microsoft.

satisfacen, se requiere desarrollar un sistema destinado a facilitar la gestión de los planes, objetivos y actividades a todos los niveles organizacionales en el proceso de Planificación de Actividades. Por este motivo el Centro de Informatización de la Gestión de Entidades (CEIGE) de la Universidad de Ciencias Informáticas (UCI), en conjunto con la Unidad de Compatibilización, Integración y Desarrollo de Software para la Defensa (UCID) de las Fuerzas Armadas Revolucionarias (FAR) se encargan de desarrollar un Sistema de Planificación de Actividades (SIPAC).

SIPAC se encarga de garantizar el seguimiento de las tareas principales en las entidades a corto, mediano y largo plazo, sin embargo, en este sistema se dificulta el seguimiento e integridad de las necesidades de información en el proceso de Planeación Estratégica y Operativa, ya que no ofrece la posibilidad de llevar la gestión documental asociada a los elementos de la planificación, lo que trae consigo el desconocimiento en determinadas ocasiones de la procedencia, metas y procedimientos específicos para el desarrollo y cumplimiento de los mismos.

De la problemática anteriormente expuesta se ha identificado como **problema a resolver:** en el Sistema de Planificación de Actividades SIPAC no está disponible la información asociada a los elementos planificados, limitando la realización de los procesos de ejecución y control de la Planeación Estratégica y Operativa.

Objetivo general: desarrollar la funcionalidad Gestor documental del Sistema de Planificación de Actividades SIPAC, para disponer de la información asociada a los elementos planificados de manera que facilite la realización del proceso de Planeación Estratégica y Operativa.

Objetivos específicos:

1. Elaborar el marco teórico para fundamentar la investigación.
2. Realizar la descripción de los requisitos identificados.
3. Realizar el Análisis y Diseño de la solución.
4. Implementar la solución.
5. Validar la solución.

Para darle cumplimiento a los objetivos específicos se desglosan las siguientes **tareas de la investigación:**

1. Valoración y análisis de la realización del proceso de gestión documental en sistemas informáticos.
2. Análisis de la Arquitectura Base definida en el proyecto.
3. Análisis del Modelo de Desarrollo definido por CEIGE.
4. Elaboración del modelo conceptual.
5. Identificación de los requisitos de software.
6. Especificación de requisitos de software
7. Validación de los requisitos de software identificados.
8. Elaboración de los Diseños de Casos de Prueba.
9. Actualización del Modelo de datos.
10. Definición de los prototipos de interfaz de usuario.
11. Elaboración de los diagramas de clases del diseño.
12. Elaboración de los diagramas de interacción de acuerdo a los requisitos definidos.
13. Validación del Diseño a través de métricas.
14. Actualización del modelo de componentes del sistema.
15. Implementación de las funcionalidades del sistema.
16. Validación de la solución mediante pruebas de calidad interna.
17. Validar la solución de software propuesta mediante la aplicación de pruebas de Caja Blanca y Caja Negra.

Con vista a la solución del problema se define como **objeto de estudio**: procesos de gestión documental.

El Campo de acción: la gestión documental en el sistema de planificación de actividades SIPAC.

Como **idea a defender** se plantea**:** el desarrollo de la funcionalidad Gestor de documental para el Sistema de Planificación de Actividades SIPAC, permitirá la disponibilidad de la información asociada a los elementos planificados en el proceso de Planeación Estratégica y Operativa.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

1.1 Introducción.

En el presente capítulo se realizará el análisis y valoración de sistemas nacionales e internacionales que informatizan el proceso de gestión documental, con el objetivo de conocer qué aspectos son imprescindibles para un sistema de este tipo. Para un mayor entendimiento se enuncian los conceptos más relevantes relacionados con este tema. Se exponen las características del modelo de desarrollo utilizado en CEIGE y finalmente, se especifican los lenguajes, frameworks, las tecnologías y las herramientas que se utilizarán durante la investigación.

1.2 Marco conceptual.

A continuación se definen los principales aspectos teóricos relacionados con la gestión documental.

Gestión documental.

Según el Diccionario de Terminología Archivística del Consejo Internacional de Archivos, la gestión documental es "*un área de la administración general que se encarga de garantizar la economía y eficiencia en la creación, mantenimiento, uso y disposición de los documentos administrativos durante todo su ciclo de vida*".

Por otra parte, es el *área de gestión responsable de un control eficaz y sistemático de la creación, la recepción, el mantenimiento, el uso y la disposición de documentos de archivo, incluidos los procesos para incorporar y mantener en forma de documentos la información y prueba de las actividades y operaciones de la organización*. [2]

A partir del concepto planteado anteriormente se define que la gestión documental es responsable de llevar un control eficaz y sistemático de la creación, la recepción, mantenimiento, uso y disposición de documentos de archivo, mejorando la forma de cómo se organizan y recuperan los documentos. Además abarca todo el ciclo de vida del documento desde su creación hasta su eliminación y se ha convertido en un proceso vital en las organizaciones por la magnitud que alcanzan los documentos.

1.3 Estado del arte.

Existen numerosos sistemas para la gestión documental, a continuación serán descritas y caracterizadas algunas aplicaciones informáticas que se encargan de la Gestión de documentos tanto en el ámbito nacional como internacional. Esta investigación se realiza teniendo en cuenta los siguientes indicadores: tipo de aplicación, tecnología utilizada, licencia y compartimentación de la información.

Internacional.

1.3.1 Alfresco.

Características del producto:

Es un Gestor de Contenidos Empresariales, desarrollado en Java. Es compatible con sistemas operativos como: Microsoft Windows, Linux y Unix. Existen dos versiones, una de libre distribución llamada Community Edition y otra más completa que es de pago (tiene soporte incluido y es llamada Enterprise Edition). La arquitectura de Alfresco está basada en un repositorio de contenido único, gestionando el almacenamiento de la información, indexando y categorizando los contenidos para su rápida búsqueda y localización, almacenando los metadatos de los documentos en Sistemas de Gestión de Bases de Datos (SGBD). [3]

Tipo de aplicación: web.

Tecnologías: Java, Oracle, SQL, PostgreSQL, MySQL.

Licencia: el ECM Alfresco es una alternativa libre y de código abierto que permite desarrollar proyectos de contenidos empresariales. Cuenta con una creciente y fuerte comunidad de desarrolladores, donde se insertan cada vez más nuevas tecnologías y contribuciones de la comunidad que permiten obtener un software de alta calidad. [4]

Está construido mediante los últimos componentes de infraestructuras de código abierto, que incluyen: Spring, Hibernate, Lucene y MyFaces y se basa en Programación Orientada a Aspectos (Aspect Oriented Programming). No cobra las tradicionales cuotas de licencia. [5]

Compartimentación: el módulo de gestión de registros de Alfresco cuenta con la certificación para el estándar DoD 5015.02[3] de EE.UU. y ofrece seguridad a nivel de documentos y un sistema centralizado de autenticación y autorización (SSO, por sus siglas en inglés) con Protocolo Ligero de Acceso a Directorios (LDAP, por sus siglas en inglés)o directorio activo.[6]

1.3.2 KnowledgeTree.

Características del producto:

KnowledgeTree, es una herramienta que proporciona un poderoso mecanismo de control que accede a los documentos para editarlos y poder ser enviados. Proporciona un acceso a través de interfaces familiares a su organización, ya sean basadas en la web o dentro de las aplicaciones de productividad. También es una manera de aplicar la Web 2.0 a las empresas permitiendo que cada usuario interactúe con la aplicación; creando vistas propias, contenidos de ayuda y mostrando anuncios personalizados. Es un sistema de gestión documental con arquitectura web. Ofrece un almacén o repositorio, soporta el desarrollo de flujo de trabajo, publicación de contenidos e incluso definición de métricas relativas a la gestión de contenidos. Incluye un control avanzado de versiones de documentos, múltiples tipos de búsqueda, campos para la definición por el usuario de metadatos, un panel de control configurable a medida de las necesidades del usuario. [4]

Tipo de aplicación: web.

Tecnologías: Java, Oracle, MySQL, SQL, PostgreSQL.

Licencia: existen dos versiones de esta aplicación una licencia privativa y otra versión de código abierto con licencia GNU/GPL.

Compartimentación: KnowledgeTree permite crear usuarios, grupos y roles a los cuales se les asignan de forma controlada y segura privilegios de acceso y permisos. [7]

[3] **DoD 5015.02**: manual de operaciones del programa de seguridad industrial nacional para Departamento de defensa de EE.UU. [48]

Nacional.

A nivel nacional, una de las soluciones informáticas que se ha desarrollado para la gestión de documentos es el eXcriba (realizado por la Universidad de las Ciencias Informáticas).

1.3.3 eXcriba.

Características del producto:

El eXcriba es un sistema desarrollado en la Universidad de las Ciencias Informáticas (UCI), que le permite al usuario gestionar el documento a lo largo de su ciclo de vida, desde la creación del mismo, hasta la disposición o almacenamiento permanente en un archivo, transitando por los correspondientes períodos de trámite. Además permite la colaboración a los empleados, socios, clientes y en general a los usuarios que comparten información de modo que esta pueda ser utilizada y reutilizada por los demás usuarios. [8]

El eXcriba tiene como núcleo el ECM Alfresco y ofrece una interfaz para poder llevar a los clientes las bondades que este gestor de contenidos empresarial provee como repositorio documental [4].

Desde el punto de vista conceptual “eXcriba” es un producto genérico, centrado en el cumplimiento con los requerimientos agrupados en las normas ISO15489[4], ISAD(G) [5], así como en la especificación de MoReq[6]. Es por ello que esta solución brinda a los clientes la optimización de recursos importantes en cualquier institución dígase tiempo, organización y tecnología en función del correcto desempeño documental de la institución. [9]

Tipo de aplicación: web.

Tecnologías: PHP v5.3.0, JQuery v1.3.2, Java, Alfresco.

[4] **ISO15489**: se centra en los principios de la gestión de documentos y establece los requisitos básicos para que las organizaciones puedan establecer un marco de buenas prácticas que mejore la forma sistemática y efectiva la creación y mantenimiento de sus documentos, apoyando la política y los objetivos de la organización. [42]

[5] **ISAD(G).** norma internacional para la descripción de funciones: Norma elaborada por el Consejo Internacional de Archivística (ICA). Constituye una guía general para la elaboración de descripciones archivísticas. [42]

[6] **MoReq:** modelo de requisitos para la gestión de documentos electrónicos de archivo, que incide especialmente en los requisitos funcionales mediante un sistema de gestión de documentos electrónicos de archivo, aplicable tanto para organizaciones públicas como privadas y que se puede utilizar igualmente durante la gestión, para llevarla a cabo de forma correcta y efectiva, como a posteriori, para evaluarla. [44]

Licencia: licencia GPLv2.

Compartimentación: Todas las acciones que lo requieran deben tener seguridad a nivel de permisos. Para realizar cualquier operación, el usuario debe estar autenticado garantizando la confidencialidad e integridad ante la información que maneja. [10]

1.3.5 Valoración del estado del arte.

El análisis realizado de las soluciones informáticas antes presentadas arroja como resultado un conjunto de elementos a tener en cuenta para el desarrollo de la propuesta de solución y evidencia la inexistencia de una solución informática que pueda ser integrada al sistema SIPAC para el desarrollo del proceso de gestión documental. Independientemente de que son aplicaciones web que cuentan con al menos una versión de código abierto, compatibles con Windows y que la mayoría han sido desarrollados en PHP y Postgres; no cumplen íntegramente con las características del framework sobre el cual está desarrollado SIPAC. Además, el otorgamiento de permisos a los usuarios de lectura y/o escritura sobre los documentos no está sobre la base de la compartimentación de información, como está concebido en dicho sistema de planificación. En SIPAC, cada usuario es responsable de otorgar los permisos sobre la información a sus subordinados; en dependencia de sus roles y responsabilidades, a partir de la estructura definida en el sistema y a su vez debe cumplir con lo que desde el nivel superior se le asigne. A partir del planteamiento anterior se decide desarrollar la funcionalidad Gestor documental como parte de dichos sistema de Planificación.

1.4 Modelo de desarrollo.

1.4.1 Características.

CEIGE dirige sus resultados hacia la esfera de la gestión de empresas y entidades. La producción se centra en el desarrollo de proyectos generalmente de gran magnitud, por lo que se hace necesario contar con un modelo estandarizado, que establezca las distintas fases por las que se debe transitar y el conjunto de artefactos a generar en cada una de ellas. Teniendo como precedente dicha necesidad y en colaboración con las distintas líneas de desarrollo donde se ejecutan cada uno de estos proyectos, se propone el Modelo de desarrollo de software para el CEIGE. El resultado detalla el ciclo de vida de sus

proyectos con la incorporación de los distintos subprocesos dictados por el Nivel II de CMMI[7] (*Capability Maturity Model Integration*), certificación obtenida por el centro en julio de 2011 y reconocida por el SEI (*Software Engineering Institute*), como aval de la calidad de su proceso de desarrollo de soluciones informáticas.

1.4.2 Descripción del ciclo de vida

Teniendo en cuenta que el proyecto SIPAC pertenece a CEIGE, el modelo de desarrollo que guiará la solución será precisamente el definido en dicho centro. A continuación se describen los elementos fundamentales:

Inicio o Estudio preliminar: durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto a un alto nivel, la evaluación de la factibilidad del proyecto y el registro de este. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo, y decidir si se ejecuta o no el proyecto.

Los objetivos de la fase son:

- Asegurar la factibilidad del proyecto.
- Establecer un plan para la ejecución del proyecto.

Hitos:

- Plan de desarrollo de software.
- Acta de inicio del proyecto firmada.

Desarrollo: en esta fase 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 refinan los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

El objetivo de esta fase es:

- Obtener un sistema que satisfaga las necesidades de los clientes y usuarios finales.

Hito:

[7] **CMMI** es un modelo descriptivo que detalla los atributos esenciales que deberían caracterizar a una organización en un determinado nivel de maduración. [45]

- Producto liberado por entidad certificadora de calidad.

