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

ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DEL MÓDULO DEBIDO PROCESO DEL SISTEMA DE INVESTIGACIÓN E INFORMACIÓN POLICIAL

AUTOR

LEANDRO TABARES MARTÍN

TUTORA

ING. YUNEXIS RODRÍGUEZ BARYOLO

CO-TUTORA

ING. YELEYNE MAURE DÍAZ

Ciudad de La Habana, 2011

"Año 53 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos que somos los autores de este trabajo y autorizamos a la Facultad 2 de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año 2011.

_____

Leandro Tabares Martín

Autor

_____ _____

Ing. Yunexis Rodríguez Baryolo Ing. Yeleyne Maure Díaz

Tutora Co-tutora

AGRADECIMIENTOS

A mis padres, por todo su cariño y por siempre haberme enseñado que su mayor legado es el conocimiento que pudiese obtener.

A mis abuelos, por siempre haber estado ahí cuando los necesité y tener tanta paciencia conmigo a lo largo de los años.

A mi abuela, por enseñarme que lo que tenemos en la mente, sólo Dios nos lo puede quitar.

A mi novia, por todo su amor y por haberme apoyado en el desarrollo de este trabajo.

A mi familia, por ser la mejor del mundo.

A mis amigos, por su apoyo incondicional, enseñarme que "La vida es dura" y a seguir adelante cuando el camino es más difícil.

A Oscarito, por haberle explicado binario a un niño de quinto grado que hoy se convierte en ingeniero.

DEDICATORIA

A mis padres, abuelos y familia en general porque al cumplir este sueño, cumplo uno de los suyos.

RESUMEN

El Cuerpo de Investigaciones Científicas, Penales y Criminalísticas de la República Bolivariana de Venezuela (CICPC) es una de las instituciones más importantes que luchan contra la criminalidad existente en ese país, utiliza el Sistema Integrado de Información Policial para gestionar digitalmente la información que maneja, pero este software ya se encuentra obsoleto, no satisface las necesidades actuales de esta institución y tampoco soporta los procesos judiciales relacionados con la investigación disciplinaria a la cual puede ser sometido un funcionario. Como parte del proyecto de modernización del Cuerpo de Investigaciones Científicas Penales y Criminalísticas de la República Bolivariana de Venezuela surge el presente trabajo, el cual se enmarca en la investigación y desarrollo del Módulo Debido Proceso del Sistema de Investigación e Información Policial (SIIPOL), nuevo sistema de gestión policial. El Módulo Debido Proceso se encarga de asegurar que se siga un correcto proceso judicial con aquellos funcionarios que sean objeto de investigaciones disciplinarias. Con el desarrollo e implantación del Módulo Debido Proceso en el nuevo sistema se espera facilitar el flujo de información asociada a los procesos mencionados anteriormente.

ÍNDICE DE CONTENIDOS

INTRODUCCIÓN

La República Bolivariana de Venezuela es uno de los países que cuenta con mayores índices de criminalidad en el mundo. Según el Índice de Paz Mundial se encuentra ubicado en la posición ciento veinte, por lo cual es el país más inseguro de América Latina. Para enfrentar esta problemática en el año 2001 el presidente Hugo Rafael Chávez Frías ordenó la creación del Cuerpo de Investigaciones Científicas, Penales y Criminalísticas.

El CICPC, es el encargado de garantizar la eficiencia en la investigación del delito, mediante su determinación científica, asegurando que el ejercicio de la acción penal conduzca a una sana administración de la justicia. Dicho Cuerpo está formado por funcionarios que deben cumplir y hacer que se cumpla la legislación establecida en Venezuela. Frecuentemente se dan casos de funcionarios que contravienen las leyes adoptadas y, por lo tanto, han de ser sometidos a investigaciones disciplinarias y a procesos judiciales.

Actualmente la documentación asociada a dichos procesos se maneja de forma manual, dando lugar a la pérdida y descontrol de la información; a la vez que la misma se encuentra sometida al deterioro temporal. Elementos como las Actas de Procedimiento Especial, en las cuales se recogen las declaraciones realizadas durante el análisis de faltas disciplinarias, son redactados en formato físico y en muchas ocasiones no contienen todos los elementos requeridos por este tipo de documento. Los registros de sanciones adoptadas por los encargados de juzgar a los funcionarios contraventores de la ley con gran regularidad se pierden o son muy difíciles de consultar debido al descontrol de la documentación que existe actualmente en el CICPC.

Esta situación provoca que los antecedentes penales de los funcionarios se pierdan o no sean localizados cuando son requeridos, imposibilitando esto el buen control del estado legal de los recursos humanos con que cuenta el CICPC.

Como conclusión de la situación planteada anteriormente, se define el siguiente **problema a resolver**. ¿Cómo facilitar y agilizar la gestión y control de la documentación asociada a los procesos judiciales a que se encuentren sometidos los funcionarios del CICPC?

El **objeto de estudio** de la presente investigación es el proceso de gestión de información judicial y el **campo de acción** donde se desarrolla es el proceso de Gestión de Información Judicial orientado al análisis disciplinario y judicial de los funcionarios del CICPC.

El **objetivo general** propuesto es desarrollar un subsistema que se integre al Sistema de Investigación e Información Policial, que permita gestionar y controlar la documentación asociada al análisis disciplinario y judicial a que se puedan encontrar sometidos los funcionarios del CICPC; para darle cumplimiento se plantean los siguientes **objetivos específicos**:

- ✓ Analizar los requisitos levantados asociados al proceso judicial a que se pueden encontrar sometidos los funcionarios del CICPC producto a faltas disciplinarias.
- ✓ Implementar y certificar un subsistema que permita gestionar y controlar la documentación generada por los procesos judiciales a que se sometan a los funcionarios del CICPC.
- ✓ Generar documentación sobre el desarrollo del módulo Debido Proceso.

La **idea a defender** que se plantea es: el análisis, diseño e implementación del subsistema Debido Proceso facilita a esta dirección del Cuerpo de Investigaciones Científicas, Penales y Criminalísticas, la consulta y seguimiento de la información que se genera como resultado de los procesos disciplinarios que se le aplican a los funcionarios del Cuerpo.

Las **tareas de investigación** a realizar para alcanzar los objetivos específicos son las siguientes:

1. Estudio de los conceptos sobre Sistemas de Gestión de Información y sistemas basados en la Gestión Policial y del alcance del SIIPOL para manejar los procesos de la Dirección de Inspectoría del Debido Proceso.

2. Descripción acerca del lenguaje, metodología de desarrollo y las herramientas establecidas para la solución.

3. Análisis del modelo de casos de uso que da cumplimiento a los requisitos funcionales y no funcionales asociados al módulo Debido Proceso.

4. Aplicación de los patrones para el refinamiento del modelo de diseño.

5. Elaboración de la propuesta de solución.

6. Verificación del correcto funcionamiento del módulo Debido Proceso integrado a los demás subsistemas del SIIPOL.

Para dar solución al problema se necesitará el método Modelación, fundamentalmente para visualizar los diferentes procesos, también para representar partes del sistema desde diferentes puntos de vista, como son el análisis, diseño, implementación y otros que son de vital importancia para el entendimiento del funcionamiento del sistema. Además se hace necesario el uso del Método Analítico Sintético, para realizar un análisis de la documentación referente a los procesos judiciales producto a faltas disciplinarias en que se vean envueltos los funcionarios del CICPC, para identificar lo esencial dentro de cada proceso, determinar los rasgos que los identifican y diferencian de otros. También se analizará la documentación relacionada con los Sistemas de Gestión de Información y Gestión de Información Policial, logrando así un mejor entendimiento de estos tipos de sistemas.

La investigación se organizará en tres capítulos, quedando estructurada de la siguiente forma:

Capítulo 1. Fundamentación Teórica: se tratarán elementos teóricos de la investigación relacionados con la gestión de información, gestión de información policial, metodología, lenguajes y herramientas que serán utilizadas para implementar la solución.

Capítulo 2. Análisis y Diseño de la solución propuesta: basado en los requisitos funcionales y no funcionales del sistema, se analizará la descripción de casos de uso a implementar y se diseñará la propuesta de solución. También se realizará el Modelo de Análisis y posteriormente el Modelo de Diseño, el cual será utilizado como principal fuente de entrada para el Modelo de Implementación.

Capítulo 3. Implementación y certificación de la solución propuesta: en este capítulo se elaborará el Modelo de Implementación, el cual representa la composición física en términos de subsistemas y elementos de implementación. Además se expondrán los resultados de las pruebas que serán utilizadas para certificar la solución propuesta.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1. INTRODUCCIÓN

Este capítulo tiene como objetivo fundamental mostrar el marco teórico en el cual se desarrolla el presente trabajo. Además se abordarán temas como la gestión de información y la gestión de información policial, por estar estos íntimamente relacionados al sistema para el cual se implementará el subsistema; lográndose así un mayor grado de comprensión de los conceptos y normas existentes sobre el tema en el que se enmarca el problema a resolver. También serán expuestas las principales características del ambiente de desarrollo, lenguajes y herramientas que se utilizarán en la implementación de la solución.

1.2. SISTEMAS DE GESTIÓN DE INFORMACIÓN

En la era de la información, con la explosión de sus tecnologías se ha generado un crecimiento cada vez mayor del volumen de datos a manejar por los seres humanos, provocando esto un incremento en el nivel de complejidad y en el tiempo necesario para localizar una información específica sobre un tema determinado. En este contexto “debe entenderse que las Tecnologías de Información y las Comunicaciones no son más que un medio para transmitir y gestionar datos, información y conocimiento”. (Guibert Estrada, 2008)

“Las Tecnologías de la Información han sido conceptualizadas como aquellas herramientas y métodos empleados para recabar, retener, manipular o distribuir información, que se encuentra generalmente asociada con las computadoras y las tecnologías afines aplicadas a la toma de decisiones.” (Quiroga, 2002)

La información es un elemento fundamental para el desarrollo. Con el decursar de los años, la gestión de la información ocupa, cada vez más, un espacio en la economía de los países a escala mundial, utilizando nuevas tecnologías y nuevos métodos de avance. (Quintana, 2002)

“La gestión de la información es el proceso de analizar y utilizar la información que se ha recabado y registrado para permitir a los usuarios (de todos los niveles) tomar decisiones documentadas” (Wayne Bartle, 2009).

Teniendo en cuenta lo expresado anteriormente se puede afirmar que un sistema de gestión de información es un “conjunto de elementos que interactúan entre sí con el fin de apoyar las actividades de una empresa o negocio. Son procesos automatizados que se realizan sobre un

conjunto de datos, los cuales consisten, por ejemplo, en recogerlos, agruparlos, analizarlos y difundirlos con el fin de realizar actividades de control.” (Peralta, 2003)

Por lo tanto, la gestión de la información implica (Wayne Bartle, 2009):

- Determinar la información que se precisa: durante la planificación, gestión y supervisión de diferentes procesos se genera mucha información, por tanto, un buen Sistema de Gestión de Información debe ayudar a los usuarios a saber qué información necesitan recabar, para tomar diferentes decisiones en distintos momentos.
- Recoger y analizar la información: la información puede conseguirse de informes técnicos, libros de registro, formularios de los diferentes ejecutantes, reuniones con la comunidad, entrevistas, observación y mapas comunitarios.
- Registrarla y recuperarla cuando sea necesaria: es importante guardar la información para futuras referencias. Puede guardarse en libros de registro locales, informes de progreso, formularios o incluso en la mente de las personas. El principio más importante del registro de informaciones es la facilidad con la que pueden recuperarse.
- Utilizarla: se puede utilizar para solucionar problemas comunitarios, determinar recursos (cantidad y naturaleza), solicitar apoyos y planear proyectos.
- Divulgarla: para que la información tenga un uso adecuado tiene que compartirse con los demás interesados o usuarios. Esta información puede ayudarles en sus decisiones de gestión y también puede ayudar al que la recoge a encontrar significados o usos relacionados con la gestión.