En esta fase se ejecutan las disciplinas Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas y Pruebas de liberación. [11]

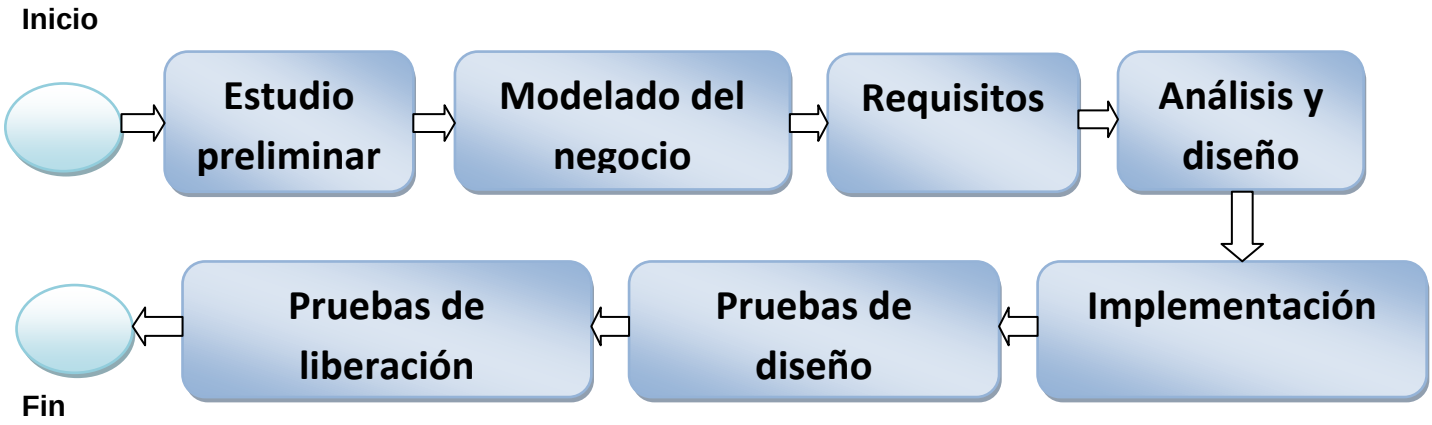


Figura 1 Fases o etapas del ciclo de vida de un proyecto.

Descripción de la fase Estudio preliminar.

Se llevan a cabo las actividades relacionadas con la planeación del proyecto a un alto nivel. Se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto y realizar estimaciones de tiempo, esfuerzo y costo.

Descripción de la disciplina Modelado del negocio.

Es la fase destinada a comprender los procesos de negocio de la organización. Se comprende cómo funciona el negocio que se desea automatizar para tener garantías de que el software desarrollado va a cumplir su propósito.

Descripción de la disciplina Requisitos.

El esfuerzo principal en la fase de Requisitos es desarrollar un modelo del sistema que se va a construir. Incluye un conjunto de artefactos que describen todas las interacciones que tendrán los usuarios con el software y que responden a los requisitos funcionales del sistema. Se especifican los requisitos funcionales y no funcionales.

Descripción de la disciplina Análisis y diseño.

Durante esta fase es modelado el sistema para que soporte todos los requisitos. Esto contribuye a una arquitectura sólida y estable que se convierte en un plano para la próxima fase. Los artefactos generados en esta etapa son más formales y específicos de una implementación. En caso de llevarse a cabo la

reutilización de componentes software ya desarrollados, durante esta fase se ajusta el modelado existente a los requisitos actuales.

Descripción de la disciplina Implementación.

A partir de los resultados del análisis y diseño se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ejecutables y similares. Al reutilizar componentes software ya implementados se lleva a cabo el desarrollo necesario para ajustar a los requisitos actuales y posteriormente realizar la integración de los componentes.

Descripción de la disciplina Pruebas internas.

Durante esta fase se desarrollan las pruebas del grupo de calidad del centro verificando el resultado de la implementación. Permite identificar posibles errores en la documentación y el software, es decir requisitos que el producto debería cumplir y que aún no los cumple.

Descripción de la disciplina Pruebas de liberación.

Se aplican pruebas diseñadas e implementadas por el Laboratorio Industrial de Pruebas de Software a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación. [11]

1.5 Lenguajes de modelado y desarrollo.

Las herramientas, lenguajes de modelado y desarrollo, además de las tecnologías que se utilizarán en la solución son las definidas por el centro CEIGE para el desarrollo de sus productos.

1.5.1 Lenguajes de modelado.

Se denomina lenguaje de modelado de objetos al conjunto estandarizado de símbolos y las distintas combinaciones de la disposición para modelar un diseño de software. [12]

UML 2.0.

(Unifed Modeling Languaje) Lenguaje unificado de Modelado por sus siglas en inglés (UML), es un lenguaje para la especificación, visualización, construcción y documentación de los artefactos de un proceso de sistema intensivo. Fue originalmente concebido por la Corporación Rational Software y tres de los más prominentes métodologistas en la industria de la tecnología y sistemas de información: Grady Booch, James Rumbaugh, e Ivar Jacobson. El lenguaje ha sido presentado al Object Management Group (OMG) y aprobado por este como un estándar (noviembre 17 de 1997). [13]

1.5.2 Lenguajes de programación:

Lenguaje del lado del servidor:

Se clasifica así al lenguaje de programación en la tecnología cliente servidor que se ejecuta del lado del servidor y del cual los usuarios solo obtienen el beneficio del procesamiento de la información. [12]

PHP 5.2.5.

PHP es un lenguaje de programación usado normalmente para la creación de páginas web dinámicas. Es conocido como una tecnología de código abierto, que resulta muy útil para diseñar de forma rápida y eficaz aplicaciones web dirigidas a bases de datos. Es un potente lenguaje de secuencia de comandos diseñado específicamente para permitir a los programadores crear aplicaciones en la web con distintas prestaciones de forma rápida. No requiere definición de tipos de variables, no es un lenguaje de marcas. Su interpretación y ejecución se realizan en el servidor en el cual se encuentra almacenada la página y el cliente solo recibe el resultado de la ejecución. Permite la conexión a numerosas bases de datos de forma nativa tales como Postgres, MySQL, Oracle, Microsoft SQL Server, entre otras, lo cual permite la creación de aplicaciones web robustas.

PHP tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos tales como UNIX, Linux y Windows. [14]

Lenguajes del lado del cliente:

Un lenguaje del lado cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio. El código, tanto del hipertexto como de los scripts, es accesible a cualquiera y ello puede afectar a la seguridad. [12]

HTML.

HTML es el acrónimo en inglés de HyperText Markup Language (en español se traduce como lenguaje de marcado de hipertexto). HTML es un lenguaje abstracto que aplicaciones pueden usar para representar documentos (se les llama documentos a instancias completas, como lo son las páginas web), y que puede ser transmitido fácilmente por algún medio, como lo es Internet. Los navegadores de Internet procesan e interpretan documentos descritos en HTML usando un analizador de HTML.

El lenguaje HTML está definido por lo que se llama etiquetas, que se encuentran entre los símbolos **<** y **>**, de la siguiente forma: **<etiqueta>**. El contenido de los documentos está definido entre estas etiquetas, mismas que tienen una representación para indicar su límite, de la siguiente forma: **<etiqueta>Contenido</etiqueta>**. A estas etiquetas y su contenido se les conoce como **elementos**. Un elemento puede consistir de varias etiquetas anidadas. [15]

JavaScript.

JavaScript, es un lenguaje interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Al contrario de Java, JavaScript no es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de herencia, es más bien un lenguaje basado en prototipos, donde las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM[8]. JavaScript es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos. Con JavaScript se puede crear diferentes efectos e interactuar con los usuarios. JavaScript es soportado por la mayoría de los navegadores como Internet Explorer, Netscape y Mozilla Firefox. [16]

CSS.

Las Hojas de Estilo en Cascada (Cascading Style Sheet) son útiles para definir los atributos visuales en documentos HTML. Esto les da a los autores métodos poderosos para definir el aspecto visual del documento, mientras que separa la parte semántica (HTML) de la estructura de la presentación (style sheets). Constituyen el estándar para la inserción de estilos a documentos estructurados, como por ejemplo, páginas HTML o Extensive Markup Language (XML). [17]

XML.

El Extensive Markup Language se trata de un estándar del W3C[9] que posibilita compartir la información de una manera segura, fiable y fácil. Además, permite compartir los datos con los que se trabaja a todos los

[8] (**DOM**)**'Modelo de Objetos del Documento' o 'Modelo en Objetos para la Representación de Documentos'** es esencialmente una interfaz de programación de aplicaciones (API) que proporciona un conjunto estándar de objetos para representar documentos HTML y XML

[9] **W3C**: El World Wide Web Consortium, abreviado W3C, es un consorcio internacional que produce recomendaciones para la World Wide Web.

niveles, por todas las aplicaciones y soportes. Se usa para la creación de reglas básicas que facilitan el intercambio de información estructurada entre aplicaciones. [18]

1.6 Frameworks.

Un framework, en el desarrollo de software es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. [13] De acuerdo con lo planteado anteriormente la dirección del proyecto determinó emplear:

1.6.1 Sauxe 1.5.

Sauxe es el framework que se empleará para el desarrollo de la solución que se propone, el cual fue desarrollado en la UCI como fruto del paradigma de independencia tecnológica por el que aboga el país. Este framework, fusionado bajo tecnología totalmente libre (entre ellas PHP, Postgresql, Apache) posee el desarrollo de tecnologías propias basadas en otros frameworks como ZendFramework para el manejo de la lógica de negocio, Doctrine para el acceso a datos y ExtJS para la capa de presentación. Cuenta con una arquitectura en capas, que a su vez presenta en su capa superior un MVC[10] Contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. [19]

1.6.2 ExtJS 2.2.

ExtJS es una librería JavaScript que permite construir aplicaciones complejas en internet, además de flexibilizar el manejo de componentes de la página como el DOM, comunicación con el servidor usando AJAX, permite crear interfaces de usuario bastante funcionales que es principalmente para lo cual se emplea, posee numerosas funcionalidades que permiten añadir interactividad a las páginas HTML. Una de las grandes ventajas de utilizar ExtJS es que permite crear aplicaciones complejas utilizando componentes predefinidos. Se distribuye la carga de procesamiento Cliente-Servidor, permitiendo que el servidor pueda atender más clientes al mismo tiempo. [20]

[10] **MVC: Modelo Vista Controlador** es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

1.6.3 Zend Framework 1.9.7.

Zend Framework se utiliza para el manejo de la lógica de negocio. Este framework de código abierto, brinda facilidades de uso y poderosas funcionalidades. Está diseñado para la versión 5 de PHP y posee buenas capacidades de ampliación. Proporciona un sistema de caché de forma que se puedan almacenar diferentes datos, así como los componentes que forman la infraestructura del patrón MVC. Consta de mecanismos de filtrado y validación de entradas de datos. Permite convertir estructuras de datos PHP a JSON[11] y viceversa, para su utilización en aplicaciones AJAX y provee capacidades de búsqueda sobre documentos y contenidos. [21]

1.6.4 Doctrine 1.2.1.

Para la capa de acceso a datos se empleará Doctrine. Este es un potente y completo sistema ORM[12] para PHP 5.2 o superior que incorpora una capa de abstracción a base de datos (DBL). Uno de sus rasgos importantes es la habilidad de escribir opcionalmente las preguntas de la base de datos orientada a objeto; proporcionándoles una alternativa poderosa a diseñadores de SQL, manteniendo un máximo de flexibilidad sin requerir la duplicación del código innecesario. Además, exporta una base de datos existente a sus clases correspondientes y convierte clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos. [22]

1.7 Tecnologías y herramientas de desarrollo:

Para la realización de SIPAC fue necesario que cada uno de los equipos de desarrollo constara de un modelo estandarizado de las tecnologías y herramientas a utilizar, así como de sus versiones, para la implementación de las capas de presentación, negocio y acceso a datos. De acuerdo con lo planteado anteriormente la dirección del proyecto determinó emplear:

1.7.1 AJAX.

Ajax por sus siglas en inglés Asynchronous JavaScript And XML (JavaScript asíncrono y XML) es la técnica de desarrollo web que se usará para poder hacer consultas asíncronas al servidor sin necesidad

[11] **JSON**: es un formato ligero para el intercambio de datos. [43]

[12] **ORM(**Object Relational Mapper**)**: componente de software que permite trabajar con los datos persistidos como si fueran parte de una base de datos
orientada a objetos. [46]

de recargar la página. Surge de la combinación de tres tecnologías ya existentes: HTML (o XHTML) y Hojas de Estilo en Cascada (CSS) para presentar la información, Document Object Model (DOM) y JavaScript, para interactuar dinámicamente con los datos, además de XML para intercambiar y manipular datos de manera asíncrona con un servidor web. [23]

1.7.2 Herramienta CASE: Visual Paradigm 5.0.

Se empleará Visual Paradigm 5.0 como herramienta CASE. Utiliza UML 8.0 como lenguaje de modelado, con soporte multiplataforma y proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. 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, demostraciones interactivas y proyectos, ayudando a construir aplicaciones de calidad más rápido, mejor y a bajo costo. [24]

1.7.3 Entorno Integrado de Desarrollo: NetBeans 7.1.

La presente solución se desarrollará sobre el Entorno Integrado de Desarrollo (IDE) de programación multiplataforma NetBeans 7.1 el cual tiene soporte para la versión 5 de PHP, ExtJS, el diseño de Hojas de Estilo (CSS) y HTML. Se integra con varias herramientas como el Subversion y servidores web, presenta una gran estabilidad. Es un producto de código abierto, con todos los beneficios del programa disponible en forma gratuita. Hace uso de plugins para ampliar sus funcionalidades, lo que le da una gran facilidad de uso. [25]

1.7.4 Servidor web: Apache 2.0.

Se utilizará como servidor web Apache 2.0 pues es una tecnología gratuita de código abierto compatible con muchos Sistemas Operativos. Tiene todo el soporte que se necesita para tener páginas dinámicas. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Posibilita configurar la creación y gestión de registros de actividad. Apache permite la creación de ficheros de registro a la medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor. [26]

1.7.5 Sistema Gestor de Bases de Datos: PostgreSQL.

Como Sistema Gestor de Base de Datos (SGBD) se empleará la versión 8.3.3 de PostgreSQL. Este es un sistema de gestión de bases de datos relacional orientada a objetos. Es una herramienta de código abierto, de bajo coste y multiplataforma. Se destaca en ejecutar consultas complejas, subconsultas y uniones de gran tamaño. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Soporta transacciones, claves ajenas con comprobaciones de integridad referencial y almacenamiento de objetos de gran tamaño. Cuenta con varias herramientas gráficas de diseño y administración de bases de datos como el pgAdmin. [27]

1.7.6 Sistema de Control de Versiones: Subversion 1.6.5.

La versión 1.6.5 de Subversion (SVN) es la herramienta de entorno colaborativo que se utilizará para el control de versiones. Se encuentra preparado para funcionar en red y se distribuye bajo licencia libre. Mantiene versiones no solo de archivos, sino también de directorios y versiones de los metadatos asociados a esos directorios. Además de los cambios en el contenido de los documentos, se mantiene la historia de todas las operaciones de cada elemento, incluyendo la copia, cambio de directorio o de nombre. Proporciona una atomicidad de las actualizaciones, una lista de cambios constituye una única transacción o actualización del repositorio. Brinda facilidades de soporte tanto de ficheros de texto como binarios. Ofrece mejor uso del ancho de banda, ya que en las transacciones se transmiten solo las diferencias y no los archivos completos. [28]

1.8 Conclusiones del capítulo.

La gestión documental es un proceso fundamental para la recepción, seguimiento, uso y disposición de documentos de archivo, en este caso relacionados con la planificación. El capítulo que recién concluye fue esencial para valorar el estado actual de la realización de este proceso en el ámbito informático a partir de un análisis de sistemas tanto nacionales como internacionales, evidenciando la no existencia de un software que pudiera ser integrado a SIPAC y demostrándose la necesidad de desarrollar la funcionalidad Gestor documental para dicho sistema. Finalmente se caracterizan los lenguajes, tecnologías y herramientas que posibilitaran el desarrollo de la funcionalidad.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA SOLUCIÓN.

2.1 Introducción.

En el presente capítulo se realiza una descripción de la solución propuesta, para proporcionar un mejor entendimiento del sistema. Se describe y fundamenta la línea base de la arquitectura, se precisan los estilos y corrientes arquitectónicas adoptadas y la estructura de empaquetamiento sobre los cuales se concebirá la solución propuesta. Se especifican los requisitos funcionales que debe cumplir la funcionalidad, contribuyendo directamente en que la solución propuesta satisfaga las necesidades del cliente y los no funcionales que se refieren a las propiedades emergentes de dicha funcionalidad. Como parte de la modelación del diseño de la solución se fundamentan los patrones de diseños empleados en su elaboración y se crean los diagramas que según la notación UML describen las relaciones entre las clases del diseño, el modelo de datos y la comunicación con el resto de las funcionalidades de SIPAC.

2.2 Modelación del negocio.

2.2.1 Modelo conceptual.

Un modelo del dominio es una representación de las clases conceptuales del mundo real. Su objetivo es comprender y describir las clases más importantes dentro del contexto del sistema.

Actividades: conjunto de operaciones o tareas, propias de una persona o entidad, destinadas para cumplir determinado(s) objetivo(s).

Plan: es un modelo sistemático que se elabora antes de realizar una acción, con el propósito de dirigirla y encauzarla. En este sentido, un plan también es un documento que precisa los detalles necesarios para realizar una misión.

Objetivos: un objetivo es una meta o finalidad a cumplir para la que se disponen medios determinados. Los objetivos se pueden clasificar en estratégicos (generales y a largo plazo), por área (funcionales), individuales (cada sujeto que forme parte de la empresa).

FIP: factores que Influyen en el Plan. Documento que contiene elementos que desencadenan cambios en los planes, el mismo es emitido por el nivel superior o propio.

Elemento de la planificación: puede ser un plan, una actividad o un objetivo.

Anexo: es un fichero que se anexa a un elemento de la planificación, el mismo puede contener cualquier objeto digitalizado de los siguientes: documentos (PDF, EXCEL, DOC) o imágenes.

Documento: pueden ser archivos de extensión Doc, Excel y Pdf.

DOC: es una extensión de documentos.

PDF: es una extensión de documentos.

EXCEL: es una extensión de documentos.

Imagen: es la apariencia visible de las cosas. Representación visual de un objeto, una persona, un animal o cualquier otra cosa plausible de ser captada por el ojo humano.

JPG: es la extensión de un archivo.

La figura 2.1 representa el modelo conceptual de la funcionalidad Gestor documental.

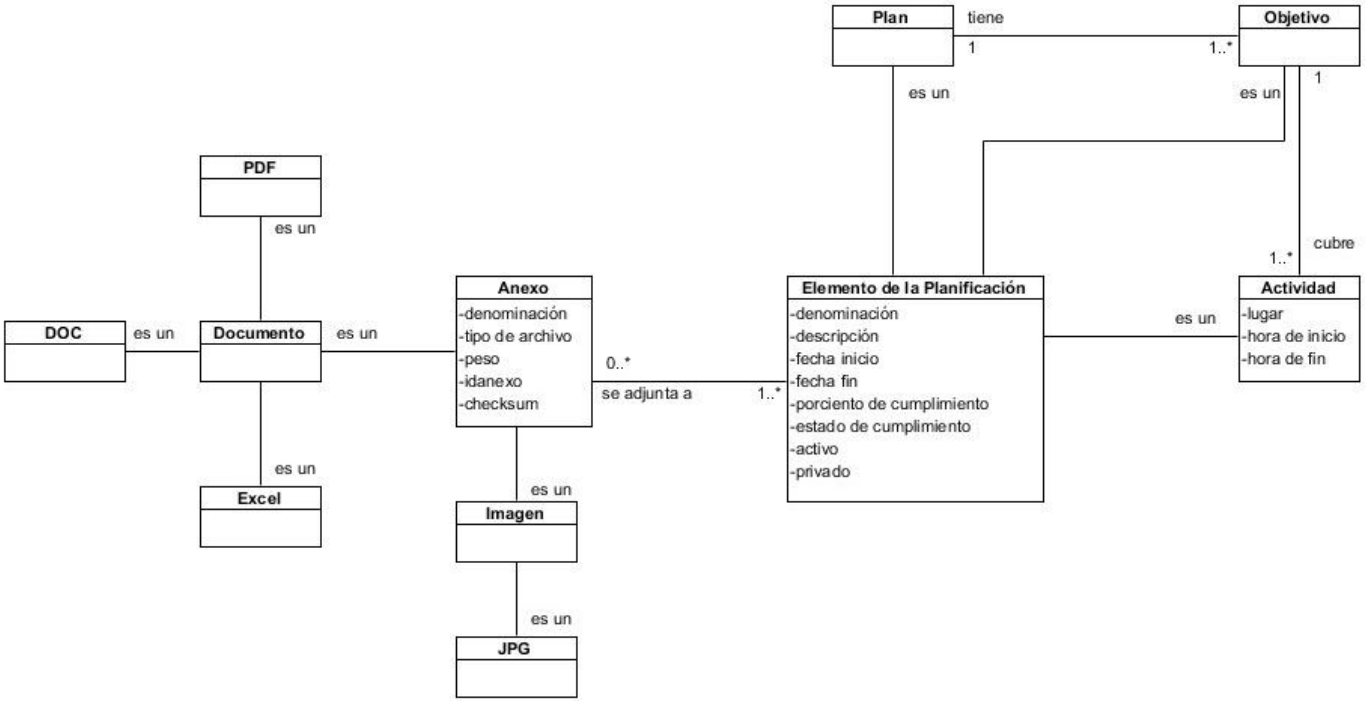


Figura 2 Modelo Conceptual.

2.2.2 Requisitos.

Un requisito de software puede ser definido como:

- Una capacidad del software necesaria por el usuario para resolver un problema o alcanzar un objetivo.
- Una capacidad del software que debe ser reunida o poseída por un sistema o componente del sistema para satisfacer un contrato, especificación, estándar, u otra documentación formal. [30]

2.2.3 Técnicas de captura de requisitos

La captura de requisitos es una actividad mediante la cual el equipo de desarrollo de un sistema de software extrae de cualquier fuente de información disponible, las necesidades que debe cubrir dicho sistema. [30]

Para extraer los requisitos del sistema se utilizaron las siguientes técnicas:

Entrevistas: es utilizada de manera frecuente en los diferentes encuentros con el cliente, dígase funcionales o especialistas de SIPAC. Se realizan preguntas con el objetivo de obtener toda la información posible sobre la visión que el entrevistado tiene de los requisitos y comprender los propósitos de la solución buscada.

Tormenta de ideas: se realizan reuniones y encuentros con todos los involucrados en el desarrollo del sistema donde cada cual expresa sus ideas y criterios. Su objetivo fundamental es dar una visión general de las necesidades del sistema.

Talleres: consiste en realizar reuniones con los involucrados en el desarrollo del sistema, para estos encuentros se debe realizar una preparación previa dirigida por un experto (en este caso el funcional o jefe del proyecto SIPAC). Su objetivo fundamental es detallar cada requisito y sirve como base para la posterior especificación de los mismos.

2.2.3 Técnicas de validación de requisitos

La validación de requisitos tiene como misión demostrar que la definición de los requisitos define realmente el sistema que el usuario necesita o el cliente desea. El proceso de validación de requisitos debe realizarse o de lo contrario se corre el riesgo de implementar una mala especificación, con el costo que eso conlleva.

Auditorías: la revisión de la documentación con esta técnica consiste en un chequeo de los resultados contra una lista de chequeo predefinida o definida a comienzos del proceso, es decir, solo una muestra es revisada.

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.

Para validar que los requisitos anteriormente identificados y descritos cumplan con las expectativas del cliente y el equipo de desarrollo, se emplea la Técnica de validación de requisitos: construcción de prototipos por cada requisito funcional. Además en el expediente de proyecto se incluye los artefactos Criterios para validar Requisitos del Cliente y Criterios para validar Requisitos del Producto con este mismo fin.

También es importante señalar que se realizaron diversas revisiones, por el administrador de calidad y la analista principal de la línea de Planificación, a los artefactos relacionados con la disciplina de Levantamiento de requisitos para lograr una correcta interpretación de la información transmitida, los señalamientos planteados fueron recogidos y aplicados posteriormente.

2.2.4 Listado de Requisitos Funcionales Identificados.

Requisitos funcionales: son declaraciones de los servicios que proveerá el sistema, de la manera en que estos reaccionaran a entradas particulares. En algunos casos, los requisitos funcionales de los sistemas también declaran explícitamente lo que el sistema no debe hacer. [31]

RF1 Adjuntar anexo a elemento de la planificación.

RF2 Eliminar anexo a elemento de la planificación.

RF3 Descargar anexo a elemento de la planificación.

RF4 Buscar anexo a elemento de la planificación.

RF5 Listar anexos a elemento de la planificación.

RF6 Filtrar anexo a elemento de la planificación por extensión.

2.2.5 Especificación de requisitos funcionales

Tabla 1: Especificación de requisito Adjuntar anexo a elemento de la planificación

Precondiciones	El elemento de la planificación ha sido creado.
Flujo de eventos	
Flujo básico Adjuntar anexo a elemento de la planificación	
1	El usuario accede a la pantalla principal de documentos anexos y selecciona la opción Adjuntar.
2	El sistema muestra una interfaz que permite al usuario seleccionar en su PC el archivo que desea adjuntar.
3	El usuario selecciona el documento que desea adjuntar y lo adjunta.
4	El sistema muestra el archivo adjunto.
5	Se concluye el requisito.
Pos-condiciones	
	El archivo fue anexado correctamente al elemento de la planificación.
Flujos alternativos	
Flujo alternativo 3.a El usuario selecciona el documento que desea adjuntar y cancela	
	Ir al punto 1 del Flujo Básico.

Ver Anexos:

Anexo 1: Especificación de requisito Eliminar Anexo a elemento de la planificación.

Anexo 2: Especificación de requisito Descargar Anexo a elemento de la planificación.

Anexo 3: Especificación de requisito Buscar Anexo a elemento de la planificación.

Anexo 4: Especificación de requisito Listar Anexo a elemento de la planificación.

Anexo 5: Especificación de requisito Filtrar Anexo a elemento de la planificación.

2.2.6 Requisitos no funcionales.

Los requisitos no funcionales definen propiedades y restricciones del sistema, estos pueden ser por ejemplo confiabilidad, tiempo de respuesta y requisitos de almacenamientos [32]. Los requisitos no funcionales de la solución se rigen por los definidos en la arquitectura del sistema que se encuentran en el artefacto CIG-SPA-N-i3514-RNF [33] perteneciente al expediente del proyecto SIPAC 2.0.

Los requisitos no funcionales de la funcionalidad Gestor documental: se encuentran descritos en el expediente de proyecto SIPAC 2.0 en la siguiente dirección: 1. Ingeniería\ 1.1 requisitos\Descripciones de requisitos\04-Gestor documental.

Usabilidad:

- **Tipo de usuario final:**

Tabla 2: Tipo de usuario final.

No.	Sexo	Edad	Nivel de Escolaridad	Ocupación	Experiencia profesional	Experiencia con la Aplicación Informática.	Tipo de discapacidad
1	M	18-60 años	Técnico Medio	Especialista de planificación	1-20 años	Web y escritorio	Ninguna
2	F	18-60 años	Técnico Medio	Especialista de planificación	1-20 años	Web y escritorio	Ninguna
3	M	25-60 años	Universitario	Directivo	5-20 años	Web y escritorio	Ninguna
4	F	25-60 años	Universitario	Directivo	5-20 años	Web y escritorio	Ninguna

- **Ambiente:**

RnF 1 El sistema requiere como componentes de software del lado cliente Navegador Mozilla Firefox versión 2.2.

RnF 2 Del lado del servidor de Aplicaciones requiere Soporte para PHP5, un servidor web apache versión 2.0 y sistema operativo Ubuntu Server.

RnF 3 Del lado del cliente de BD requiere Ubuntu Server y PostgreSQL versión 8.3.

RnF 4 Los componentes de hardware que se necesitan para el correcto funcionamiento de la aplicación son por el lado cliente: Procesador 1.40 GHZ, RAM: 256 MB (recomendado 512 Mb), Tarjeta de Red: 1

RnF 5 Los componentes de hardware que se necesitan para el correcto funcionamiento de la aplicación por parte del servidor de aplicaciones son: Tarjeta de Red: 1, Procesador: 3.00 GHZ, RAM: 1GB, Disco duro: 160 GB, UPS: 1, Lector de CD: 1.

RnF 6 Los componentes de hardware que se necesitan para el correcto funcionamiento de la aplicación por parte del servidor de Base de datos son: Tarjeta de Red: 1, Procesador: 3.00 GHZ, RAM: 1GB, Procesador: 3.00 GHZ, RAM: 1GB, Disco duro: 160 GB,UPS: 1, Lector de CD: 1

Restricciones de diseño

RnF 7 El sistema se implementará teniendo en cuenta los estándares definidos en el Modelo de desarrollo de CEIGE.

Requisitos para la documentación de usuarios en línea y ayuda del sistema.

RnF 8 El sistema debe actualizar su manual de usuario con la inclusión de las nuevas funcionalidades desarrolladas en esta versión.

Interfaz

RnF 9 Las interfaces de usuario de la funcionalidad deben cumplir con los principios de accesibilidad, usabilidad y operatividad del sistema.

Estándares Aplicables

RnF 10 La aplicación debe ser implementada siguiendo los estándares de documentación, implementación de interfaz de usuario e implementación de lógica de negocio que han sido

diseñados para el proceso de desarrollo de software del ERP y la UCID

2.3 Diseño de la solución.

2.3.1 Diseño de la solución en términos de componentes.

Todas las funcionalidades capturadas y modeladas en las disciplinas de negocio y requisitos quedan expresadas o contenidas en al menos un componente, y las distintas interacciones entre estos componentes originan funcionalmente la existencia de subsistemas.

De esta forma el sistema queda constituido por un conjunto de componentes que responden a un cúmulo de funcionalidades, un grupo de interacciones entre estos componentes respondiendo a las distintas integraciones y dependencias originadas en el negocio, estos componentes están agrupados en una unidad mayor denominada subsistemas que responden a las áreas de procesos más generales identificadas en el negocio.

Cada funcionalidad por su parte, puede abarcar o no, un conjunto de requisitos funcionales, los cuales convergen en un solo controlador de eventos.

La Especificación de componentes se puede encontrar descrita en el expediente de proyecto SIPAC 2.0 en la siguiente dirección: 1. Ingeniería\1.2 arquitectura y diseño\Arquitectura de Software\Análisis de componentes.

A continuación se presenta el Mapa de componentes que responde al funcionamiento de SIPAC, el cual está estructurado de acuerdo a los diferentes niveles de empaquetamiento de la Arquitectura Base: subsistema, componente, funcionalidad y requisito; punto de partida de la solución para la Gestión documental.