Para desarrollar la gestión de información, se deben seguir 4 objetivos fundamentales (Barreiro Noa, 2003):

- Maximizar el valor y los beneficios derivados del uso de la información.
- Minimizar el costo de adquisición, procesamiento y uso de la información.
- Determinar responsabilidades para el uso efectivo, eficiente y económico de la información.
- Asegurar un suministro continuo de la información.

1.3. SISTEMAS DE GESTIÓN POLICIAL

Los índices de crimen, delitos por armas de fuego y violencia se han incrementado a nivel mundial en las últimas décadas, sobre todo en América Latina. Por esta razón muchas organizaciones policiales han decidido introducir los Sistemas de Gestión de Información Policial para el desarrollo o el apoyo de las investigaciones, ya que permiten almacenar la información recuperada, que facilita el esclarecimiento de determinados casos investigativos, al brindar información actualizada y de manera rápida. Unido a esto, se están estableciendo además sistemas de control de armas por ejemplo en Argentina, Ecuador, Rusia, Japón, que sirven no solo para mejorar la calidad de la información interna, sino para tener un registro exacto de todas las armas de fuego que se encuentran circulando en un determinado país. (Segura, 2005)

Los Sistemas de Gestión Policial brindan un apoyo al desarrollo y esclarecimiento de las investigaciones policiales, poseen como objetivo principal el registro, control y seguimiento de las denuncias de los hechos delictivos, así como otros casos que no constituyen delitos, como son: los índices de peligrosidad; las muertes, lesiones y daños por accidentes de tránsito, los ausentes a domicilio, los suicidios y las muertes naturales. Estos constituyen el medio automatizado que permite medir el comportamiento del delito en diferentes períodos. (Diaz Cabrera, y otros, 2009).

El software de este tipo que se encuentra en el mercado actual son sistemas desarrollados a la medida de las organizaciones que los utilizan, provocando esto la no estandarización de los Sistemas de Gestión Policial.

Entre los sistemas policiales que existen en el mundo se cuentan el Sistema de Información de Schengen en Europa, el cual permite a las autoridades competentes de los estados miembros disponer de información relativa a algunas categorías de personas y objetos. (Unión Europea). El Sistema Inteligente de Respuesta Eficiente (SIRE) es utilizado por la Policía Local de Valencia para procesar las llamadas de los ciudadanos, evaluar su urgencia y recursos necesarios para atenderlas. Además sitúa en un plano de la ciudad las distintas incidencias y las patrullas más cercanas a esos puntos, que reciben las alertas en sus PDA. (LASPROVINCIAS.ES)

El Sistema Automatizado Jurídico Operativo (SAJO), perteneciente al Ministerio del Interior de la República de Cuba, maneja el registro, control y seguimiento de los hechos delictivos denunciados en cualquier lugar del territorio nacional. Su objetivo principal es el registro y

control de todas las denuncias ocurridas en cualquier lugar del país; para esto emplea una base de datos central, única en cada provincia, a fin de evitar que existan diferencias en los contenidos de las bases de datos de uno u otro lugar de la estructura. (Herrera Pereda, 2009)

Todos los sistemas analizados carecen de soporte para la gestión de información asociada a los procesos disciplinarios a que se sometan los integrantes de los distintos cuerpos policiales que los utilizan, a la vez que no permiten la gestión de información similar a la manejada por la Dirección de Inspectoría del Debido Proceso del CICPC.

1.4. MÓDULO DEBIDO PROCESO

Los recursos humanos son la base para el funcionamiento del CICPC. El proceso de enfrentamiento del delito necesita de hombres y mujeres íntegros que den al traste con los contraventores de la ley. Para evitar la corrupción y la falta de disciplina en sus filas, el CICPC tiene una serie de procesos judiciales a los cuales se puede someter a los funcionarios que violen las leyes establecidas. El módulo Debido Proceso se encarga de gestionar y controlar la información generada a lo largo de dichos procesos y persistirla a través del tiempo.

1.5. METODOLOGÍA, LENGUAJES Y HERRAMIENTAS DE DESARROLLO

Por pertenecer el módulo Debido Proceso al sistema SIIPOL, debe ajustarse al uso de las tecnologías y metodologías orientadas por la dirección del proyecto, quedando las mismas fuera del objetivo a investigar. Por esta razón el presente trabajo se limitará a hacer una breve exposición de las mismas.

1.5.1. METODOLOGÍA

Durante todo el desarrollo de software muchas tareas y actividades se tornan engorrosas, trayendo esto consigo que el proceso de desarrollo de software se convierta en riesgoso y difícil de controlar, y de no dársele un tratamiento correcto lo que se obtiene son clientes insatisfechos con el resultado obtenido. La forma para solucionar o tratar de llevar a cabo un eficiente desarrollo de software es aplicando una metodología de desarrollo de software, que es un conjunto de pasos y procedimientos que deben seguirse para desarrollar software; indican quién debe hacer cada actividad, cuándo hacerla y qué debe hacer (Letelier, 2003).

El Proceso Unificado de Desarrollo (RUP) describe cómo aplicar efectivamente enfoques comprobados comercialmente para el desarrollo de software. Es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes

áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos. (Jacobson, y otros, 2000)

Basado en lo anteriormente expuesto se decidió que el sistema SIIPOL fuese desarrollado utilizando la metodología RUP, ya que es un software de gran volumen, con clientes a distancia, con un equipo de desarrollo compuesto en gran medida por estudiantes que cambian al pasar los años, por lo que se hace indispensable que todo esté lo mejor documentado posible.

El Proceso Unificado de Desarrollo divide el proceso de desarrollo en 4 fases, la fase de Inicio o Conceptualización cuyo objetivo es determinar la visión del proyecto. La fase de Elaboración, en esta etapa el objetivo es determinar la arquitectura óptima. La siguiente fase es Construcción, donde el objetivo es llegar a obtener la capacidad operacional del software y por último la fase de Transición, el objetivo de la misma es llegar a obtener versiones ejecutables del producto. Cada una de estas fases es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. (Guibert Estrada, 2008)

RUP se divide en nueve disciplinas, las mismas son (Rational Corporation, 2003):

- Modelamiento del negocio: su propósito es entender los problemas existentes en la organización e identificar mejoras potenciales; evaluar su impacto desde el punto de vista de cambio organizacional; asegurar que los clientes, usuarios finales, desarrolladores y otras partes tengan un entendimiento común de la organización; obtener los requisitos del sistema necesarios para la organización cliente y entender cómo encaja la utilización del software dentro de la organización.
- Requerimientos: define qué es lo que el sistema debe hacer; provee a los desarrolladores del sistema un mejor entendimiento de las cosas que se necesitan del sistema; define el alcance del sistema; provee una base para determinar el contenido técnico de las iteraciones; provee las bases para determinar un aproximado del costo y tiempo necesario para desarrollar el sistema; se definen las interfaces de usuario determinadas por las necesidades reales del cliente.
- Análisis y Diseño: permite pasar de los requerimientos a lo que será el sistema; evolucionar a una arquitectura robusta del sistema; adaptar el diseño a la forma en que se implementará con el objetivo de lograr optimizaciones.
- Implementación: en la misma se define la organización del código, dividiéndolo en subsistemas y capas; se implementan los elementos del diseño en términos de ficheros

de implementación; se prueban los componentes desarrollados unitariamente; se integran los resultados producidos por las distintas partes en un sistema de software.

- Pruebas: en ella se localizan y documentan los defectos en la calidad del software; se provee asesoramiento en materia de calidad del software; se validan las cosas que se asumieron en las especificaciones de diseño y requisitos a través de demostraciones concretas; se valida que el producto de software funciona como fue diseñado y se valida que los requerimientos fueron correctamente implementados.
- Despliegue: describe las actividades necesarias para asegurar que el producto de software es accesible a los usuarios finales.
- Configuración y cambios: en ella se describen las acciones a tomar para asegurar que no existan conflictos al integrar las distintas partes del sistema de software implementadas por los integrantes de los equipos de desarrollo.
- Administración del proyecto: provee una estructura para administrar proyectos intensivos de software; proporciona una guía práctica para planificaciones, manejo del personal, ejecución y monitoreo de proyectos; además brinda una estructura para el manejo de riesgos.
- Ambiente: esta disciplina se enfoca en las actividades necesarias para configurar el proceso de desarrollo para un proyecto y describe las actividades necesarias para desarrollar una guía de soporte del proyecto.

RUP está basado en componentes. Utiliza el Lenguaje Unificado de Modelado (UML) para preparar todos los esquemas de un sistema de software. De hecho, UML es una parte esencial de la metodología, no obstante, los verdaderos aspectos definitorios del proceso unificado se resumen en tres características claves (Guibert Estrada, 2008):

- Dirigido por casos de uso: quiere decir que el proceso de desarrollo sigue un hilo – avanza a través de una serie de flujos de trabajo que parten de los casos de uso. (Jacobson, y otros, 2000)
- Centrado en la arquitectura: la arquitectura surge de las necesidades de la empresa y se refleja en los casos de uso. Por un lado los casos de uso deben encajar en la arquitectura cuando se llevan a cabo; por otro lado, la arquitectura debe permitir el desarrollo de todos los casos de uso requeridos en el momento y en el futuro; por lo que tanto la arquitectura como los casos de uso deben evolucionar en paralelo. (Jacobson, y otros, 2000)

- Iterativo e incremental: el desarrollo de un producto software comercial supone un gran esfuerzo que puede durar entre varios meses hasta posiblemente un año o más. Es práctico dividir el trabajo en partes más pequeñas o mini proyectos. Cada mini proyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en el flujo de trabajo, y los incrementos, al crecimiento del producto. (Jacobson, y otros, 2000)

UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. Está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. (Larman, 2002)

UML es un lenguaje más expresivo, claro y uniforme que los anteriores definidos para el diseño Orientado a Objetos, que no garantiza el éxito de los proyectos pero sí mejora sustancialmente el desarrollo de los mismos, al permitir una nueva y fuerte integración entre las herramientas, los procesos y los dominios. (Larman, 2002) El mismo permite realizar una descripción altamente detallada del sistema, de procesos del negocio, de componentes de software reutilizables entre otros; por estas razones se ha convertido prácticamente en un estándar en la modelación de sistemas orientados a objetos.

Dada la condición de lenguaje que tiene UML cuenta con reglas para combinar todos los elementos. UML estandariza 9 tipos de diagramas para representar gráficamente un sistema desde distintos puntos de vista (Booch, y otros, 1999):

- Diagramas de estructura estática: describen las propiedades estructurales del sistema.
  - ✓ Diagrama de clases: conjunto de clases, interfaces y colaboraciones; así como sus colaboraciones.
  - ✓ Diagrama de objetos: conjunto de objetos y sus relaciones.
  - ✓ Diagrama de casos de uso: conjunto de casos de uso, actores y sus relaciones.
- Diagramas de comportamiento: hacen énfasis en lo que debe suceder en el sistema modelado.
  - ✓ Diagramas de interacción (secuencia y colaboración): objetos y sus relaciones, incluyendo los mensajes que pueden ser enviados entre ellos.
  - ✓ Diagrama de estados: muestra una máquina de estado que consta de estados, transiciones, eventos y actividades.

- ✓ Diagrama de actividad: es un tipo especial de diagrama de estados que muestra el flujo de actividades dentro de un sistema.

- Diagramas de implementación: muestra las dependencias entre las partes de código del sistema y la estructura del sistema en ejecución.
  - ✓ Diagrama de componentes: organización y las dependencias entre un conjunto de componentes.
  - ✓ Diagrama de despliegue: configuración de nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos.