Especificación de los niveles de empaquetamiento subsistema y componente:

Subsistema Planificación: permite el registro, seguimiento y control de la Planeación Estratégica y Operativa para todos los niveles organizacionales. Dicho subsistema contiene los componentes:

- *Planeación*: permite la gestión de los elementos de la planificación: Planes, Actividades, Objetivos, Factores que Influyen en Plan (FIP) y Áreas de Resultados Clave (ARC).
- *Configuración*: brinda las funcionalidades para crear grupos de usuarios a partir de los usuarios sobre los cuales se puede planificar, teniendo en cuenta el dominio de acceso y la estructura definida configurados en los subsistemas Seguridad y Estructura respectivamente.
- *Notificaciones*: permite la generación de notificaciones a partir de las acciones que se realizan sobre los diferentes elementos.

Subsistema_Configuración: brinda una serie de funcionalidades que deben ser ejecutadas antes de comenzar a utilizar el resto de los subsistemas, específicamente para el funcionamiento del sistema SIPAC es necesario definir el flujo de información de los elementos de la planificación a partir de estados de los documentos y las transiciones, el cual es posible a través del componente: WorkFlow.

Ver Figura 3: Mapa general de Componentes.

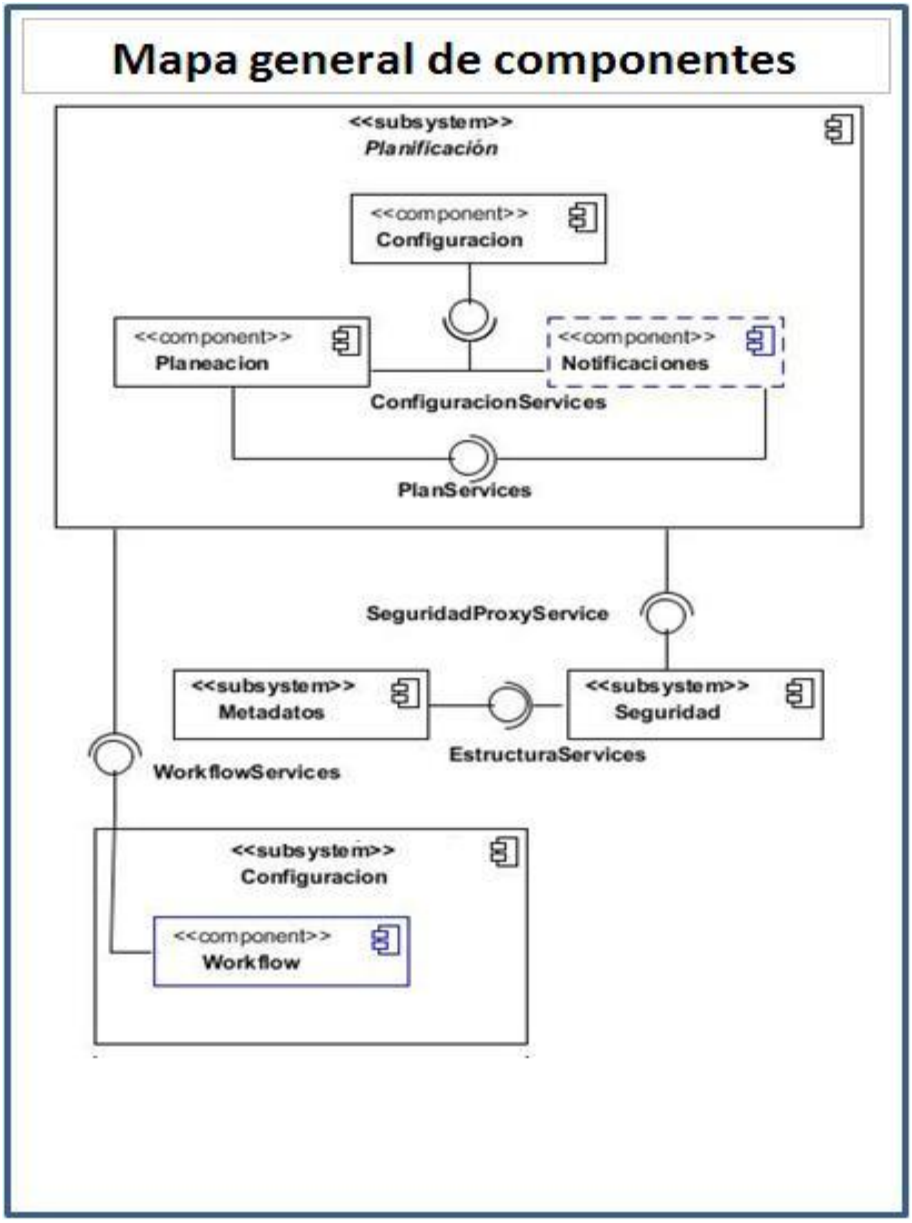


Figura 3 Mapa general de Componentes.

Especificación de los niveles de empaquetamiento funcionalidades y requisitos propios del componente Planeación:

Solo se representan las funcionalidades del componente Planeación que se relacionan con la solución para la Gestión documental en el sistema SIPAC, en este caso: Gestionar Plan, Gestionar Objetivo,

Gestionar Actividades, Gestionar FIP y Gestor documental, que a su vez incluye los requisitos que fueron especificados en la sección: 2.2.4 listado de requisitos funcionales identificados.

Ver Figura 4 Componente Planeación.

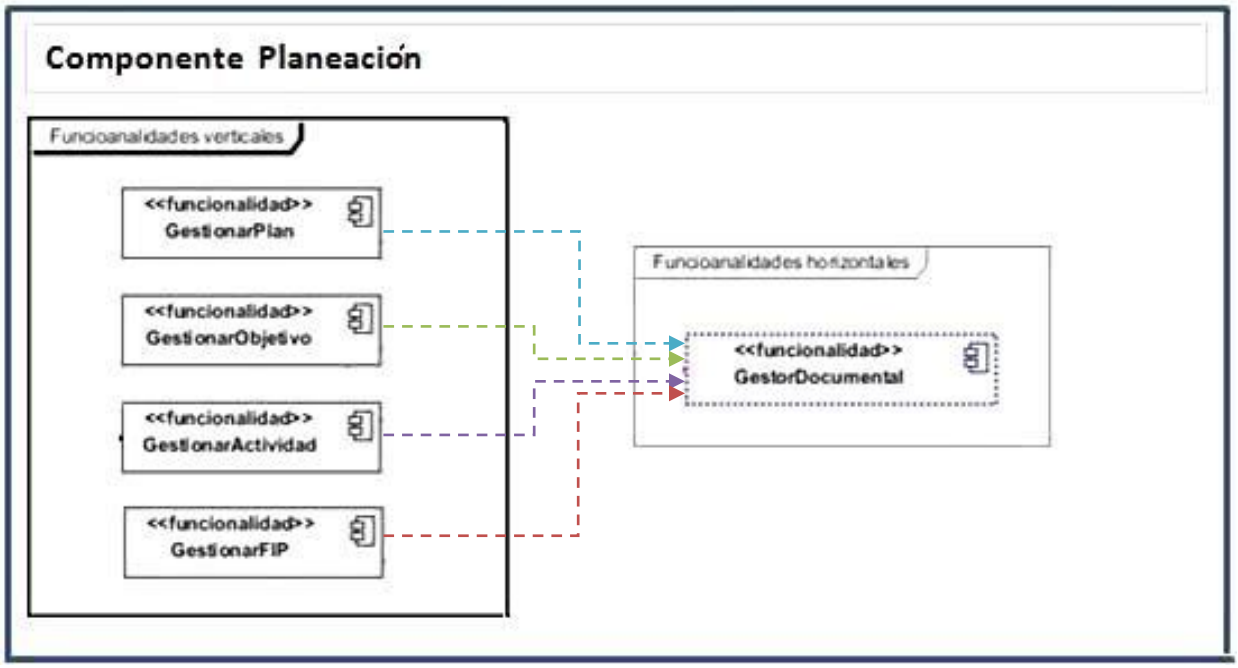


Figura 4 Componente Planeación.

2.3.2 Diseño de clases.

Para la transición a la fase de implementación es de gran importancia tener definidas las clases que intervendrán y las relaciones entre ellas. Es por esta razón que a continuación se presenta el diagrama de clases donde se representan las clases de la funcionalidad y sus interrelaciones.

2.3.2.1 Diagramas de clases de diseño.

El diseño de la solución para la gestión documental toma como base la arquitectura de SIPAC que implementa como patrón arquitectónico MVC, que controla el flujo de datos entre la interfaz de usuario y la lógica de negocio. Donde la vista se compone de las clases phtml y JS pertenecientes a la solución. El

controlador es el encargado de recibir, interpretar y mandar los datos entre la vista y el modelo. Por último la capa del modelo que es donde se enviar toda la lógica del negocio y el acceso a datos.

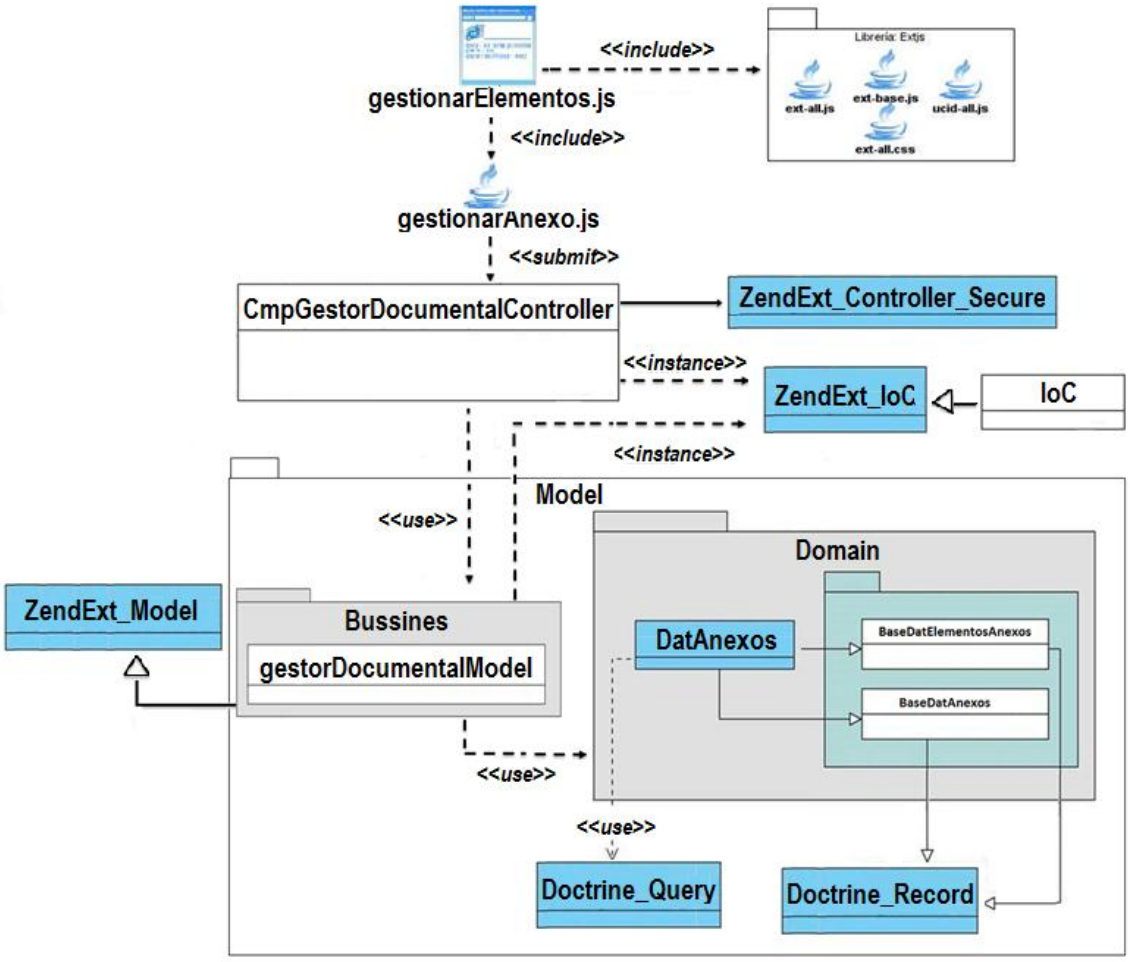


Figura 5Diagrama de Clases del diseño de la funcionalidad Gestor documental.

2.3.2.2 Diagrama de secuencia.

Un diagrama de secuencia es una forma de diagrama de interacción que muestra los objetos como líneas de vida a lo largo de la página y con sus interacciones en el tiempo representadas como mensajes dibujados como flechas desde la línea de vida origen hasta la línea de vida destino. Los diagramas de secuencia son buenos para mostrar qué objetos se comunican con qué otros objetos y qué mensajes

disparan esas comunicaciones. Los diagramas de secuencia no están pensados para mostrar lógicas de procedimientos complejos. [37]

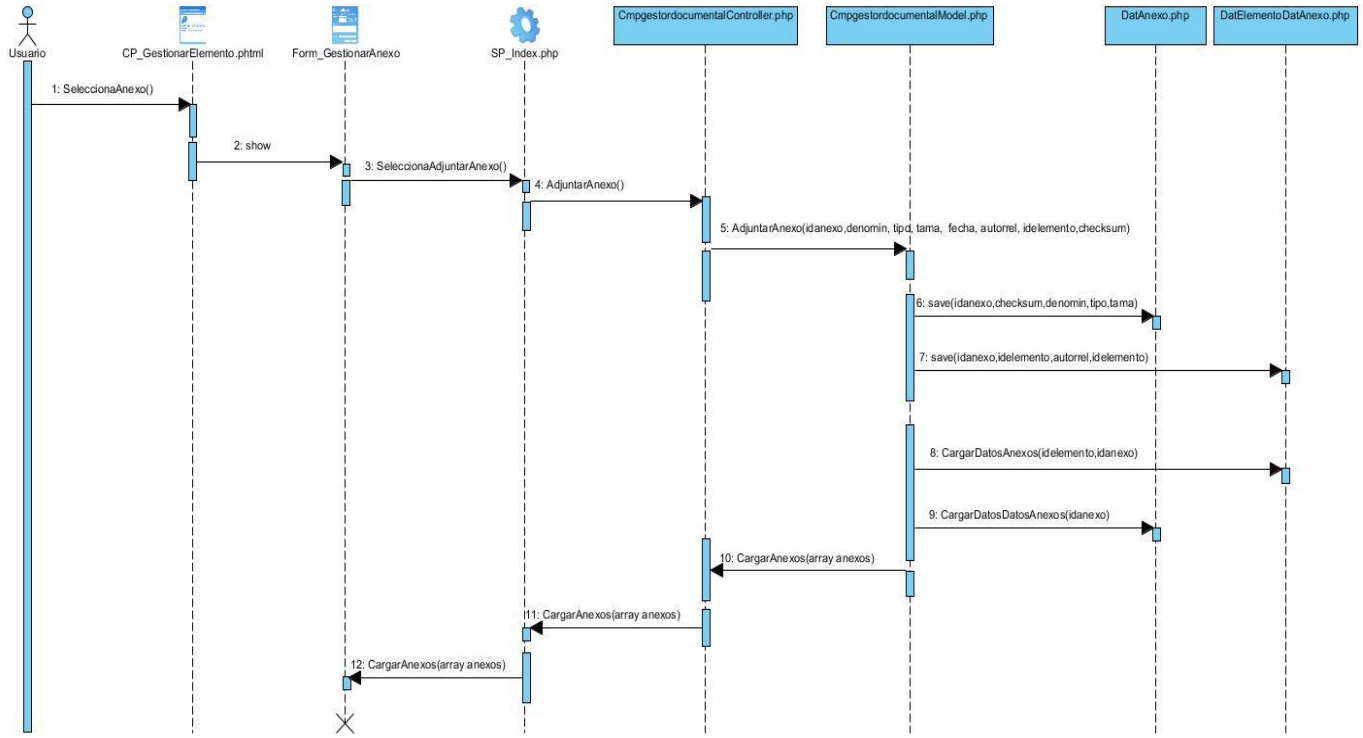


Figura 6 Diagrama de secuencia Adjuntar anexos.

Ver Anexos:

Anexo 6: Diagrama de secuencia Descargar anexos.

Anexo 7: Diagrama de secuencia Eliminar anexos.

Anexo 8: Diagrama de secuencia Buscar anexos.

Anexo 9: Diagrama de secuencia Filtrar anexos.

Anexo 10: Diagrama de secuencia Listar anexos.

2.3.2.3 Modelos de datos.

Un modelo de datos es un lenguaje utilizado para la descripción de una Base de Datos. Por lo general, permite describir las estructuras de datos de la base, las restricciones de integridad y las operaciones de manipulación de los datos. [38]

El modelo de datos de SIPAC resume los conceptos y define las relaciones, ajustándose a las necesidades de almacenamiento de datos del sistema. Posee tercera forma normal y cuenta con 52 tablas. En el expediente de proyecto SIPAC 2.0 se encuentra el modelo de datos en cuestión y en la Figura 7 se detallan las tablas encargadas de almacenar los datos correspondientes a la funcionalidad Gestor documental como son dat_elementos_dat_anexos, dat_anexos y dat_elementos.

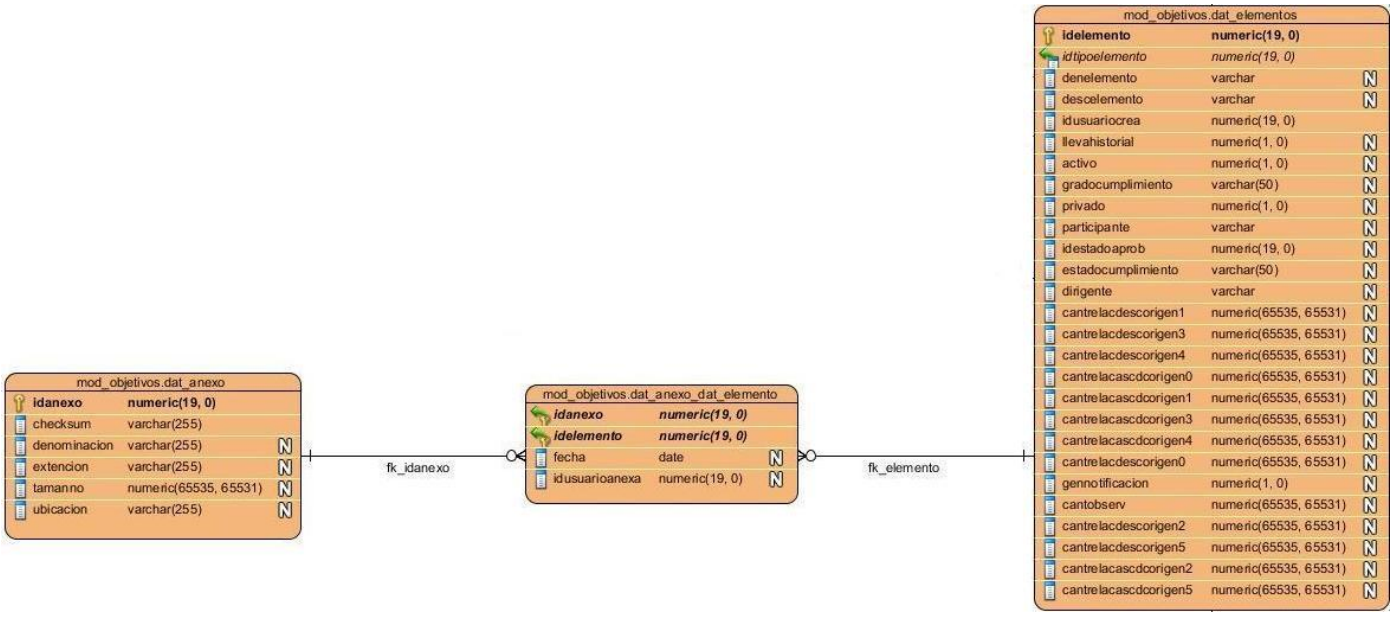


Figura 7 Modelos de Datos de la funcionalidad Gestor documental.

Teniendo en cuenta los requisitos funcionales identificados se crean las tablas dat_anexos y dat_elementos_dat_anexos; en la primera se almacenan todos los datos de los anexos junto con los atributos idanexo y checksum, este último a través de la funcionalidad de php md5_file se encarga de generar un número que se basa en el contenido del fichero, es decir, si por cualquier causa se modifica dicho fichero el checksum será diferente, de esta manera se logra que no se adjunte el mismo archivo dos veces. Mientras que en la tabla dat_elementos_dat_anexos se almacena la relación de los anexos con los

elementos junto con el usuario que adjuntó el anexo y la fecha en que se creó el mismo. Finalmente en la tabla dat_elementos se encuentran los elementos de la planificación a los cuales se les adjuntará los documentos.

2.4 Patrones de diseño empleados en la solución propuesta.

Patrones de diseño:

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

2.4.1 Patrones GRASP.

GRASP Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones.

GRASP es un acrónimo que significa General Responsibility Assignment Software Patterns. El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos. [34]

Patrón Experto en información

El patrón experto en información es el principio básico de asignación de responsabilidades. Indica la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo se obtendrá un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento). [35] Por ejemplo la clase DatAnexos que es la encargada de realizar la funcionalidad de anexar los documentos.

Patrón Creador

El patrón creador ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases.
La nueva instancia deberá ser creada por la clase que tiene la información necesaria para realizar la creación del objeto, usa directamente las instancias creadas del objeto, almacena o maneja varias instancias de la clase y contiene o agrega la clase. [35]

Una de las consecuencias de usar este patrón es la visibilidad entre la clase creada y la clase creador. Una ventaja es el bajo acoplamiento, lo cual supone facilidad de mantenimiento y reutilización. La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia es útil contar con un principio general para la asignación de las responsabilidades de creación. Si se asignan bien el diseño puede soportar un bajo acoplamiento, mayor claridad, encapsulación y reutilización. [35] Este patrón se evidencia en la clase CmpgestordocumentalModel, la cual es la responsable de crear instancias de la clase DatAnexos, para así utilizar sus funciones.

Patrón Controlador

El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado.

Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control. [35]

Se recomienda dividir los eventos del sistema en el mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento. La clase CmpgestordocumentalController es un ejemplo de la aplicación de este patrón.

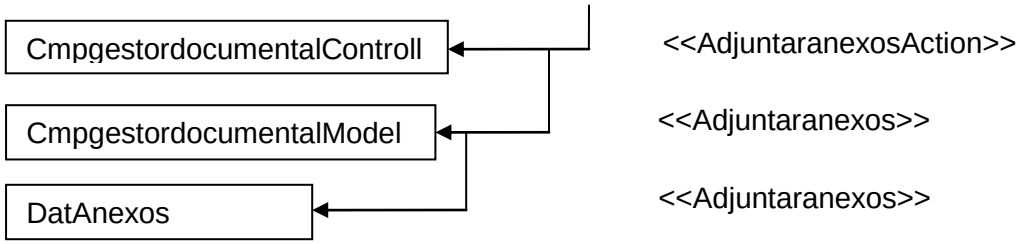
Patrón Alta cohesión

Los conceptos de cohesión y acoplamiento están íntimamente relacionados. Un mayor grado de cohesión implica uno menor de acoplamiento. Maximizar el nivel de cohesión intramodular en todo el sistema resulta en una minimización del acoplamiento intermodular. [35]

Este patrón dice que la información que almacena una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la clase. [35] De esta forma CmpgestordocumentalModel solamente se encarga de Adjuntar, cargar y eliminar anexos, así como otras operaciones que se realizan sobre esta área y no de otro tipo de operación.

Patrón Bajo acoplamiento

Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma 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 ellas.



Se puede evidenciar que solo se le atribuye la responsabilidad de crear objetos de DatAnexos a la clase CmpgestordocumentalModel evitando así que el número de recurrencias de la clase CmpgestordocumentalController a otras clases sea mayor.

2.4.2 Patrones GoF.

GOF: of Four o Banda de Cuatro, llamados así por los cuatro autores del libro Patrones de Diseño que recoge alrededor de 23 patrones de los más utilizados. [35]

Fachada:

Este patrón proporciona una interfaz unificada de alto nivel para un subsistema, que oculta las interfaces de bajo nivel de las clases que lo implementan simplificando así la interacción con el subsistema. Se utiliza para proporcionar un fácil acceso a subsistemas complejos. [36] Este se usa fundamentalmente en los servicios, donde la relación existente entre las clases controladoras y los servicios, permite acceder a métodos que no están implementados en el componente donde se encuentra la funcionalidad y que se encuentran, tanto en otros componentes pertenecientes a SIPAC, como en otros subsistemas externos. En el caso específico de la aplicación, a través del fichero de configuración Ioc se utilizan métodos de la clase SeguridadService que es considerada la fachada del subsistema seguridad.

Cadena de Responsabilidades:

El patrón Cadena de Responsabilidades permite establecer la línea que deben llevar los mensajes para que los objetos realicen la tarea indicada. Su aplicación principal en el diseño del sistema es en el

tratamiento de Excepciones. [35] Por ejemplo en el caso del método Adjuntar Anexo ante la ocurrencia de un error, desde la clase Model se envía el valor 0 que representa que no se pudo concretar la acción, dicha excepción debe ser propagada al Controlador, el cual será el encargado de capturarla y enviarla a la Vista. Esta última por su parte mostrará un mensaje al usuario en un lenguaje entendible notificando el error y sin especificar detalles del mismo, distribuyéndose de esta manera las responsabilidades entre las diferentes clases.

2.5 Conclusiones del capítulo

Durante el desarrollo del presente capítulo se obtuvieron los artefactos relacionados con el análisis y diseño de la solución propuesta. Se especificaron los requisitos para definir cómo implementar el sistema, creando así el punto de partida para las actividades de implementación. Se mostraron patrones de diseño que ofrecen un código con ventajas de reutilización y facilidad de comprensión y modificación. El diseño de la solución en términos de componentes y la estrategia de integración permitieron establecer las bases por las cuales se regirá la implementación de la solución, por lo que se puede concluir que es posible dar comienzo a la construcción de la funcionalidad.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DE LA SOLUCIÓN.

3.1 Introducción.

En el presente capítulo se exponen los estándares que regirán el código fuente a implementar. Se definen las pruebas que se le deben realizar a la funcionalidad propuesta para detectar la existencia de errores y se mostrarán las métricas aplicadas al diseño propuesto en el capítulo anterior, en aras de garantizar calidad de dicha solución.

3.2 Implementación.

Con el resultado obtenido del diseño comienza la implementación. Su principal propósito es lograr el desarrollo de la arquitectura y del sistema como un todo. De forma más específica, los propósitos de la Implementación son:

-Planificar las integraciones del sistema necesarias en cada iteración. Siguiendo para ello un enfoque incremental.

-Implementar clases, componentes y subsistemas encontrados durante el diseño.

-Integrar componentes. [39]

3.2.1 Estándares de código.

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. Un estándar de codificación completo comprende todos los aspectos de la generación de código, reflejando un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Con la utilización de estándares se logra mayor mantenibilidad del código proporcionando facilidades en el sistema de software para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento; logrando de esta manera alta calidad y buen rendimiento del software. Se definen tres partes fundamentales dentro de un estándar de programación:

Convención de nomenclatura: define cómo nombrar variables, funciones y clases.

Convenciones de legibilidad de código: es la forma de organizar el código y lograr que independientemente de quien desarrolle se entienda como un todo.

Convenciones de documentación: define cómo establecer comentarios, archivos de ayuda, entre otros.

Nomenclatura de las clases.

Los nombres de las clases deben comenzar con mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing, la cual define que los identificadores y nombres de variables, métodos y clases que están compuestos por múltiples palabras juntas, inicia cada palabra con letra mayúscula y sin usar ningún artículo posibilitando que con solo leer el nombre de la clase ya se reconozca la función de la misma.

1. Nomenclatura según el tipo de clases.

Clases controladoras: las clases controladoras después del nombre llevan la palabra “Controller”.

Ejemplo: CmpgestordocumentalController.

Clases de los modelos:

Bussines (Negocio): las clases que se encuentran dentro del paquete Bussines después del nombre llevan la palabra “Model”. Ejemplo: CmpgestordocumentalModel.

Domain (Dominio): Las clases que se encuentran dentro de Domain reciben el nombre de las tablas de la base de datos. Ejemplo: “DatAnexos”.

Generated (Dominio base): el nombre de las clases que se encuentran dentro de Generated comienza con la palabra: “Base” y seguido el nombre de la tabla en la base de datos. Ejemplo: BaseDatAnexos.

2. Nomenclatura de las funcionalidades:

El nombre a emplear para las funciones se escribe con la inicial del identificador en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing que es similar a la PascalCasing con la excepción de la primera letra.

Nomenclatura según la clase donde se encuentren las funciones:

En la clase controladora: las principales funcionalidades de las clases controladoras se les pone el nombre y seguida la palabra:"Action". Ejemplo: eliminarAnexoAction().

En las clases de los modelos: las funcionalidades se nombran de manera que al leerlo se identifique su propósito. Ejemplo de función: eliminarAnexo().

3. Nomenclatura de los comentarios:

Los comentarios deben ser lo bastante claros y precisos de forma tal que se entienda el propósito de lo que se está desarrollando. En caso de ser una función complicada se debe comentar para lograr una mejor comprensión del código.