1.5.2. HERRAMIENTAS DE MODELADO

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora) proporcionan al ingeniero la posibilidad de automatizar actividades manuales y de mejorar su visión general de la ingeniería. Al igual que las herramientas de ingeniería y de diseño asistidos por computadora que utilizan los ingenieros de otras disciplinas, las herramientas CASE ayudan a garantizar que la calidad se diseñe antes de llegar a construir el producto. (Pressman)

En la actualidad la tecnología CASE ha permitido evolucionar la actividad de desarrollar software hacia un proceso automatizado, ayudando a perfeccionar la calidad y la productividad en el desarrollo de Sistemas de Gestión de Información, entre sus principales objetivos tiene (Collado Cabeza, y otros, 2003):

- Permitir aplicar en la práctica metodologías, agilizando el trabajo.
- Facilitar la realización de prototipos y el desarrollo conjunto de aplicaciones.
- Simplificar el mantenimiento del software.
- Mejorar y estandarizar la documentación.
- Aumentar la portabilidad de las aplicaciones.
- Facilitar la reutilización de componentes software.
- Permitir un desarrollo y un refinamiento visual de las aplicaciones, mediante la utilización de gráficos.

La herramienta CASE seleccionada para el modelado del SIIPOL fue Visual Paradigm debido a que soporta el ciclo de vida del software definido por RUP. También Visual Paradigm para UML es compatible con una serie de lenguajes para la generación de código ampliamente difundidos como son: Java, C++, PHP, entre otros. Una cuestión muy importante en la selección de esta herramienta para el modelado del SIIPOL es que soporta ingeniería inversa a partir de clases Java (lenguaje definido para la implementación del sistema) y de mapeos de Hibernate (framework utilizado para el acceso a bases de datos en el SIIPOL). Permite exportar e importar archivos de Rational Rose (.MDL / .CAT), como también modelos de proyectos de/a XML abierto y proporciona abundantes tutoriales de UML, demostraciones interactivas y proyectos. (Visual Paradigm Organización, 2009)

1.5.3. LENGUAJE DE PROGRAMACIÓN

Java es un lenguaje de desarrollo de propósito general, y como tal es válido para realizar todo tipo de aplicaciones profesionales. Incluye una combinación de características que lo hacen único y está siendo adoptado por multitud de fabricantes como herramienta básica para el desarrollo de aplicaciones comerciales de gran repercusión.

Algunas de las principales características de Java son:

- Los programas ejecutables, creados por el compilador de Java, son independientes de la arquitectura. Se ejecutan indistintamente en una gran variedad de equipos con diferentes microprocesadores y sistemas operativos.
- Permite escribir Applets (pequeños programas que se insertan en una página HTML y se ejecutan en el ordenador local).
- Se pueden escribir aplicaciones cliente/servidor, aplicaciones distribuidas en redes locales y en Internet.
- Es fácil de aprender y está bien estructurado.
- Las aplicaciones son fiables. Puede controlarse su seguridad frente al acceso a recursos del sistema y es capaz de gestionar permisos y criptografía.
- Aprovecha características de la mayoría de los lenguajes modernos evitando sus inconvenientes.
- Tiene una gran funcionalidad gracias a sus librerías.
- No tiene punteros manejables por el programador, aunque los maneja interna y transparentemente.

- El manejo de memoria no es un problema, la gestiona el propio lenguaje y no el programador; esto es posible porque incorpora un recolector automático de memoria (garbage collector).

A causa de todas estas ventajas que presenta el lenguaje Java, creado por Sun Microsystems Inc., fue seleccionado para el desarrollo del sistema SIIPOL.

1.5.4. PLATAFORMA DE DESARROLLO

La evolución de Java, que ha pasado de ser un medio de desarrollo de applets para ser ejecutados en navegadores a un modelo de programación capaz de manejar las aplicaciones de una empresa de hoy día, ha sido extraordinaria. Con el paso de los años, Java ha desarrollado tres ediciones de plataformas diferentes, cada una de ellas destinada a cubrir un conjunto diferente de necesidades de programación (Allamaraju, y otros):

- **La plataforma Java 2 Standard Edition (J2SE):**
  Es la plataforma más utilizada dentro de Java, consistente en un entorno de tiempo de ejecución y un conjunto de varios API para crear una amplia variedad de aplicaciones que abarca desde applets, pasando por aplicaciones independientes ejecutables en distintas plataformas, hasta aplicaciones de cliente para diversas aplicaciones de empresa.
- **La plataforma Java 2 Enterprise Edition (J2EE):**
  Es una plataforma para crear aplicaciones de servidor. Las aplicaciones de servidor tienen requisitos adicionales durante la fase de desarrollo. J2EE proporciona la infraestructura necesaria para satisfacer estas necesidades.
- **La plataforma Java 2 Micro Edition (J2ME):**
  Esta edición, la última en agregarse a la familia Java, permite la creación de aplicaciones Java para “micro-dispositivos” (dispositivos con un apoyo de pantalla y memoria limitados, como teléfonos móviles, PDAs, etc.)

Para la creación del SIIPOL la plataforma elegida fue la J2EE. La misma proporciona un entorno gestionado para componentes y las aplicaciones de J2EE son céntricas respecto del contenedor. Al ser gestionados, los componentes de esta plataforma utilizan la infraestructura proporcionada por los servidores de J2EE sin que el programador sea consciente de ello. Las aplicaciones de J2EE también son declarativas, un mecanismo con el que se puede modificar y controlar el funcionamiento de las aplicaciones sin cambiar de código. (Allamaraju, y otros)

La reutilización de código es uno de los aspectos principales en la programación orientada a objetos. La Enterprise Edition de Java ofrece una arquitectura notablemente rigurosa para la reutilización de componentes. Los componentes de empresa (llamados Enterprise JavaBeans) desarrollados en J2EE son de básica granulometría y reproducibles. Manteniendo estos componentes básicos, es posible crear funcionalidades de empresa complejas de un modo libremente coordinado. Además, ya que los componentes son reproducibles, lo cual quiere decir que es posible identificar ciertos metadatos sobre los componentes, las aplicaciones pueden ser creadas componiendo tales componentes. (Allamaraju, y otros)

En resumen se puede afirmar que: "La idea subyacente a la plataforma J2EE es proporcionar un estándar sencillo y unificado para aplicaciones distribuidas en un modelo de aplicación basado en componentes." (Allamaraju, y otros)

1.5.5. ENTORNO DE DESARROLLO

Se entiende como Entorno de Desarrollo Integrado, en inglés Integrated Development Environment (IDE), a un programa compuesto por un conjunto de herramientas para un programador, el cual forma un entorno de programación que ha sido empaquetado como un programa de aplicación; es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. (Universidad de Oviedo, 2003)

El proyecto Eclipse fue creado originalmente por IBM en el año 2001 (Eclipse Software Foundation). El mismo está dividido en tres sub-proyectos, los cuales son (Gallardo, y otros, 2003):

- La plataforma.
- El conjunto de herramientas para desarrollo en Java (JDT[1]).
- El Entorno de desarrollo de módulos (PDE[2]).

Eclipse es una plataforma universal para la integración de herramientas de desarrollo, esto es posible gracias a que cuenta con una arquitectura abierta y extensible basada en componentes añadidos (plug-ins).

De manera general, presenta las siguientes características (Berroa Álvarez, 2009):

- Es multiplataforma.
- Soporta distintas arquitecturas (x86, x64).

[1] JDT son las siglas en inglés para "Java Development Toolkit".
[2] PDE son las siglas en inglés para "Plug-in Development Environment".

- Posee una estructura de plug-in que hace sencillo añadir nuevas características y funcionalidades.
- Dispone de un control de versiones integrado a Subversion.
- Incluye muchas utilidades de edición que ayudan al programador a desarrollar el producto con rapidez, algunas son: resaltado de sintaxis, auto completamiento de código, tabulador de un bloque de código seleccionado y formateado automático de código, tributando a la obtención de código de mayor calidad.
- Posee asistentes para la creación, exportación e importación de proyectos y para generar esqueletos de códigos.

1.6. FRAMEWORKS UTILIZADOS PARA EL DESARROLLO.

El desarrollo de aplicaciones de gran envergadura como SIIPOL es un proceso que suele tornarse complejo, por lo que se hace necesario abstraer al programador de la mayor cantidad de detalles tecnológicos posibles, para que el mismo se concentre en las funcionalidades en sí que debe poseer el sistema. Además de esto, todo producto de software debe contar con una arquitectura tolerante al cambio. Por dichas razones se decidió utilizar para el desarrollo del SIIPOL los frameworks JavaServer Faces (JSF), Spring e Hibernate.

1.6.1. JAVASERVER FACES

Un framework define una estructura que puede servir para organizar y desarrollar un proyecto de software. Suelen incluir soporte de programas, bibliotecas y un lenguaje interpretado que ayudan a desarrollar e incluir los demás componentes de un proyecto. (Guevara Turruelles, 2009)

La tecnología JavaServer Faces es útil con aplicaciones basadas en la arquitectura Modelo – Vista – Controlador (MVC). Los principales componentes de esta tecnología son:

- Un interfaz de programación de aplicaciones (API por sus siglas en inglés) y una implementación de referencia para representar componentes de interfaces de usuario y manejar su estado; manejo de eventos, validación del lado del servidor y conversión de datos; definir la navegación entre páginas; soportar internacionalización y accesibilidad y además proporcionar extensibilidad para todas estas características.
- Una librería de etiquetas JavaServer Pages (JSP) personalizadas para dibujar los componentes de interfaz de usuario dentro de las páginas JSP.

Este modelo de programación bien definido y la librería de etiquetas para componentes de interfaz de usuario (IU) facilita de forma significativa la construcción y mantenimiento de aplicaciones web con IU del lado del servidor.

Existen varios términos que forman la base conceptual de JSF, estos son (Mann, 2005):

- Componente IU (También conocido como control o simplemente componente): Es un objeto mantenido en el servidor que provee funcionalidades específicas para interactuar con un usuario final. Estos son JavaBeans[3] con propiedades, métodos y eventos. Se organizan en una vista, que normalmente es un árbol de componentes visualizado como una página.
- Renderer: Es el responsable de mostrar los componentes IU y transformar las entradas del usuario en valores del componente.
- Validator (Validador): Su función es verificar que la entrada del usuario es un valor aceptable por el componente.
- Backing Beans (Beans de respaldo): JavaBeans especializados que almacenan los valores de los componentes IU e implementan métodos que reaccionan según los eventos lanzados por los componentes. También pueden contener referencias a componentes IU.
- Converter (Convertidor): Convierten valores desde y hacia los componentes en cadenas para mostrarlos al usuario. Un componente IU solo puede tener asociado un convertidor.
- Events and listeners (Eventos y escuchadores): JSF utiliza el modelo evento/escuchador de los JavaBeans. Los componentes IU y otros objetos generan eventos que son capturados y procesados por los escuchadores.
- Messages (Mensajes): Información que se le muestra de vuelta a los usuarios.
- Navigation (Navegación): Consiste en moverse de una página hacia otra. JSF tiene un potente sistema de navegación integrado con escuchadores de eventos especializados.

[3] Los JavaBeans son componentes de software reutilizables que se desarrollan e integran para crear aplicaciones más sofisticadas. Extraído de: http://www.oracle.com/technetwork/java/javase/tech/index-jsp-138795.html

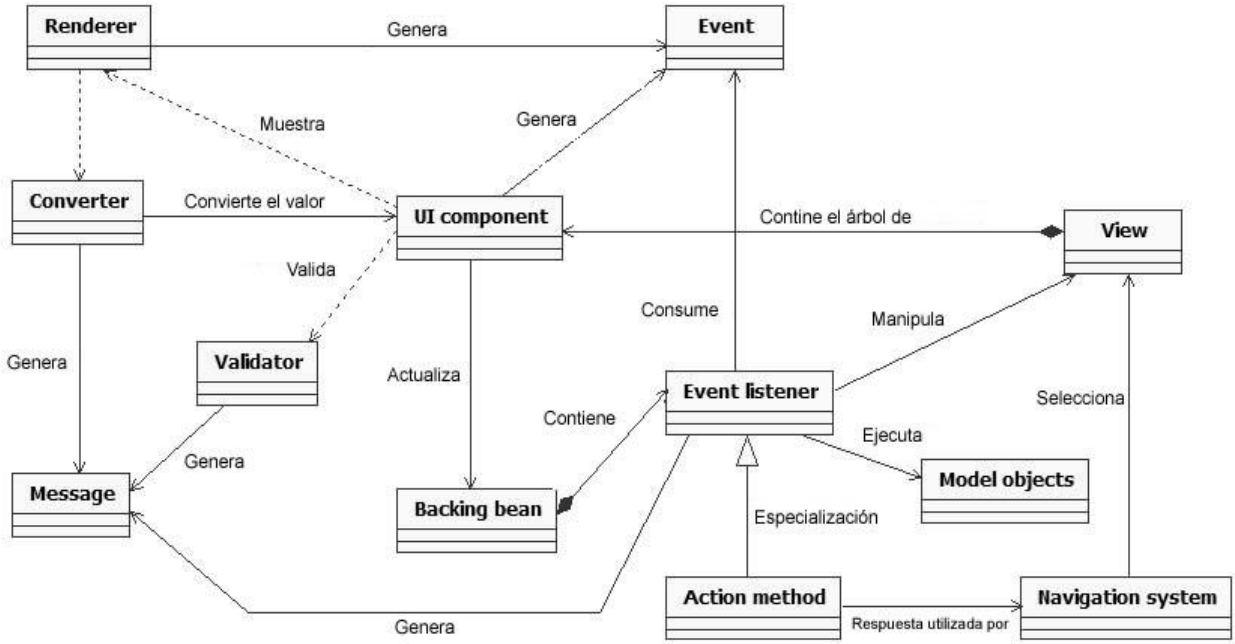

Fig 1.1 Modelo de Relación entre los términos de JSF.

Existen diferentes implementaciones de JSF, una de ellas es MyFaces, elaborada por Apache MyFaces Project perteneciente a la Apache Software Foundation. Esta implementación brinda (Apache Software Foundation):

- Implementación de JSF (API de MyFaces, módulos de implementación de MyFaces).
- Diferentes librerías de componentes que contienen etiquetas de componentes IU para el desarrollo de aplicaciones web con JSF (MyFaces Tomahawk).
- Paquetes de extensión para JavaServer Faces (MyFaces Extensions Validator).
- Módulos de integración con otras tecnologías y estándares (MyFaces Portlet Bridge).

1.6.2. SPRING

Spring es un framework de código abierto desarrollado para la plataforma J2EE. Por su diseño ofrece mucha libertad a los desarrolladores en Java, a la vez que proporciona soluciones muy bien documentadas y fáciles de usar para las prácticas comunes en la industria.

Spring promueve el bajo acoplamiento de las clases conformantes de la solución por medio de la técnica conocida como inversión del control (IoC). Soporta la Programación Orientada a Aspectos (AOP) que complementa la Programación Orientada a Objetos (POO), proponiendo otra manera de pensar sobre la estructura de un programa. Mientras que la POO descompone las aplicaciones en una jerarquía de objetos, la AOP descompone los programas en aspectos o

preocupaciones. Dichas preocupaciones se convierten en servicios del sistema, separados de la lógica de negocio y que se ejecutan de manera trasversal a la funcionalidad base, lo que proporciona una definición de responsabilidades superior. Spring básicamente es un contenedor, que se encarga de gestionar y administrar el ciclo de vida y configuración de las clases de la aplicación. (Hernández Vega, y otros, 2009)

Algunos de las características de Spring son (SpringSource Organization):

- Es considerado el framework ligero más completo que facilita una configuración y escritura de objetos muy centralizada y automatizada. No es muy agresivo a la arquitectura del proyecto, permite el montaje de complejos sistemas de componentes (POJOs[4]) de forma coherente y transparente. El framework brinda agilidad y mejora grandemente las capacidades de prueba de la aplicación.
- Contiene una capa común de abstracción para el manejo de las transacciones que permite conectar los administradores de las mismas, haciendo que sea más fácil para demarcar las operaciones sin tratar con bajo nivel de cuestiones. Estrategias genéricas para JTA[5] y conexiones simples de JDBC[6] están incluidas.
- Contiene una capa de abstracción de JDBC que ofrece una significativa jerarquía de excepciones, lo cual simplifica el tratamiento de errores y reduce enormemente la cantidad de código a escribir.
- Es integrable con Toplink, Hibernate, JDO (Java Data Object) e iBatis, a la vez que provee una magnífica jerarquía de excepciones para el caso de Hibernate.

[4] POJO: Plain Old Java Object, siglas utilizadas para enfatizar el uso de clases simples y que no dependen de un framework en especial.
[5] JTA: del inglés Java Transaction API. Establece una serie de interfaces Java entre el manejador de transacciones y las partes involucradas en el sistema de transacciones distribuidas. Es una especificación construida bajo el proceso de comunidad Java JSR 907.
[6] JDBC: del inglés Java Database Conectivity, es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java.

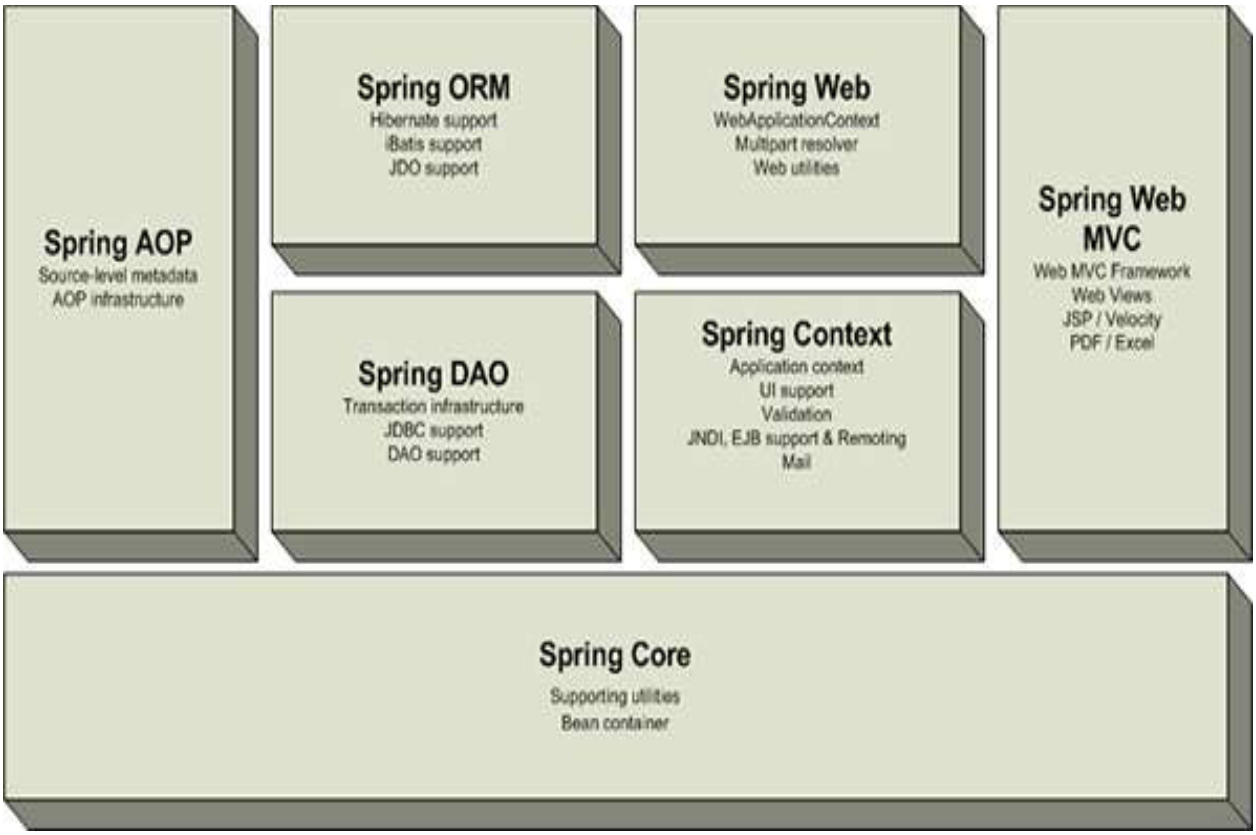

Fig 1.2 Arquitectura por módulos de Spring.

1.6.3. HIBERNATE

En el desarrollo de gran número de aplicaciones empresariales se hace necesario persistir la información manejada por el sistema en bases de datos. En este proceso se genera una gran cantidad de código repetitivo y que, además, es obligatorio (código boilerplate) para manejar las conexiones a la base de datos, generar consultas entre otros. Esto provoca que el código de las aplicaciones se vuelva engorroso y que los programadores se desconcentren de los problemas que realmente tienen que resolver de lógica del negocio por atender detalles de una tecnología en específico.

Para hacer frente a esta problemática surgen los Mapeadores Objeto Relacional (ORM por sus siglas en inglés). Hibernate es una solución ORM para Java y .Net que busca solucionar el problema de la diferencia entre el modelo orientado a objetos y el usado en las bases de datos

modelo relacional mediante archivos de configuración declarativos (XML[7]). A partir de la información proporcionada por el programador sobre su modelo de objetos en los archivos XML, Hibernate es capaz de generar consultas SQL[8], liberando así al programador de esta responsabilidad y permitiendo la eliminación de código boilerplate.

Este ORM también provee su propio lenguaje de consulta, el Hibernate Query Language (HQL); al mismo tiempo que ofrece una API para construir consultas programáticamente, conocida como Criteria.

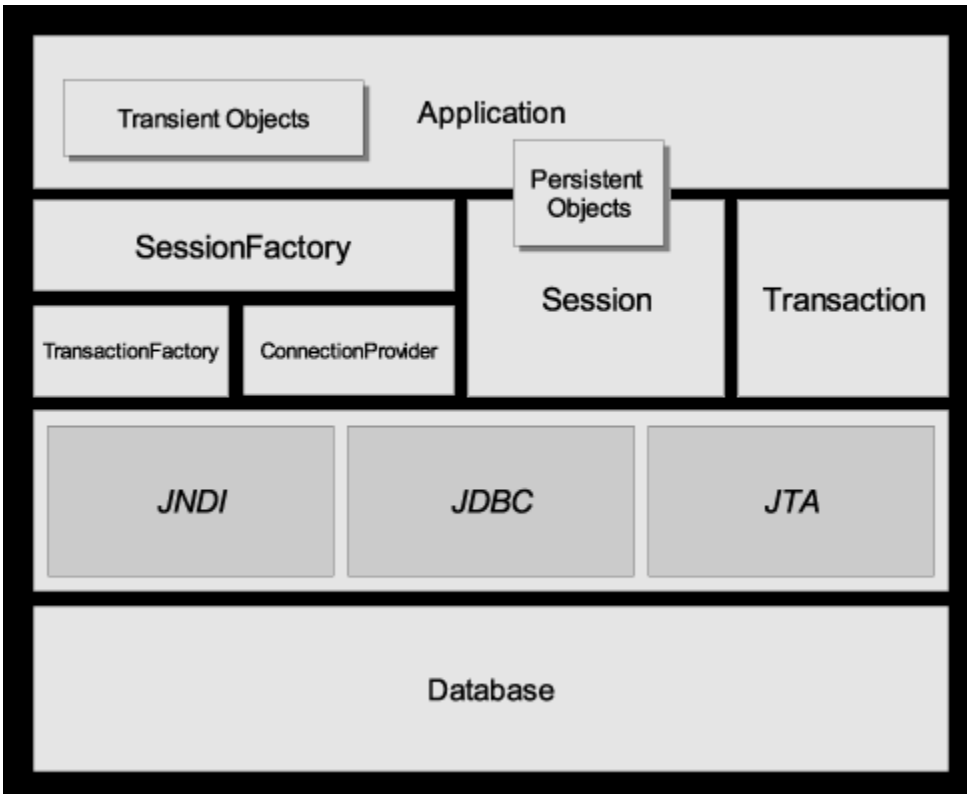

Fig 1.3 Arquitectura por capas de Hibernate

[7] XML: Extensible Markup Language – Lenguaje de Marcado Extensible. Es un lenguaje usado por los desarrolladores web y diseñadores para crear lenguajes de marcado declarativos. (Wiley Publishing, Inc.)