3.2.2 Interfaces de usuario de la funcionalidad Gestor documental

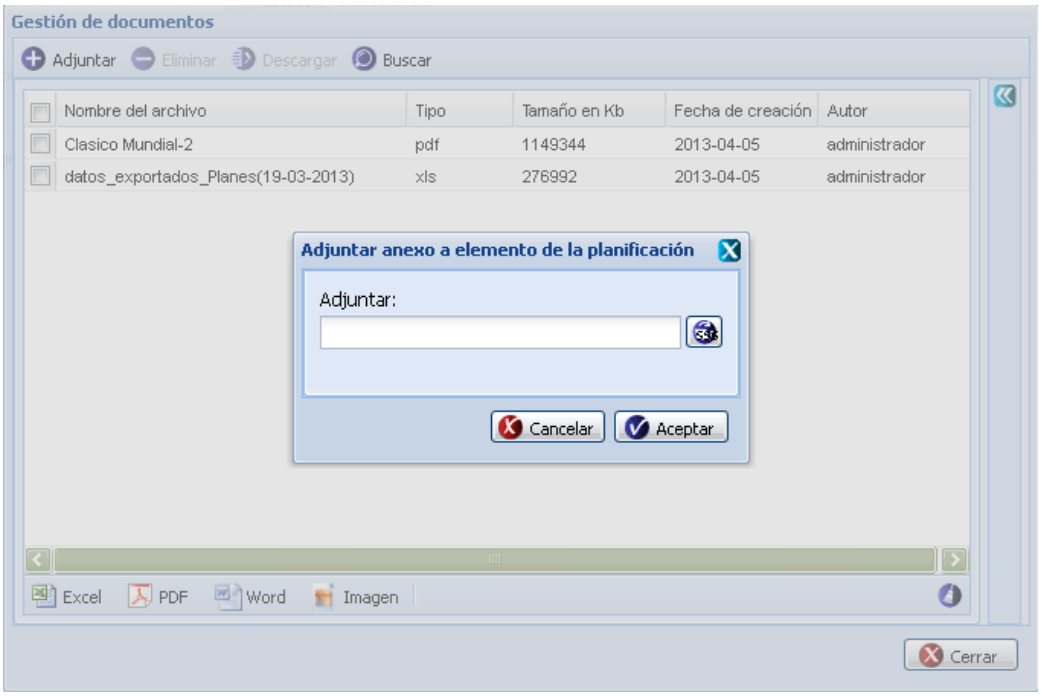


Figura 8 Adjuntar Anexo a elemento de la planificación.

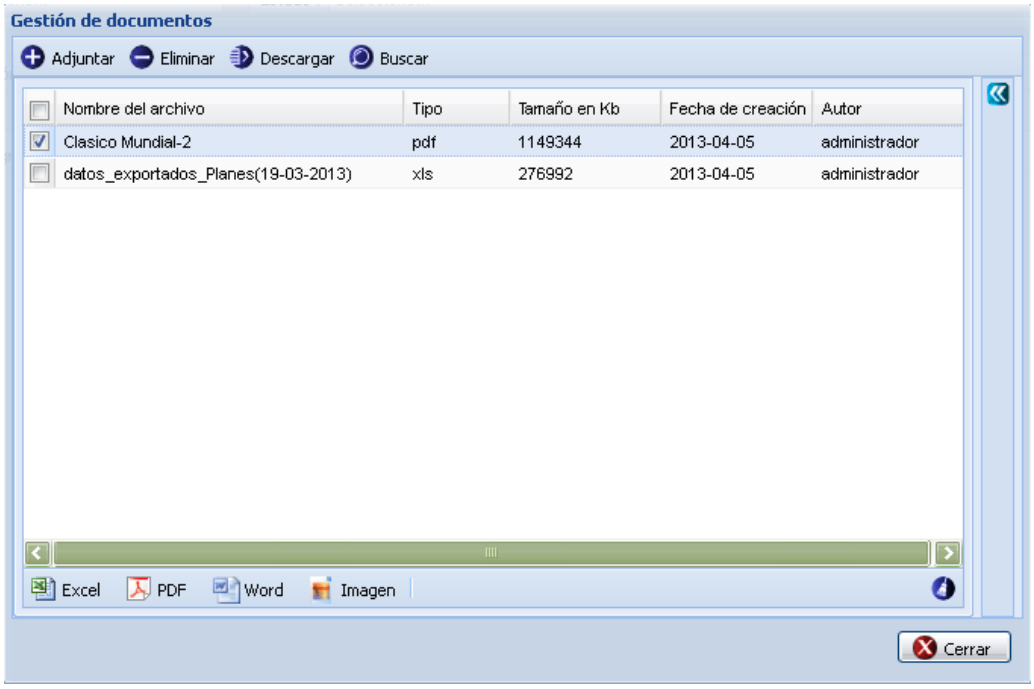


Figura 9 Listar Anexo a elemento de la planificación.

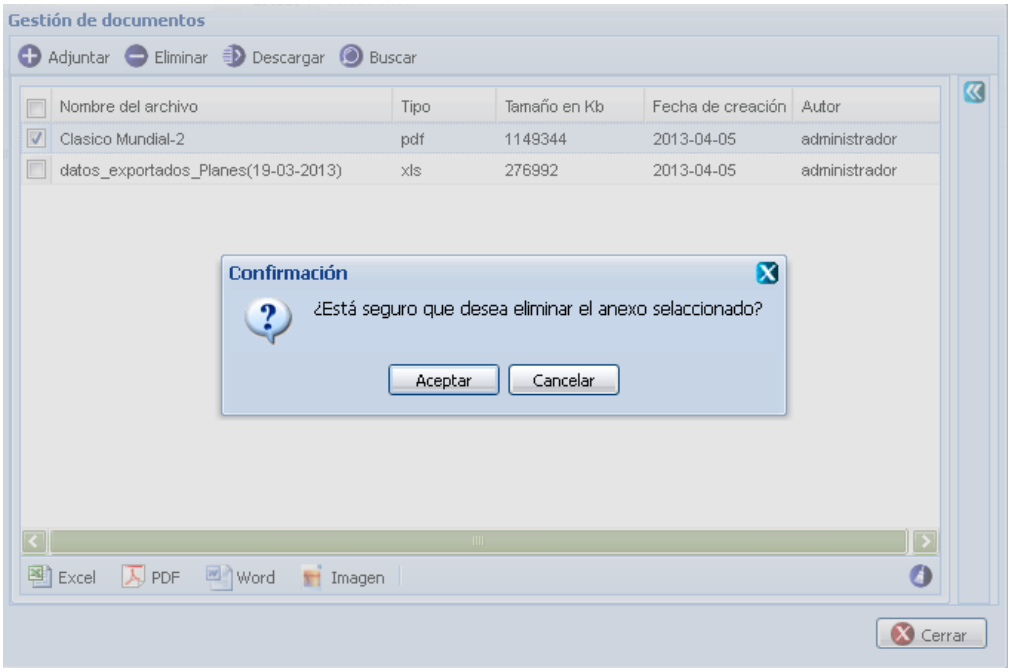


Figura 10 Eliminar Anexo a elemento de la planificación.

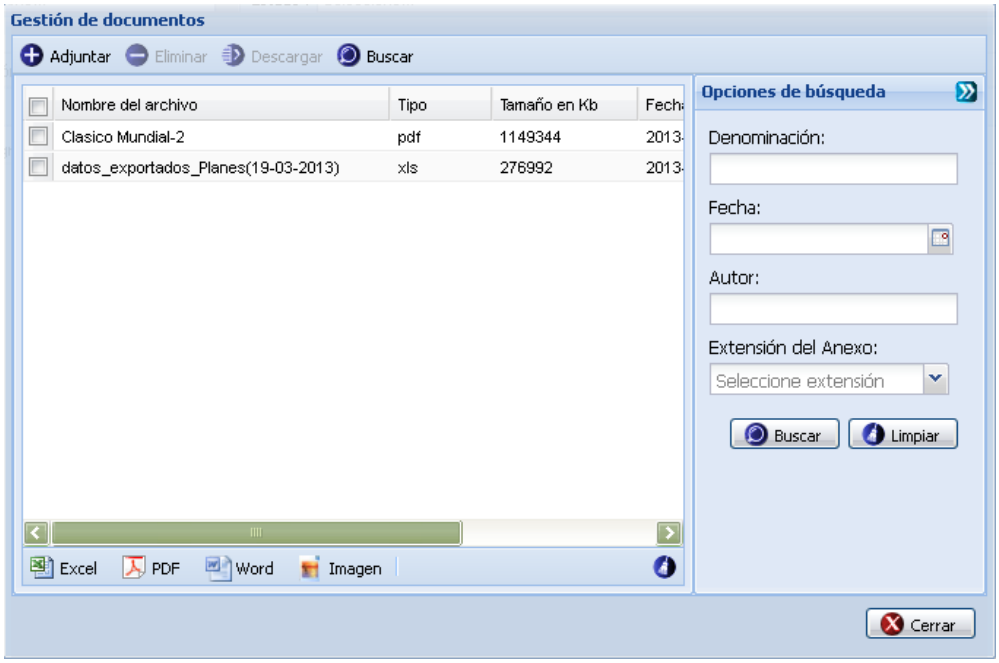


Figura 11 Buscar Anexo a elemento de la planificación.

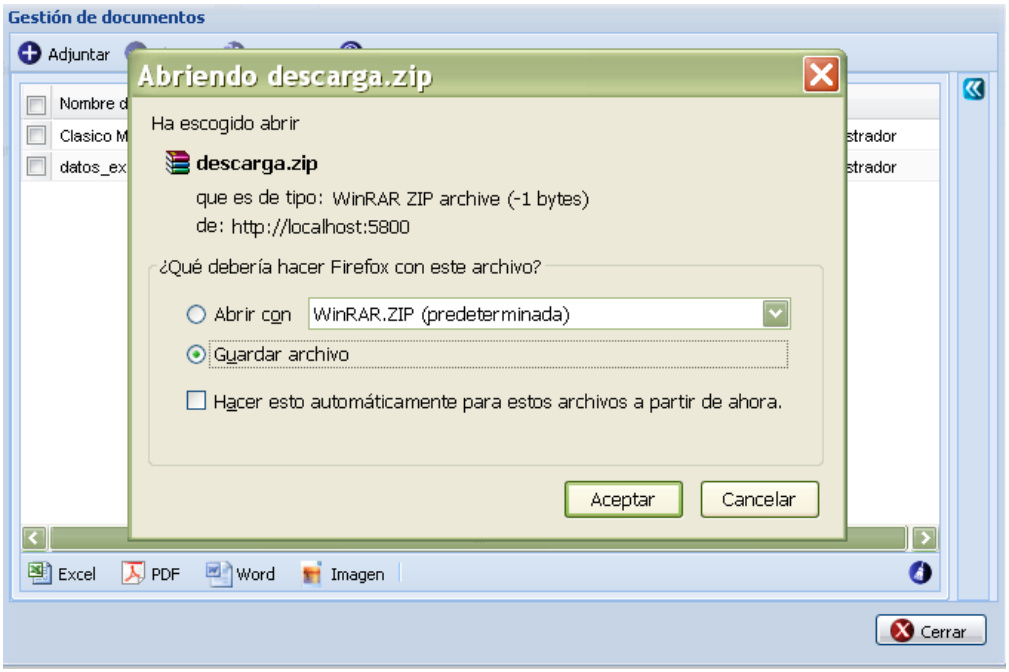


Figura 12 Descargar Anexo a elemento de la planificación.

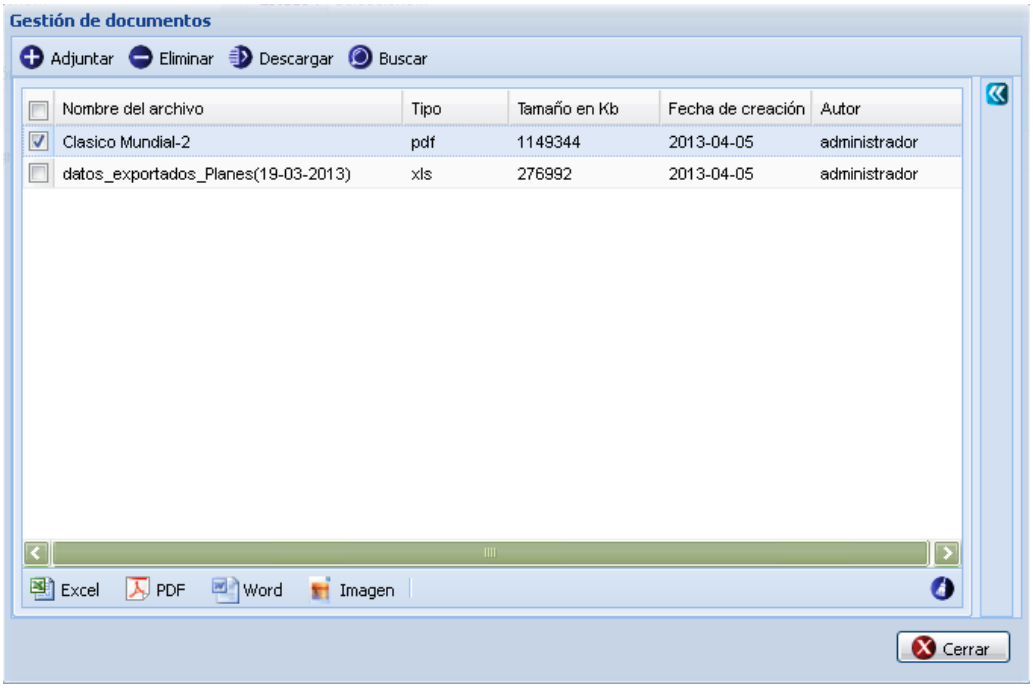


Figura 13 Filtrar Anexo a elemento de la planificación por extensión.

3.3 Validación de la solución propuesta.

Con la validación de la solución propuesta se establece una forma objetiva de verificar si el problema ha quedado efectivamente resuelto. A continuación se exponen los procedimientos empleados para detectar y corregir el máximo de errores posibles, primeramente se presentan las métricas del diseño que permiten juzgar la calidad de este y luego se describen las pruebas de aplicación realizadas.

3.3.1 Validación del diseño propuesto.

Las métricas permiten medir de forma cuantitativa la calidad de los atributos internos del software. Esto permite al ingeniero evaluar la calidad durante el desarrollo del sistema. A continuación se presentarán las métricas Tamaño Operacional de Clase (TOC) dada por el número de métodos asignados a una clase y Relaciones entre Clases (RC) dada por el número de relaciones de uso de una clase con otra. [40]

Las métricas TOC y RC incluyen medidas de los siguientes atributos de calidad:

Responsabilidad: responsabilidad que posee una clase en un marco conceptual correspondiente al modelado de la solución propuesta.

Complejidad del mantenimiento: nivel de esfuerzo necesario para sustentar, mejorar o corregir el diseño de software propuesto. Puede influir significativamente en los costes y la planificación del proyecto.

Complejidad de implementación: grado de dificultad que tiene implementar un diseño de clases determinado.

Reutilización: significa cuán reutilizada es una clase o estructura de clase dentro de un diseño de software.

Acoplamiento: dependencia o interconexión de una clase o estructura de clase respecto a otras.