[8] SQL: Structured Query Language – Lenguaje estructurado de consulta. Lenguaje utilizado para realizar consultas y manipular datos en bases de datos relacionales. (Computer Language Company Inc.)

Hibernate permite incrementar la portabilidad de las aplicaciones con respecto a los distintos gestores de bases de datos, ya que basta con hacer un cambio en la librería que utiliza para generar el dialecto SQL y se podrá cambiar de gestor. A continuación se muestra una tabla con los dialectos soportados por Hibernate 3.2.0.ga (JBoss Organization).

Servidor Base de Datos	Clase del dialecto
DB2	org.hibernate.dialect.DB2Dialect
DB2 AS/400	org.hibernate.dialect.DB2400Dialect
DB2 OS390	org.hibernate.dialect.DB2390Dialect
PostgreSQL	org.hibernate.dialect.PostgreSQLDialect
MySQL	org.hibernate.dialect.MySQLDialect
MySQL con InnoDB	org.hibernate.dialect.MySQLInnoDBDialect
MySQL con MyISAM	org.hibernate.dialect.MySQLMyISAMDialect
Oracle (cualquier versión)	org.hibernate.dialect.OracleDialect
Oracle 9i/10g	org.hibernate.dialect.Oracle9Dialect
Sybase	org.hibernate.dialect.SybaseDialect
Sybase Anywhere	org.hibernate.dialect.SybaseAnywhereDialect
Microsoft SQL Server	org.hibernate.dialect.SQLServerDialect
SAP DB	org.hibernate.dialect.SAPDBDialect
Informix	org.hibernate.dialect.InformixDialect
HypersonicSQL (HSQLDB)	org.hibernate.dialect.HSQLDialect
Ingres	org.hibernate.dialect.IngresDialect
Progress	org.hibernate.dialect.ProgressDialect
Mckoi SQL	org.hibernate.dialect.MckoiDialect
Interbase	org.hibernate.dialect.InterbaseDialect
Pointbase	org.hibernate.dialect.PointbaseDialect
FrontBase	org.hibernate.dialect.FrontbaseDialect
Firebird	org.hibernate.dialect.FirebirdDialect

1.6.4. SISTEMA DE GESTIÓN DE BASES DE DATOS

Un Sistema Gestor de Bases de Datos (SGBD) o DBMA (DataBase Management System) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje

de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos.

Oracle es considerado uno de los SGBD más completos que existen hoy día. Surge al final de los años 70 y comienzos de los 80, actualmente es desarrollado y se le brinda soporte por la Oracle Corporation. Es un manejador de base de datos relacional que hace uso de los recursos del sistema informático en todas las arquitecturas de hardware, para garantizar su aprovechamiento al máximo en ambientes cargados de información. Además es una suite de productos que ofrece una gran variedad de herramientas. (Diaz Cabrera, y otros, 2009)

Algunas de las ventajas que proporciona Oracle son (Hernández Vega, y otros, 2009):

- Puede ejecutarse en todas las plataformas, desde una simple computadora personal hasta un supercomputador utilizado como servidor. El software del servidor puede ejecutarse en multitud de sistemas operativos.
- Oracle soporta todas las funciones que se esperan de un buen servidor de bases de datos: un lenguaje de diseño de bases de datos muy completo (PL/SQL, lenguaje de quinta generación, muy potente para tratar y gestionar la base de datos) que permite implementar diseños con disparadores y procedimientos almacenados, con una integridad referencial declarativa muy potente.
- Permite el uso de particiones para la mejora de la eficiencia, de replicación e incluso ciertas versiones admiten la administración de bases de datos distribuidas.

El principal inconveniente que presenta Oracle es el elevado costo de sus licencias y soporte, no obstante, debido a las ventajas expuestas anteriormente se decidió utilizar Oracle 10g como SGBD para el desarrollo del SIIPOL.

1.7 Conclusiones

A lo largo del capítulo que aquí concluye se han abordado conceptos relacionados con los sistemas de gestión de información y sistemas de información policial. También fueron expuestos algunos elementos de la metodología RUP, utilizada en el proceso de desarrollo del sistema SIIPOL. Herramientas utilizadas en el desarrollo de la presente solución de software como Eclipse, Visual Paradigm y frameworks como JavaServer Faces, Spring e Hibernate, se encuentran liderando la producción de software a nivel mundial; las mismas brindarán

numerosas facilidades en el desarrollo del módulo Debido Proceso, centro de la presente investigación.

CAPÍTULO 2. ANÁLISIS Y DISEÑO DE LA PROPUESTA DE SOLUCIÓN.

2.1 Introducción

En el proceso de desarrollo de software, se hace necesario analizar los requisitos solicitados por el cliente, a fin de implementar un sistema acorde a sus necesidades. En el presente capítulo, se presenta un análisis de la ingeniería de requisitos desarrollada por el equipo de analistas que especificó los casos de uso por los que se rige la implementación del SIIPOL. Se muestran el Modelo de Análisis y posteriormente el Modelo de Diseño, entre otros diagramas utilizados con el fin de modelar la estructura del subsistema Debido Proceso.

2.2 Requerimientos

En el flujo de trabajo de requerimientos se definen cuestiones esenciales para el desarrollo de productos de software. Uno de los artefactos fundamentales que se generan en este flujo de trabajo son las especificaciones de casos de uso. A continuación se muestran resúmenes de las especificaciones de casos de uso del subsistema Debido Proceso.

Caso de Uso	Gestionar Solicitud de Asignación de Abogado de Oficio
Propósito	Incluir, ver, modificar una Solicitud de Asignación de Abogado de Oficio.
Actores	Gestor de Solicitudes (Inicia): Incluye, registra, ve, modifica una Solicitud de Asignación de Abogado de Oficio.
Resumen	
El caso de uso inicia cuando el actor accede a la opción de realizar alguna acción sobre una Solicitud de Asignación de Abogado de Oficio. El actor puede incluir una nueva Solicitud de Asignación de Abogado de Oficio, ver los detalles de una solicitud ya existente o modificar una Solicitud de Asignación de Abogado de Oficio previamente incluida en el sistema. En caso que seleccione la opción de incluir una Solicitud de Asignación de Abogado de Oficio, es importante tener en cuenta que la solicitud se comporta como una Diligencia que sustancia el Expediente Disciplinario, independientemente de la forma en que se haya creado, el sistema permite introducir los datos correspondientes a la misma. Si el actor selecciona la opción de ver una Solicitud de Asignación de Abogado de Oficio, el sistema muestra el contenido de la misma, con posibilidad de imprimir y/o exportar a PDF. Si selecciona la opción de modificar la solicitud el sistema muestra aquellos datos editables de la misma, permitiendo realizar modificaciones en ellos. El caso de uso termina.	

Caso de Uso	Consultar Actas de Procedimiento Especial y/o Diferimiento
Propósito	Buscar y listar de manera ordenada un resumen de los datos de un Acta de Procedimiento Especial y/o Diferimiento, coincidentes con uno o varios criterios de búsqueda.
Actores	Consultor (Inicia): Consulta un listado ordenado de Actas de Procedimiento Especial y/o Diferimiento.
Resumen	
El caso de uso se inicia cuando el actor accede a la opción que le permite consultar un Acta de Procedimiento Especial y/o Diferimiento, el sistema permite especificar los criterios de búsqueda y luego lista las posibles coincidencias. El sistema brinda la posibilidad atendiendo a los permisos del usuario de generar una Notificación de Sanción a partir de un Acta de Procedimiento Especial e imprimir y/o exportar el listado de Actas a PDF. El caso de uso termina.	

Caso de Uso	Gestionar Acta de Procedimiento Especial
Propósito	Incluir, ver o modificar un Acta de Procedimiento Especial.
Actores	Gestor de Actas (Inicia): Incluye, ve o modifica un Acta Procedimiento Especial.
Resumen	
El caso de uso inicia cuando el actor selecciona la opción de realizar una acción sobre un Acta de Procedimiento Especial. El actor puede incluir, ver o modificar un Acta de Procedimiento Especial. En caso de que seleccione la opción de incluir un Acta de Procedimiento Especial, el sistema permite insertar los datos que se necesitan para registrar la misma. Si el actor elige la opción de ver un Acta de Procedimiento Especial, el sistema muestra el contenido del Acta en cuestión. Si el actor elige la opción de modificar el Acta de Procedimiento Especial, el sistema muestra los datos que pueden ser editables dentro de la misma y una vez realizados los cambios, guarda las modificaciones. El sistema permite ver una vista previa del Acta de Procedimiento Especial, con posibilidad de imprimir y/o exportar a PDF. El caso de uso termina.	

Caso de Uso	Gestionar Acta de Diferimiento
Propósito	Incluir, ver o modificar un Acta de Diferimiento.
Actores	Gestor de Actas (Inicia): Incluye, ve o modifica un Acta de Diferimiento.

Resumen
El caso de uso inicia cuando el actor selecciona la opción de realizar una acción sobre un Acta de Diferimiento. El actor puede incluir, ver o modificar un Acta de Diferimiento. En caso de que seleccione la opción de incluir un Acta de Diferimiento, el sistema permite insertar los datos que se necesitan para registrar la misma. Si el actor elige la opción de ver un Acta de Diferimiento, el sistema muestra el contenido del Acta en cuestión. Si el actor elige la opción de modificar el Acta de Diferimiento, el sistema muestra los datos que pueden ser editables dentro de la misma y una vez realizados los cambios, guarda las modificaciones. El sistema permite ver una vista previa del Acta de Diferimiento, con posibilidad de imprimir y/o exportar a PDF. El caso de uso termina.

Caso de Uso	Gestionar Boleta de Notificación
Propósito	Incluir, ver o modificar una Boleta de Notificación.
Actores	Gestor de Boleta de Notificación (Inicia): Incluye, ve o modifica una Boleta de Notificación.
Resumen	
El caso de uso inicia cuando el actor selecciona la opción de realizar una acción sobre una Boleta de Notificación. El actor puede incluir, ver o modificar una Boleta de Notificación. En caso de que seleccione la opción de incluir una Boleta de Notificación, el sistema permite insertar los datos que se necesitan para registrar la Boleta de Notificación. Si el actor elige la opción de ver una Boleta de Notificación, el sistema muestra el contenido de la Boleta en cuestión. Si el actor elige la opción de modificar una Boleta de Notificación, el sistema muestra los datos que pueden ser editables dentro de la misma y una vez realizados los cambios dentro de la misma, guarda las modificaciones. El sistema permite ver una vista previa de una Boleta de Notificación, con posibilidad de imprimir y/o exportar a PDF. Termina el caso de uso.	

Caso de Uso	Gestionar Escrito de Consideración
Propósito	Incluir, ver o modificar un Escrito de Consideración.
Actores	Gestor de Escrito de Consideración (Inicia): Incluye, ve o modifica un Escrito de Consideración.
Resumen	
El caso de uso inicia cuando el actor accede a realizar alguna acción sobre un Escrito	

de Consideración. El actor puede incluir un nuevo Escrito de Consideración, ver los detalles de un Escrito de Consideración ya existente o modificar un Escrito de Consideración previamente incluido en el sistema. En caso que se seleccione la opción de incluir un Escrito de Consideración el sistema permite introducir los datos correspondientes al mismo. Si el actor selecciona la opción de ver un Escrito de Consideración, el sistema muestra el contenido del mismo, con posibilidad de imprimir y/o exportar a PDF. Si selecciona la opción de modificar el Escrito de Consideración el sistema muestra aquellos datos editables del mismo, permitiendo realizar modificaciones en ellos y guardarlas en el sistema. El caso de uso termina.