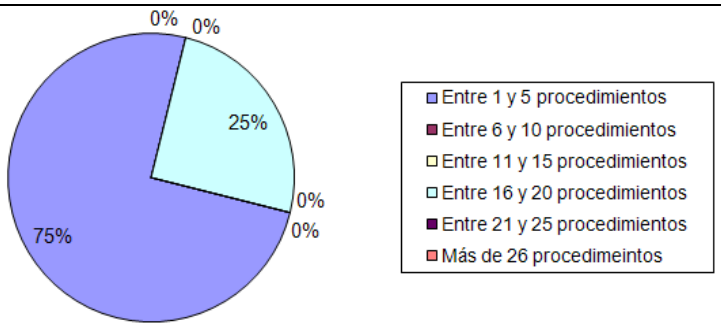
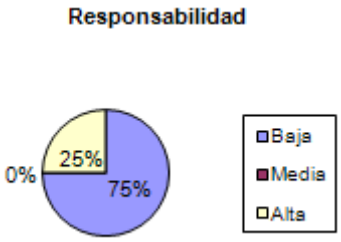
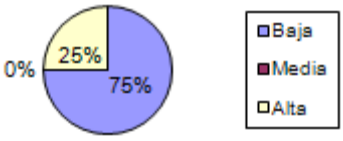
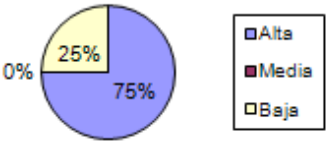
Cantidad de pruebas: número o grado de esfuerzo necesario para realizar las pruebas de calidad al producto diseñado. [51]

Métrica TOC: Tamaño operacional de clase.

Se refiere al número de procedimientos existentes en una clase. Determina una relación directa entre los atributos Responsabilidad y Complejidad de implementación, sin embargo establece una relación inversa entre estos últimos y el atributo Reutilización.

Como resultado de la evaluación de la métrica TOC se obtuvo lo siguiente:

Tabla 3: Métrica Tamaño Operacional de Clases (TOC)

Tamaño Operacional de Clases (TOC).	
	
Atributo que evalúa: Responsabilidad: Un aumento del TOC provoca un aumento de la responsabilidad asignada a la clase.	
Atributo que evalúa: Complejidad de implementación: El aumento del TOC provoca un aumento de la Complejidad de implementación.	
Atributo que evalúa: Reutilización: Un aumento del TOC provoca una disminución en el grado de reutilización de la clase.	

Ver Anexo:

Anexo 11: Instrumento de medición de la métrica Tamaño Operacional de Clase (TOC).

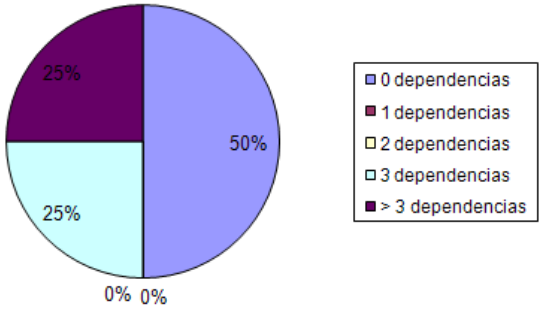
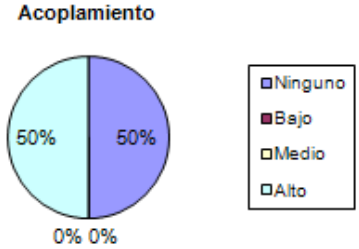
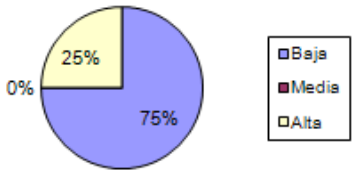
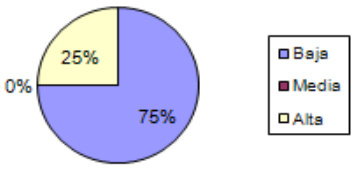
Después de analizar los resultados alcanzados en la aplicación de la métrica Tamaño Operacional de Clase (TOC), se llega a la conclusión de que el diseño propuesto está entre los límites aceptables de calidad; teniendo en cuenta que la métrica TOC demostró que el 75% de las clases posee menos cantidad de operaciones que la media registrada en las mediciones, en el 75% de las clases los atributos de calidad se encuentran en un nivel satisfactorio; observándose un alto nivel de Reutilización y cómo están reducidas en menor grado la Responsabilidad y la Complejidad de implementación.

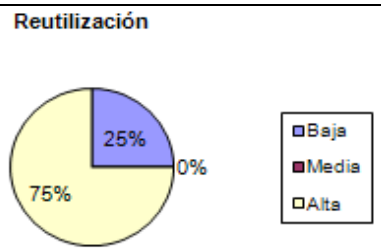
Métrica RC: Relaciones entre clases.

Se refiere al número de relaciones de uso de una clase, determinada por los atributos: Acoplamiento, Complejidad de mantenimiento, Reutilización y Cantidad de pruebas, existiendo una relación directa con los tres primeros e inversa con el último antes mencionado.

Como resultado de la evaluación de la métrica RC se obtuvo lo siguiente:

Tabla 4: Métrica Relaciones entre Clases (RC)

Relaciones entre Clases (RC).	
	
Atributo que evalúa: Acoplamiento: El aumento de las RC provoca un aumento del acoplamiento de la clase.	
Atributo que evalúa: Complejidad del mantenimiento: El aumento de las RC provoca un aumento de la complejidad del mantenimiento de la clase.	
Atributo que evalúa: Cantidad de Pruebas: El aumento de las RC provoca un aumento de la cantidad de pruebas de unidad necesarias para probar una clase.	

Atributo que evalúa: Reutilización: El aumento de las RC provoca una disminución en el grado de reutilización de la clase.	

Ver Anexo:

Anexo 12: Instrumento de medición de la métrica Relaciones entre Clases (RC).

Análisis de los resultados obtenidos en la evaluación de las métricas TOC y RC.

Después de analizar los resultados alcanzados en la aplicación de la métrica RC se demostró que el 75% de las clases poseen menos de 4 dependencias respecto a otras. En el 50% de las clases el grado de dependencia o acoplamiento es mínimo, la Complejidad de Mantenimiento, la Cantidad de Pruebas y la Reutilización se comportan favorablemente para un 75% de las clases.

3.3.2 Pruebas de Aplicación.

Tipos de Pruebas de Software.

Las pruebas en conjunto tienen como objetivo general verificar y validar un software, independientemente de las características y el entorno donde se desarrollen, además de los recursos y los factores vinculados al proceso de desarrollo. Según en el nivel de trabajo en que se encuentre un software se usa un tipo de prueba, entre algunos de estos se definen:

Pruebas de Unidad: se focaliza en ejecutar cada módulo, lo que proporciona un mejor modo de manejar la integración de las unidades en componentes mayores. Busca asegurar que el código funciona de acuerdo con las especificaciones.

Pruebas de Integración: se especializa en identificar errores introducidos por la combinación de programas probados unitariamente y verifica que las interfaces entre las entidades externas y las aplicaciones funcionan correctamente.

Pruebas del Sistema: se encarga de asegurar la apropiada navegación dentro del sistema, ingreso de datos, procesamiento y recuperación.

Pruebas de Aceptación: ayuda a la determinación por parte del cliente de la aceptación o rechazo del sistema desarrollado, es ejecutada antes de que la aplicación sea instalada dentro de un ambiente de producción.

Pruebas realizadas a la funcionalidad:

Un método de prueba es un enfoque sistemático, independiente del nivel en que se enmarque la prueba, que ayuda a encontrar buenos conjuntos de casos de prueba[13] para detectar diferentes tipos de errores. [41]

Descripción y aplicación de la Prueba de Caja Blanca o Estructural.

Descripción:
La Prueba de Caja Blanca también se conoce como Prueba de Caja Transparente o de Cristal.

En ese sentido el criterio de selección de casos de prueba buscará cierta cobertura no solo para la determinación de caminos independientes, sino también de valores para las condiciones de bucles dentro y fuera de sus límites operacionales basados en el contenido de los módulos.

Esta prueba consiste específicamente en cómo diseñar los casos de prueba atendiendo al comportamiento interno y la estructura del programa, examinándose la lógica interna sin considerar los aspectos de rendimiento.

Dentro de la prueba de caja blanca se incluyen las Técnicas de Pruebas que serán descritas a continuación:

Prueba del Camino Básico: Permite obtener una medida de la complejidad lógica de un diseño y usar la misma como guía para la definición de un conjunto de caminos básicos. Los casos de prueba obtenidos garantizan que durante la prueba se ejecute al menos una vez cada sentencia del programa.

[13] **Casos de prueba**: especifican una forma de probar el componente, incluye: la entrada, las condiciones bajo las cuales ha de probarse y los resultados esperados.

Prueba de Condición: Ejercita las condiciones lógicas contenidas en el módulo de un programa. Garantiza la ejecución por lo menos una vez de todos los caminos independientes de cada módulo, programa o método.

Prueba de Flujo de Datos: Se seleccionan caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa. Garantiza que se ejerciten las estructuras internas de datos para asegurar su validez.

Prueba de Bucles: Método de prueba que se centra exclusivamente en la validez de las construcciones de bucles. Garantiza la ejecución todos los bucles en sus límites operacionales. [49]

Aplicación:

Según descripción de la prueba de caja blanca presentada con anterioridad y a partir de la necesidad de crear un producto de alta calidad que cumpla con los requisitos deseados por el cliente es preciso valorar qué tan certera ha sido la implementación de la solución para la obtención de gráficos de los indicadores de la planificación y para ello es necesario aplicar una de las técnicas que esta comprende, en este caso la del camino básico.

Para ello es necesario conocer el número de caminos independientes de un determinado algoritmo mediante el cálculo de la complejidad ciclomática. Se debe comenzar por un análisis del código, posteriormente son enumeradas cada una de las instrucciones, se construye el grafo de flujo asociado y mediante las fórmulas pertinentes se calcula dicha complejidad:

A continuación se analizan y enumeran las sentencias de código de uno de los procedimientos contenidos en la clase CmpgestordocumentalModel, específicamente de la funcionalidad: eliminarAnexo(), la misma por su parte se encarga de eliminar los anexos seleccionados por el usuario.

```
public function eliminarAnexo($arrayid, $idelemento) {
    $result = array();    1
    if (!isset($arrayid)) {    2
         $array=DatAnexos::Idanexo($idelemento); 2
         $cont=0;    2
         foreach($array as $elemento){    3
             $arrayid[$cont]=$elemento[0]; 3
        } 3
    } 2
    for ($k = 0; $k < count($arrayid); $k++) { 4
        DatAnexos::EliminaranexoElemento($arrayid[$k], $idelemento); 4
    } 4
    for ($k = 0; $k < count($arrayid); $k++) { 5
        $repetido = count(DatAnexos::AnexoRepetido($arrayid[$k])); 5
        if ($repetido == 0) { 6
            $denominacionSelec = DatAnexos::SelectDenominacion($arrayid[$k]); 6
            $extencionSelec = DatAnexos::SelectExtencion($arrayid[$k]); 6
            $result[$k][0] = $denominacionSelec[0][0];  6
            $result[$k][1] = $extencionSelec[0][0];  6
            DatAnexos::Eliminaranexo($arrayid[$k]);  6
        } 6
    } 5
    for ($k = 0; $k < count($result); $k++) { 7
            $nombre = $result[$k][0];   7
            $extencion = $result[$k][1];   7
            $dir_index = $_SERVER['SCRIPT_FILENAME'];  7
            $ubicacion = substr($dir_index, 0, strrpos($dir_index, 'web')) . "apps/planificacion/comun/doc_anexos/$n
            unlink($ubicacion);  7
        } 7
    return $result; 8
}
```

Figura 14 Funcionalidad eliminaranexo().

Luego del paso anterior, es necesario representar el grafo de flujo asociado al código antes presentado a través de nodos, aristas y regiones, en ese caso:

Nodo: Círculos representados en el grafo de flujo, el cual representa una o más secuencias del procedimiento, un nodo en sí puede representar un proceso, una secuencia de procesos o una sentencia de decisión. Los nodos que no están asociados se utilizan al inicio y final del grafo.

Aristas: Saetas a través de las cuales se unen los Nodos y constituyen el flujo de control del procedimiento.

Regiones: Las regiones son las áreas delimitadas por las aristas y nodos.

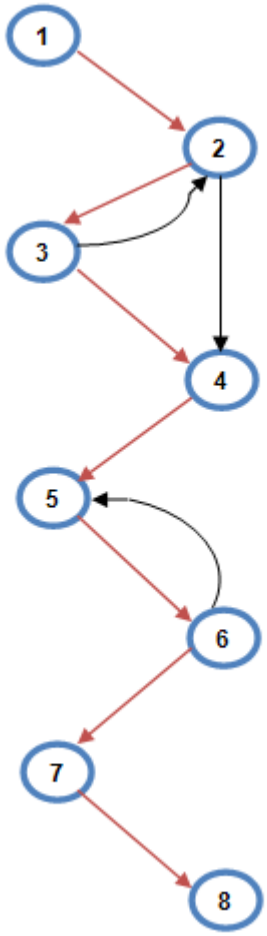


Figura 15 Grafo de Flujo asociado al algoritmo eliminaranexo().

Una vez construido el grafo de flujo asociado al procedimiento anterior se determina la complejidad ciclomática, el cálculo es necesario efectuarlo mediante tres vías o fórmulas de manera tal que quede justificado el resultado, siendo el mismo en cada caso:

1. V (G) = (A – N) + 2

Siendo “A” la cantidad total de aristas y “N” la cantidad total de nodos.

V (G) = (10-8) + 2

V (G) = 4.

2. V (G) = P + 1

Siendo “P” la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

V (G) = 3 + 1

V (G) = 4.

3. V (G) = R

Siendo “R” la cantidad total de regiones, se incluye el área exterior del grafo, contando como una región más.

V (G) = 4.

El cálculo efectuado mediante las fórmulas antes presentadas muestran una complejidad ciclomática de valor 4, de manera que existen cuatro posibles caminos por donde el flujo puede circular, este valor representa el número mínimo de casos de pruebas para el procedimiento tratado.

Seguidamente es necesario especificar los caminos básicos que puede tomar el algoritmo durante su ejecución.

Camino básico #1: 1 – 2 – 4 – 5 – 6 – 7– 8.

Camino básico #2: 1 – 2 – 3 – 4 – 5 – 6 – 7– 8.

Camino básico #3: 1 – 2 – 4 – 5 – 6 – 5 – 7 – 8.

Camino básico #4: 1 – 2 – 3– 4 – 5 – 6 – 5 – 7 – 8.

Se procede a ejecutar los casos de pruebas para cada uno de los caminos básicos determinados en el grafo de flujo.

La prueba de caja blanca se realizará a la funcionalidad eliminarAnexo() invocada cuando se pretende eliminar uno o varios anexos seleccionados por el usuario.

De forma general:

Se eliminará el anexo o los anexos seleccionados por el usuario, en caso de que se elimine un elemento de la planificación se eliminarán todos los anexos que este contenga.

1. Caso de prueba para el camino básico # 1.

Descripción: se debe eliminar un anexo seleccionado por el usuario.

Condición de ejecución: para ejecutar el algoritmo es necesario que los datos de entrada cumplan con los siguientes requisitos: en el parámetro $idelemento se va a encontrar el identificador del elemento al que se le van a eliminar los anexos y en el parámetro $arrayid se va a almacenar el anexo seleccionado por el usuario.

Entrada:

$ idelemento = 9000003030.

$ arrayid =[9000000030].

Resultados esperados: teniendo en cuenta los datos pasados por parámetro se debe eliminar el anexo seleccionado:

El algoritmo fue ejecutado correctamente.

2. Caso de prueba para el camino básico # 2.

Descripción: se debe eliminar el anexo del elemento eliminado por el usuario.

Condición de ejecución: para ejecutar el algoritmo es necesario que los datos de entrada cumplan con los siguientes requisitos: en el parámetro $idelemento se va a encontrar el identificador del elemento eliminado, este elemento no debe poseer más de un anexo.

Entrada:

$ idelemento = 9000003030.

Resultados esperados: el anexo perteneciente al elemento seleccionado debe ser eliminado.

El algoritmo fue ejecutado correctamente.

3. Caso de prueba para el camino básico # 3.

Descripción: se deben eliminar los anexos seleccionados por el usuario.

Condición de ejecución: para ejecutar el algoritmo es necesario que los datos de entrada cumplan con los siguientes requisitos: en el parámetro $idelemento se va a encontrar el identificador del elemento al que se desea eliminarle los anexos y el parámetro $arrayid va a almacenar los anexos seleccionados por el usuario.

Entrada:

$ idelemento = 9000003030.

$ arrayid =[9000000030, 9000000046, 9000000049].

Resultados esperados: teniendo en cuenta los datos pasados por parámetro se deben eliminar los anexos seleccionados:

El algoritmo fue ejecutado correctamente.

4. Caso de prueba para el camino básico # 4.

Descripción: se deben eliminar los anexos del elemento eliminado por el usuario.

Condición de ejecución: para ejecutar el algoritmo es necesario que los datos de entrada cumplan con los siguientes requisitos: en el parámetro $idelemento se va a encontrar el identificador del elemento eliminado, este elemento debe poseer más de un anexo.

Entrada:

$ idelemento = 9000003030.

Resultados esperados: todos los anexos pertenecientes al elemento seleccionado deben ser eliminados.

El algoritmo fue ejecutado correctamente.

Descripción y aplicación de la Prueba de Caja Negra o Funcional.

Descripción:

A este tipo de prueba también se le conoce como Prueba de Caja Opaca o Inducida por los Datos.

Se centra en lo que se espera de un módulo, es decir, intenta encontrar casos en que el módulo no se atiene a su especificación [50], esta prueba se limita a brindar solo datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo internamente, es decir, solo trabaja sobre su interfaz externa.

En esencia permite encontrar:
Funciones incorrectas o ausentes.
Errores de interfaz.
Errores en estructuras de datos o en accesos a las bases de datos externas.
Errores de rendimiento.
Errores de inicialización y terminación.

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. [49]

Aplicación:

A continuación se aplica la prueba de partición de equivalencia como parte de la realización de la prueba de caja negra sobre la interfaz que responde al requisito funcional **“**Adjuntar Anexo a elemento**”.**

La Partición de Equivalencia divide el dominio de entrada de un programa en un número finito de variables de equivalencia. Las variables de equivalencia representan un conjunto de estados válidos y no válidos

para las condiciones de entrada de un programa. Se definen dos tipos de variables de equivalencia, las válidas, que representan entradas válidas al programa, y las no válidas, que representan valores de entrada erróneos, aunque pueden existir valores no relevantes a los que no sea necesario proporcionar un valor real de dato.

Ver Anexos:

Anexo13: Caso de prueba “Adjuntar Anexo a elemento”.

Anexo 14: Descripción de variable.

Anexo15: Juegos de datos a probar.

La especificación de los restantes diseños de caso de prueba se encuentran descritos en el expediente de proyecto SIPAC 2.0 en la siguiente dirección: 1. Ingeniería\1.3 implementación y prueba\Diseño de casos de prueba\04-Gestor documental.

Resultados de las pruebas realizadas

Después de aplicados los métodos de prueba Caja Negra y Caja Blanca, se destaca que los resultados obtenidos hasta el momento han sido satisfactorios. Se realizaron pruebas de calidad interne del centro donde las 14 no conformidades detectadas fueron debidamente resueltas, quedando de esta forma la solución liberada y emitiéndose el Acta de liberación.

3.4 Conclusiones del capítulo.

El presente capítulo fue fundamental para garantizar la calidad de la funcionalidad desarrollada. Se precisaron los estándares de código para su implementación, se aplicaron las métricas: Relaciones entre Clases y Tamaño Operacional de la Clase para validar y evaluar el diseño, las cuales arrojaron valores satisfactorios para cada uno de los indicadores correspondientes. Finalmente se efectuaron pruebas a la funcionalidad para verificar su correcto funcionamiento, aplicándose pruebas de caja negra y de caja blanca donde cada una de las no conformidades detectadas fue debidamente resuelta.

CONCLUSIONES GENERALES.

Una vez terminado el trabajo de diploma se obtuvieron las siguientes conclusiones:

- Se evidenció la necesidad de realizar una solución informática capaz de ejecutar las funcionalidades referentes a la gestión documental, a partir del estudio de diferentes sistemas informáticos.
- La solución propuesta es factible para SIPAC, debido a que existe un conjunto mínimo de información necesaria, asociada a los elementos de la planificación a la que los usuarios van a tener acceso de manera local, haciendo una descarga en el sistema de la misma.
- Se exhibe valor técnico, donde se destaca la incorporación de principios por los que se mide la factibilidad de un diseño de software, ejemplo: la utilización de patrones que posibilita la reutilización, garantizando la sostenibilidad y mantenimiento del sistema.

RECOMENDACIONES.

- Concebir y desarrollar una segunda versión de la solución que incluya la funcionalidad de graficar atendiendo a: formato de los anexos y usuarios que anexaron los documentos.
- Fomentar el uso del sistema en los ministerios del país que carecen de un software para la gestión documental.

BIBLIOGRAFÍA REFERENCIADA.

[1] Bernal, Nestor. Proceso de Planificación por Objetivos en las entidades de las FAR. La Habana. 2008. [Consultado diciembre 2012].

[2] UNE-ISO 15489-1. Información y documentación. Gestión de documentos. http://redc.revistas.csic.es/index.php/redc/article/view/244/300. [Consultado diciembre 2012].

[3] Sobre alfresco la alternativa para la gestión de contenidos empresariales de código libre. http://www.alfresco.com/es/about/. [Consultado diciembre 2012].

[4] Grupo de Vigilancia Tecnológica, Dirección de Información, Universidad de las Ciencias Informáticas. Perfil de producto sobre Software para la Gestión Documental y Archivística. Diciembre 2011. [Consultado diciembre 2012].

[5] MunwarShariff. Enterprise Content Management Implementation. Pack, 2006. ISBN 1-904811-11-6. [Consultado diciembre 2012].

[6] Alfresco. http://www.alfresco.com/es/alfresco-para-el-gobierno. [Consultado enero de 2013].

[7] JumpBox for KnowledgeTree for Mac. http://es.download.cnet.com/JumpBox-for-KnowledgeTree/3000-10743_4-10905615.html. [Consultado enero de 2013].

[8] Sonia García Matureel, Taimy de la Caridad Castro García. Análisis y diseño del módulo Gestión de Tareas para el Gestor de Documentos Administrativos eXcriba. [Consultado enero de 2013].

[9] Marcel Sánchez Góngora Misael Fonseca Mata, Reinier Elejal de Chacon. Sistema de Gestión Integral de Documentos y Archivos, XVI Fórum de ciencia y Técnica. 2010. [Consultado enero de 2013].

[10] Sonia García Matureel y Taimy de la Caridad Castro García. Análisis y diseño del módulo Gestión de Tareas para el Gestor de Documentos Administrativos eXcriba. [Consultado enero de 2013].

[11] Centro de Informatización de Gestión de Entidades. Modelo De Desarrollo De Software. Octubre 2012. [Consultado enero de 2013].

[12] Sistemas Computin. http://sistemacomputin.blogspot.com/2010_06_01_archive.html. [Consultado enero de 2013].

[13] Mastermagazine. http://www.mastermagazine.info/termino/7006.php. [Consultado enero de 2013].

[14] CodeBox. Glosario php. http://www.codebox.es/glosario. [Consultado enero de 2013].

[15] about.com. http://aprenderinternet.about.com/od/Glosario/g/Que-Es-Html.htm. [Consultado enero de 2013].

[16] Wep developers notes. http://www.webdevelopersnotes.com/basics/languages_on_the_internet.php3. [Consultado enero de 2013].

[17] Html Quick.com. http://www.htmlquick.com/es/tutorials/css.html. [Consultado enero de 2013].

[18] Tutorial de UML. http://users.dcc.uchile.cl/~psalinas/uml/introduccion.html. [Consultado enero de 2013].

[19] Gómez Baryolo, Oiner, Morejón Borbón, Yoandry y García Tejo, Darien. Arquitectura tecnológica para el desarrollo de software. Habana: UCI. [Consultado enero de 2013].

[20] 'Cutter' Blades, Steve, Ramsay, Colin y Frederick, Shea. 2008. Learning Ext JS. s.l. Packt Publishing Ltd, 2008. ISBN 978-1-847195-14-2. [Consultado enero de 2013].

[21] zend framework2 . http://framework.zend.com/. [Consultado enero de 2013].

[22] Doctrine. http://www.doctrine-project.org. [Consultado enero de 2013].

[23] adaptive path. http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications. [Consultado enero de 2013].

[24] Visual-paradigm. http://www.visual-paradigm.com. [Consultado enero de 2013].

[25] Netbeans. http://netbeans.org/community/releases/roadmap.html. , [Consultado enero de 2013].

[26] Documentación del Servidor HTTP Apache 2.0. http://httpd.apache.org/docs/2.0/es/. [Consultado enero de 2013].

[27] PostgreSQL. The world's most advanced open source database Postgresql. http://www.postgresql.org. [Consultado enero de 2013].

[28] Subversion. http://subversion.apache.org. [Consultado enero de 2013].

[30] Rotta, Ing. Luis Zuloaga. Galeon. Galeon. http://www.galeon.com/zuloaga/Doc/AnalisisRequer.pdf. [Consultado marzo de 2013].

[31] Méndez, Gonzalo. Facultad de Informática Universidad Complutense de Madrid. Facultad de Informática Universidad Complutense de Madrid. http://www.fdi.ucm.es/profesor/gmendez/docs/is0809/03-Requisitos.pdf. [Consultado enero de 2013].

[32] Sommerville, Ian. Requerimientos del software. [Consultado enero de 2013].

[33] Ariadna Rendón Artola. Requisitos no funcionales SIPAC 2.0. [Consultado enero de 2013].

[34] Prácticas de software. http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/. [Consultado enero de 2013].

[35] Pensar en C++ (Volumen 2). Bruce Eckel. http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf. [Consultado enero de 2013].

[36] Universidad de valladolid. Departamento de informática. http://www.infor.uva.es/~felix/datos/priii/tr_patrones-2x4.pdf. [Consultado marzo de 2013].

[37] Sparx. http://www.sparxsystems.com.ar/resources/tutorial/uml2_sequencediagram.html. [Consultado marzo de 2013].

[38] Definicion.de. 2008. http://definicion.de/modelo-de-datos. [Consultado marzo de 2013].

[39] http://serk.kualtus.com/codigo.htm. [Consultado marzo de 2013].

[40] IngSoftwareII-CUFM. http://cufmingsoftware.wordpress.com/estandares-de-diseno/. [Consultado marzo de 2013].

[41] Ramírez, Jaime. Unidad de Programación. Métodos de Prueba del Software. [Consultado abril de 2013].

[42] La norma ISO 15489: un marco sistemático de buenas prácticas de gestión documental en las organizaciones. http://eprints.rclis.org/12263/1/Alonso_Garcia_Lloveras_-_La_norma_ISO_15489.pdf. [Consultado enero de 2013].

[43] Custodia documental. http://www.custodia-documental.com/2011/moreq-modelo-de-requisitos-para-la-gestion-de-documentos-de-archivo/. [Consultado enero de 2013].

[44] Introducing JSON. http://www.json.org/.[Consultado abril de 2013].

[45] Vates S.A. http://www.vates.com/cmmi/que-es-cmmi.html. [Consultado enero de 2013].

[46] IEEE-SA. 2010. “IEEE SA - 1471-2000 - IEEE Recommended Practice for Architectural Description for Software-Intensive Systems”. http://standards.ieee.org/findstds/standard/1471-2000.html. [Consultado enero de 2013].

[48] Electronic records management software applications design criteria standard. http://www.dtic.mil/whs/directives/corres/pdf/501502std.pdf. [Consultado enero de 2013].

[49] Universidad Autónoma de Baja California. [Consultado enero de 2013]. fcqi.tij.uabc.mx/usuarios/luisgmo/data/8.1%20prb-cal-mant.pdf. [Consultado enero de 2013].

[50] uc3m. http://www.it.uc3m.es/ttrd/material/05-pruebas-de-programas.pdf. [Consultado enero de 2013].

[51] Férnandez González, Mairelys y Zorrilla Rivera, Osley. Diseño e implementación del componente Ajuste al Costo del Subsistema Costos y Procesos del Sistema Integral de Gestión de Entidades CEDRUX Habana: UCI. [Consultado enero de 2013].

BIBLIOGRAFÍA CONSULTADA.

[1] Centro de Informatización de Gestión de Entidades. Modelo De Desarrollo De Software. Octubre 2012.

[2] Férnandez González, Mairelys y Zorrilla Rivera, Osley. Diseño e implementación del componente Ajuste al Costo del Subsistema Costos y Procesos del Sistema Integral de Gestión de Entidades CEDRUX.

[3] Ariadna Rendón Artola y Manuel Alejandro Castellanos Pérez. Modelación y construcción de los componentes Gestor de actividades y Calendario para el Subsistema Planificación por Objetivos del Sistema Integral de Gestión de Entidades.

[4] Liu Pérez Ballester y Jiorqui Vázquez Fitó. Componente Gestor de Intercambio de Información JIT para el Sistema para la Planificación de Actividades (SIPAC).

[5] Centro de Informatización de la Gestión de Entidades. SIPAC 2.0 Gestor documental: DISEÑO DE CASOS DE PRUEBA.

[6] Centro de Informatización de la Gestión de Entidades. SIPAC 2.0 Gestor documental: ESPECIFICACION DE REQUISITOS DE SOFTWARE.

[7] Centro de Informatización de la Gestión de Entidades. SIPAC 2.0 Gestor documental: DESCRIPCIÓN DE REQUISITOS.

[8] Grupo de Vigilancia Tecnológica, Dirección de Información, Universidad de las Ciencias Informáticas. Perfil de producto sobre Software para la Gestión Documental y Archivística. Diciembre 2011.

[9] Centro De Informatización De La Gestión De Entidades. Arquitectura De Software. 28/02/2011. [Consultado enero de 2013].