Caso de Uso	Gestionar Excusa
Propósito	Incluir, ver o modificar una Excusa.
Actores	Gestor de Excusa (Inicia): Incluye, ve o modifica una Excusa.
Resumen	
El caso de uso inicia cuando el actor accede a la opción de realizar alguna acción sobre una Excusa. El actor puede incluir una nueva Excusa, ver los detalles de una Excusa ya existente o modificar una Excusa previamente incluida en el sistema. En caso que seleccione la opción de incluir una Excusa el sistema permite introducir los datos correspondientes a la misma. Si el actor selecciona la opción de ver una Excusa, el sistema muestra el contenido de la misma, con posibilidad de imprimir y/o exportar a formato PDF. Si selecciona la opción de modificar la Excusa el sistema muestra aquellos datos editables de la misma, permitiendo realizar modificaciones en ellos y guardarlas en el sistema. El caso de uso termina.	

Caso de Uso	Gestionar Notificación de Sanción
Propósito	Incluir, ver o modificar una Notificación de Sanción.
Actores	Gestor de Notificación (Inicia): Incluye, ve o modifica una Notificación de Sanción.
Resumen	
El caso de uso inicia cuando el actor accede a realizar alguna acción sobre una Notificación de Sanción. El actor puede incluir una nueva Notificación de Sanción, ver los detalles de una Notificación ya existente o modificar una Notificación de Sanción previamente incluida en el sistema. En caso que se seleccione la opción de incluir una	

Notificación de Sanción, el sistema permite introducir los datos correspondientes a la misma. Si el actor selecciona la opción de ver una Notificación de Sanción, el sistema muestra el contenido de la misma, con posibilidad de imprimir y/o exportar a formato PDF. Si selecciona la opción de modificar la Notificación de Sanción, el sistema muestra aquellos datos editables de la misma, permitiendo realizar modificaciones en ellos y guardarlas en el sistema. El caso de uso termina.

Caso de Uso	Gestionar Recurso Jerárquico
Propósito	Incluir, registrar, ver o modificar un Recurso Jerárquico.
Actores	Gestor de Recurso Jerárquico (Inicia): Incluye, registra ve o modifica un Recurso Jerárquico.
Resumen	
El caso de uso inicia cuando el actor accede a la opción de realizar alguna acción sobre un Recurso Jerárquico. El actor puede incluir un nuevo Recurso Jerárquico, ver los detalles de un Recurso Jerárquico ya existente o modificar un Recurso Jerárquico previamente incluido en el sistema. En caso que seleccione la opción de incluir un Recurso Jerárquico el sistema permite introducir los datos correspondientes al mismo. Si el actor selecciona la opción de ver un Recurso Jerárquico, el sistema muestra el contenido del mismo, con posibilidad de imprimir y/o exportar a PDF. Si selecciona la opción de modificar el Recurso Jerárquico, el sistema muestra aquellos datos editables del mismo, permitiendo realizar modificaciones en ellos y guardarlas en el sistema. El caso de uso termina.	

Caso de Uso	Gestionar Solicitud de Revisión de Expediente Disciplinario
Propósito	Incluir, registrar, ver o modificar una Solicitud de Revisión de Expediente Disciplinario.
Actores	Gestor de Solicitudes (Inicia): Incluye, registra, ve, modifica una Solicitud de Revisión de Expediente Disciplinario.
Resumen	
El caso de uso inicia cuando el actor accede a realizar alguna acción sobre una Solicitud de Revisión de Expediente Disciplinario. El actor puede incluir una nueva Solicitud de Revisión de Expediente Disciplinario, ver los detalles de una Solicitud ya existente o modificar una Solicitud de Revisión de Expediente previamente incluida en	

el sistema, también se le permite al usuario eliminar una Solicitud de Revisión de Expediente Disciplinario. En caso que se seleccione la opción de incluir una Solicitud de Revisión de Expediente Disciplinario, el sistema permite introducir los datos correspondientes a la misma. Si el actor selecciona la opción de ver una Solicitud de Revisión de Expediente Disciplinario, el sistema muestra el contenido de la misma, con posibilidad de Imprimir y/o Exportar a formato PDF. Si selecciona la opción de modificar la Solicitud de Revisión de Expediente Disciplinario, el sistema muestra aquellos datos editables de la misma, permitiendo realizar modificaciones en ellos. El caso de uso termina.

Caso de Uso	Gestionar Solicitud de Representación Legal en Procedimiento Especial
Propósito	Incluir, ver y registrar una Solicitud de Representación Legal en Procedimiento Especial.
Actores	Gestor de Solicitudes (Inicia): Incluye, registra o ve una Solicitud de Representación Legal en Procedimiento Especial.
Resumen	
El caso de uso inicia cuando el actor selecciona la opción de realizar una acción sobre una Solicitud de Representación Legal en Procedimiento Especial. El actor puede incluir, ver o registrar una Solicitud de Representación Legal en Procedimiento Especial. En caso de que seleccione la opción de incluir una Solicitud de Representación Legal en Procedimiento Especial, el sistema permite incluir y seleccionar los datos de la solicitud. Si el actor desea registrar una Solicitud de Representación Legal en Procedimiento Especial, el sistema muestra los datos predeterminados de la comunicación asociada y permite introducir y seleccionar los datos de la solicitud. Si el actor elige la opción de ver una Solicitud de Representación Legal en Procedimiento Especial, el sistema muestra los datos de la solicitud en cuestión y permite imprimir y/o exportar a PDF sus datos. El caso de uso termina.	

2.3 Modelo de Análisis

El Modelo de Análisis es un modelo objetual que describe la realización de casos de uso y sirve como abstracción del Modelo de Diseño. El mismo contiene los resultados del análisis de los casos de uso expresados en clases del análisis. (Rational Corporation, 2003)

La metodología RUP establece una serie de estereotipos para las clases del análisis, uno de ellos es el que se muestra en la figura 2.1 y que se aplica a las interfaces con las que el usuario interactúa.

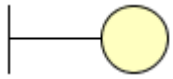

Fig 2.1 Clase Interfaz

Las clases controladoras, reflejadas en la figura 2.2, son las encargadas de almacenar la lógica de la aplicación.

Fig 2.2 Clase Controladora

La información que va a ser persistida durante largo tiempo en las bases de datos se modela, en el mundo objetual, como clases entidad. El estereotipo usado para las mismas se muestra en la figura 2.3.

Fig 2.3 Clase Entidad

A continuación se muestran los diagramas de clases del análisis de los casos de uso del módulo Debido Proceso.

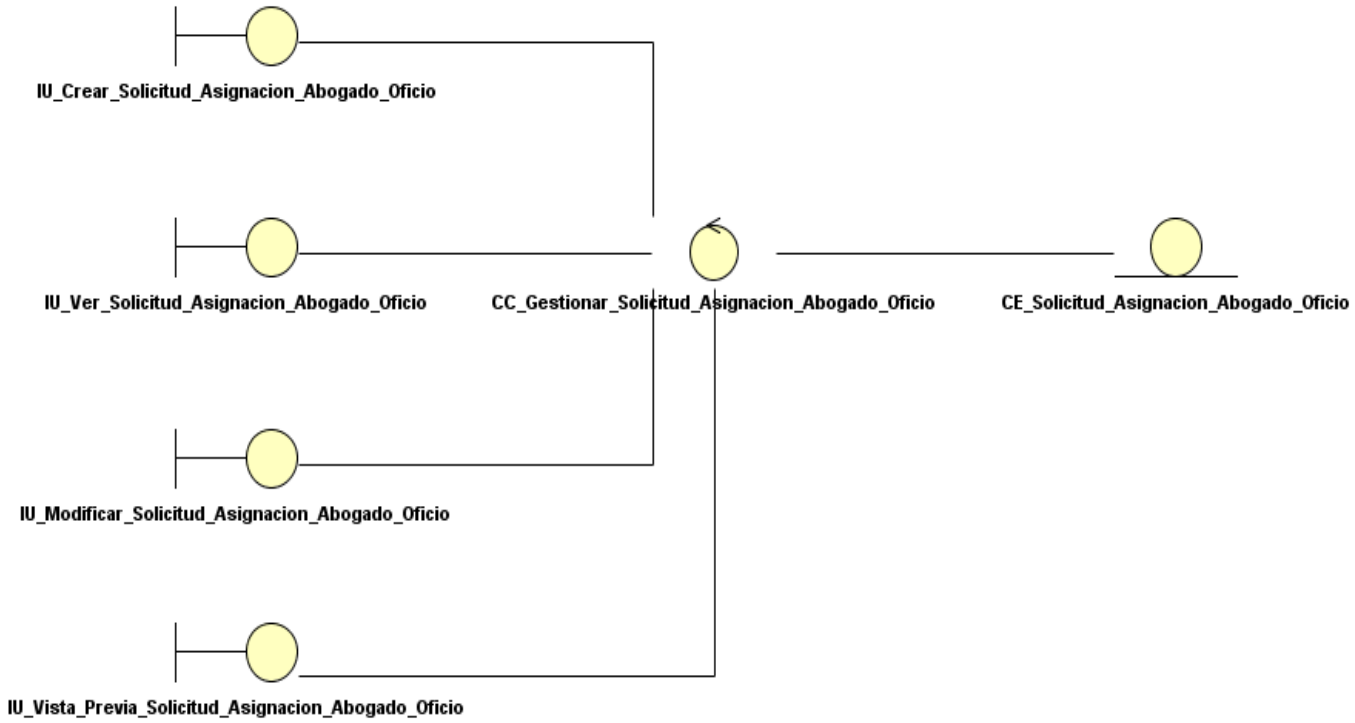

Fig 2.4 CU Gestionar Solicitud Abogado Oficio

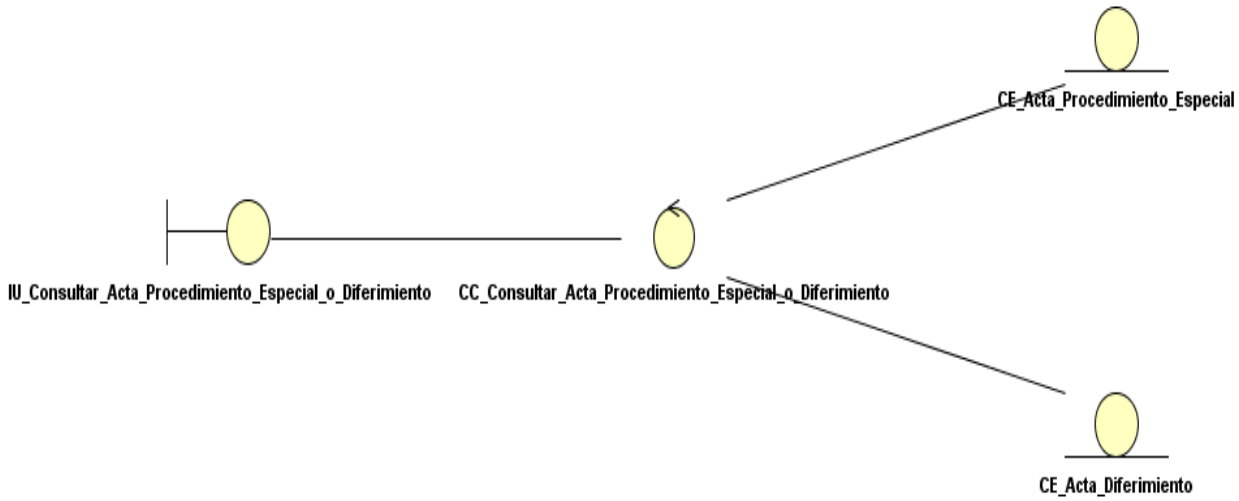

Fig 2.5 CU Consultar Acta Procedimiento Especial o Diferimiento

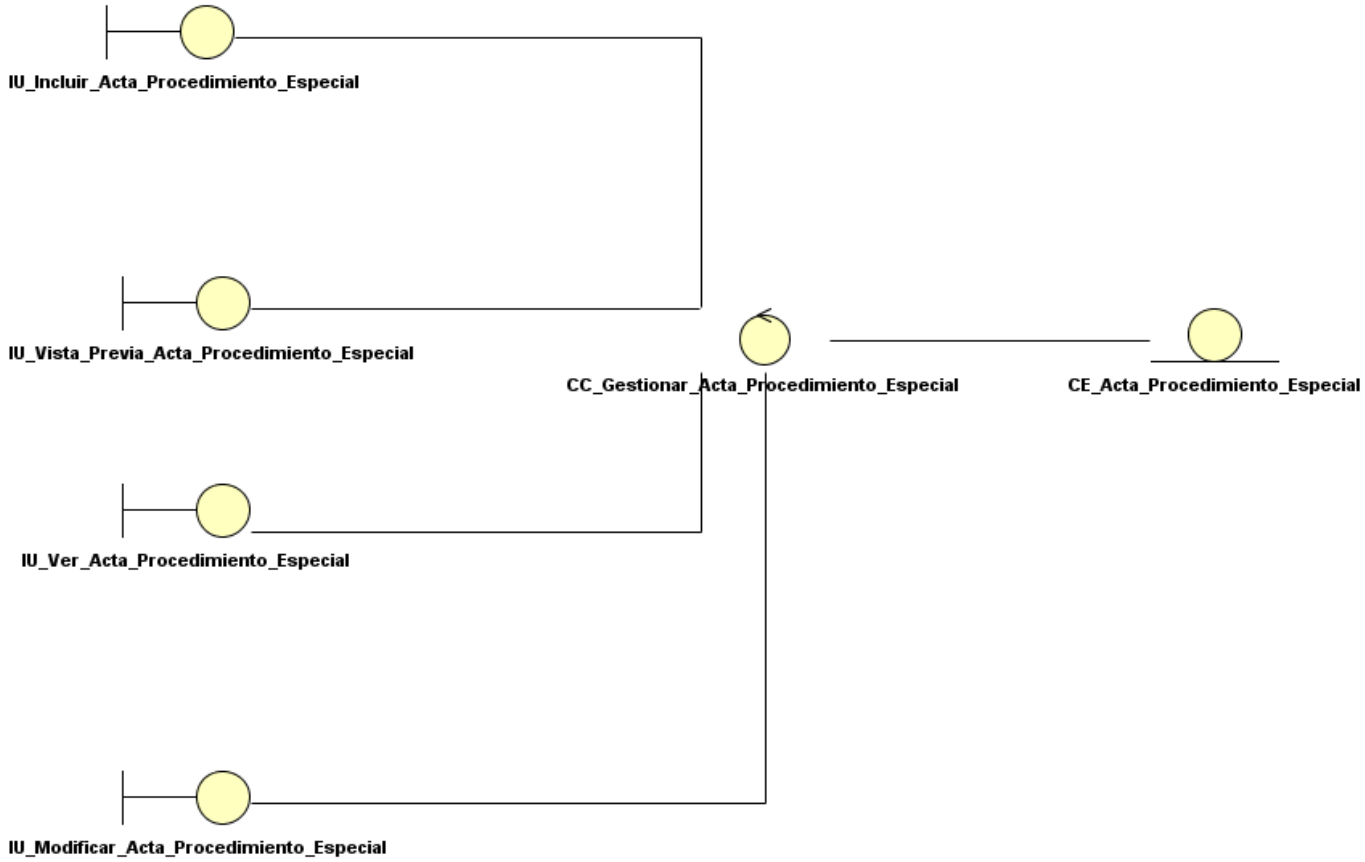

Fig 2.6 CU Gestionar Acta de Procedimiento Especial

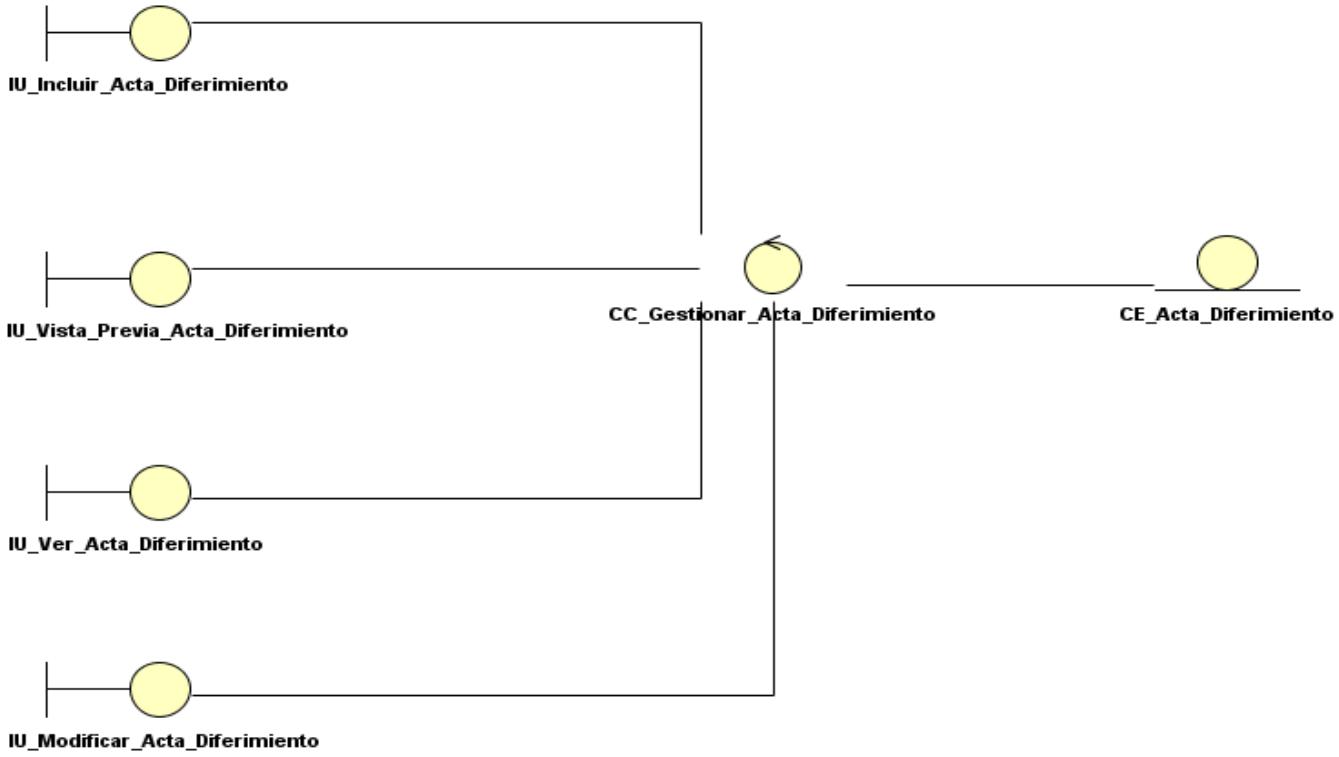

Fig 2.7 CU Gestionar Acta de Diferimiento

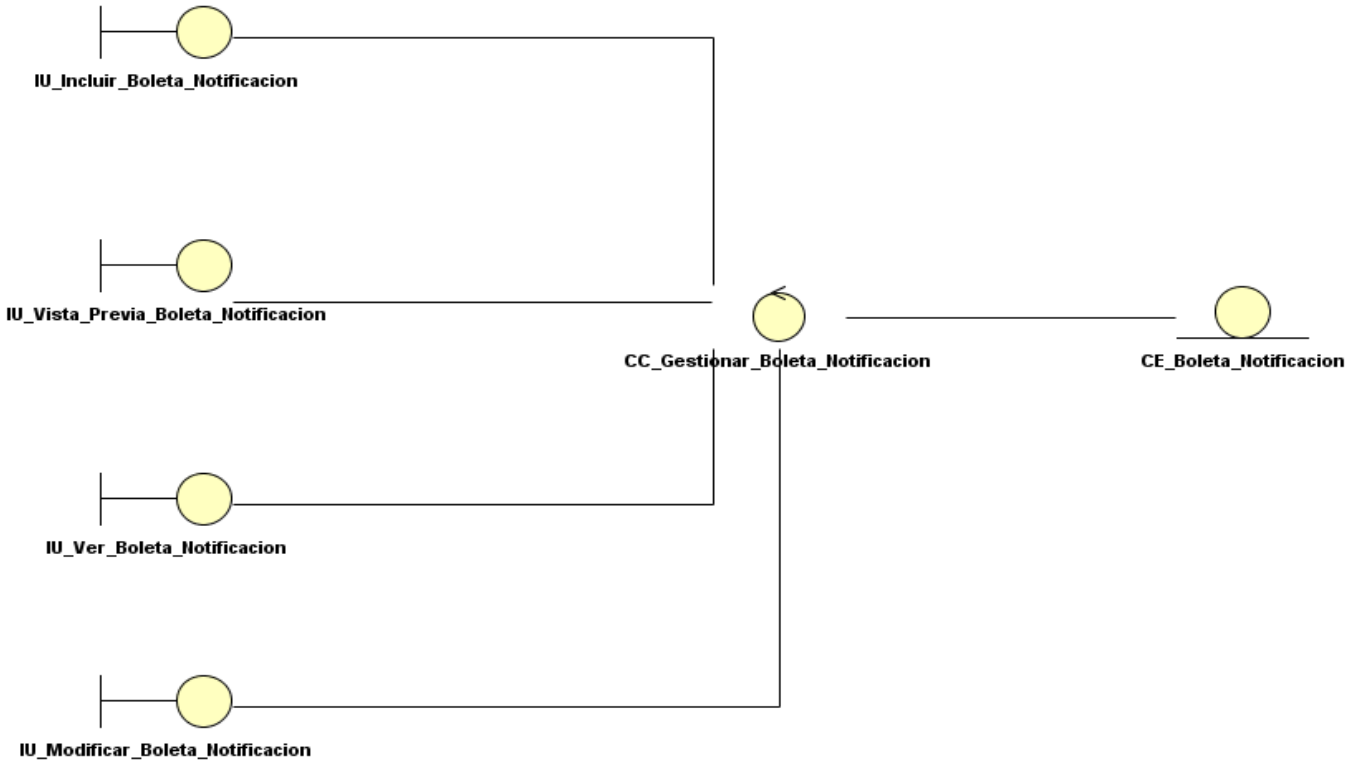

Fig 2.8 CU Gestionar Boleta de Notificación

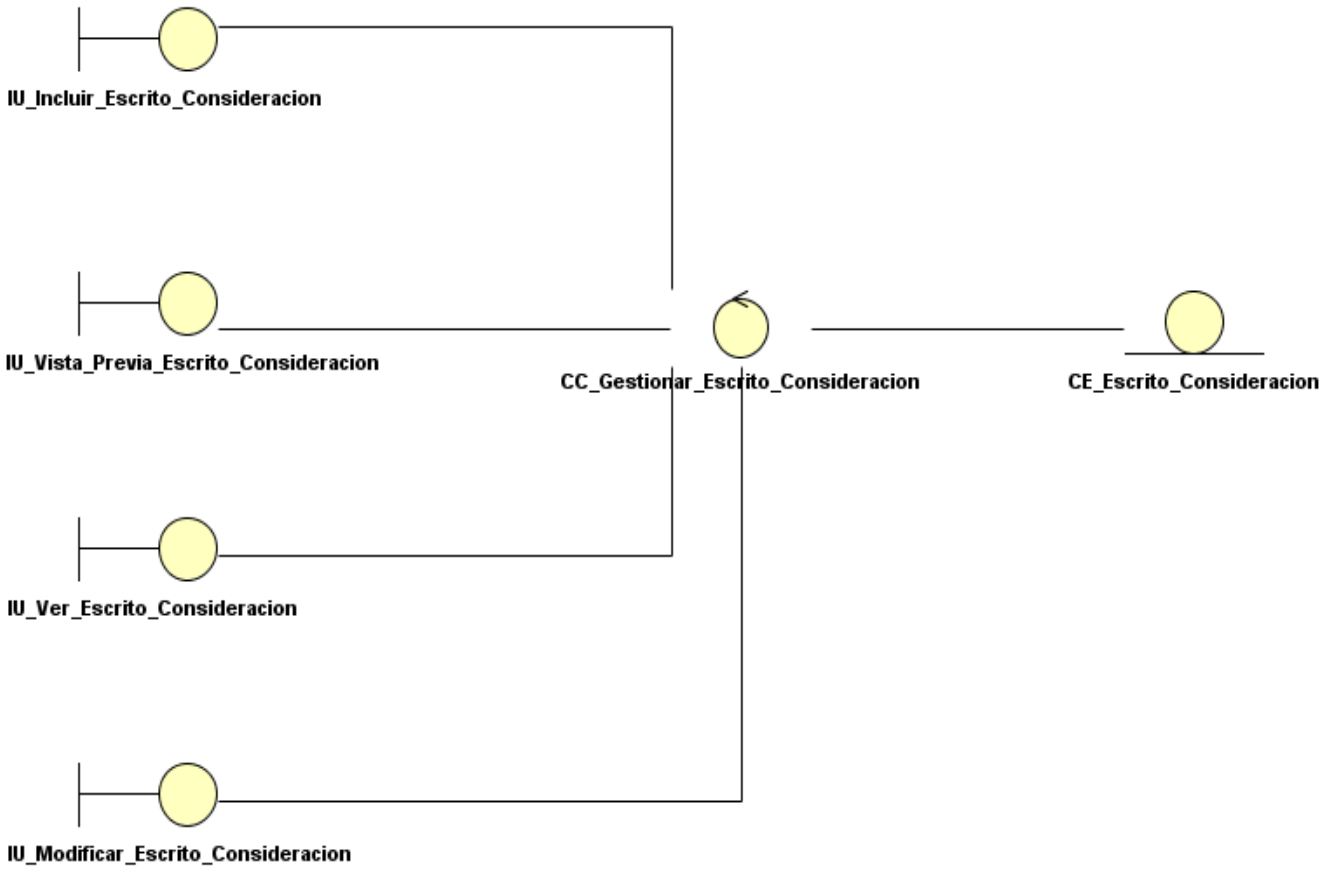

Fig 2.9 CU Gestionar Escrito de Consideración

2.4 Modelo de Diseño

La metodología RUP plantea que el modelo de diseño es un modelo objetual que describe la realización de los casos de uso, y sirve como una abstracción del modelo de implementación y del código fuente. (Rational Corporation, 2003)

Dicho modelo se compone de once artefactos que se mencionan a continuación (Rational Corporation, 2003):

1. Protocolo.
2. Cápsula.
3. Realización de Casos de Uso.
4. Señal.
5. Evento.
6. Subsistemas del Diseño.
7. Paquetes del Diseño.

8. Interfaces.
9. Clases del Diseño.
10. Clases de Prueba.
11. Diseño de Pruebas.

En el presente trabajo serán descritos aquellos artefactos del modelo de diseño que fueron utilizados en el desarrollo del subsistema Debido Proceso.

2.4.1 Paquetes del Diseño

Un paquete de diseño es una colección de clases, relaciones, realizaciones de casos de usos, diagramas y otros paquetes. Se utiliza para estructurar el modelo de diseño dividiéndolo en pequeñas partes. (Rational Corporation, 2003)

Para el desarrollo de este artefacto a continuación se detallan el diagrama de paquetes, las clases del diseño y la realización de casos de uso.

2.4.1.1 Diagrama de Paquetes

Los paquetes reflejan la arquitectura de alto nivel de un sistema: su descomposición en subsistemas y sus dependencias. Están organizados de manera funcional, siguiendo un cierto principio racional, tal como funcionalidad común, implementación estrechamente relacionada, y un punto de vista común. Una dependencia entre paquetes resume las dependencias entre los contenidos del paquete.

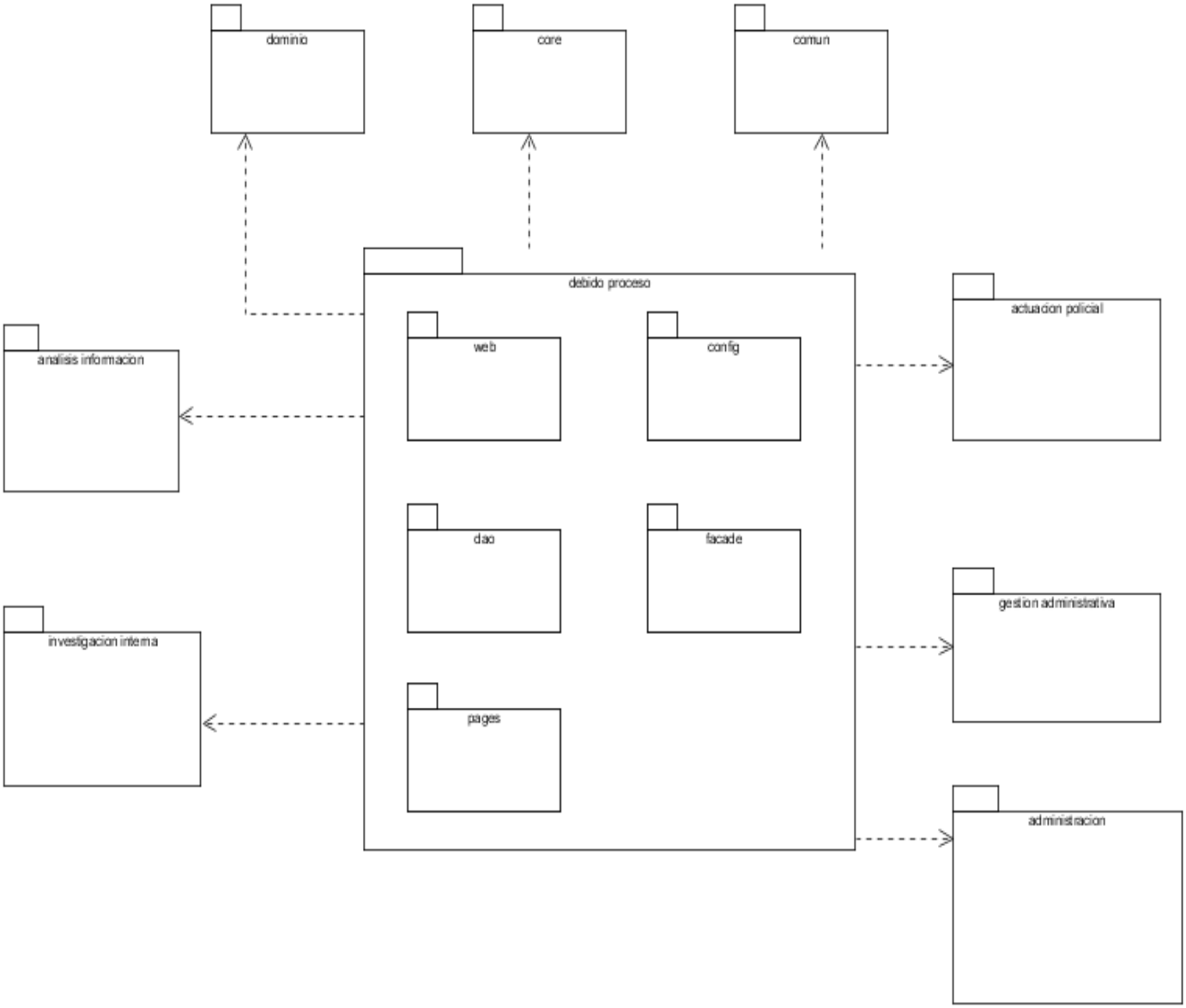

Fig 2.10 Diagrama de paquetes del diseño pertenecientes al subsistema Debido Proceso

2.4.1.2 CLASES DEL DISEÑO

En UML una clase es la descripción de un conjunto de objetos que comparten las mismas responsabilidades, relaciones, operaciones y atributos. (Rational Corporation, 2003)

Las clases del diseño son representaciones de las clases utilizadas en la implementación del sistema, en ellas se describen las responsabilidades de los métodos implicados en el desarrollo de una funcionalidad determinada. Para estructurar las clases del diseño y asignarles las

responsabilidades se utilizaron patrones de diseño que permiten lograr una arquitectura consistente.

Un patrón de diseño es una estructura de clases recomendada que ha sido probada exitosamente en múltiples entornos de desarrollo para resolver situaciones determinadas. (Kuchana, 2004)

Para el diseño e implementación del subsistema Debido proceso se utilizaron patrones GRASP[9], creacionales y estructurales. A continuación se da una breve descripción de los mismos.

Modelo Vista Controlador (MVC): Es un principio de diseño arquitectónico que separa los componentes de una aplicación web. Esta separación ofrece más control sobre las partes individuales de la aplicación, lo cual permite desarrollarlas, modificarlas y probarlas más fácilmente. Los componentes se separan en tres categorías: modelos, para almacenar los datos; vistas, para mostrar los datos almacenados y controlador para manejar los eventos que afectan a las vistas o a los modelos. Este patrón es implementado por el framework JavaServer Faces.

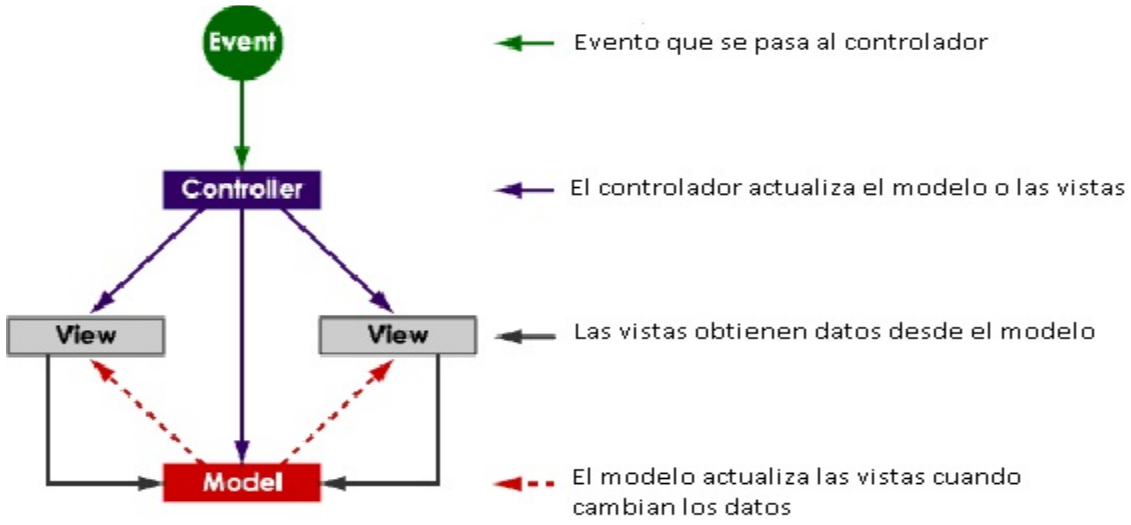

Fig 2.11 Diagrama de funcionamiento del Modelo Vista Controlador

[9] Acrónimo de General Resposability Assignment Software Patterns, en español Patrones de Software para la Asignación General de Responsabilidades. (Hurtado Bustamante, et al.)

Patrón N Capas: Una capa es una separación lógica del software que permite dividir más fácilmente las responsabilidades con respecto al sistema. El sistema SIIPOL es una aplicación en tres capas, presentación, negocio y acceso a datos.

Facade: Es un patrón estructural cuya intención es proporcionar una interfaz unificada de alto nivel que, representando a todo un subsistema, facilite su uso. Esto permite ocultar la lógica de un subsistema y garantiza la comunicación de otros subsistemas a través de métodos transparentes para los subsistemas externos.

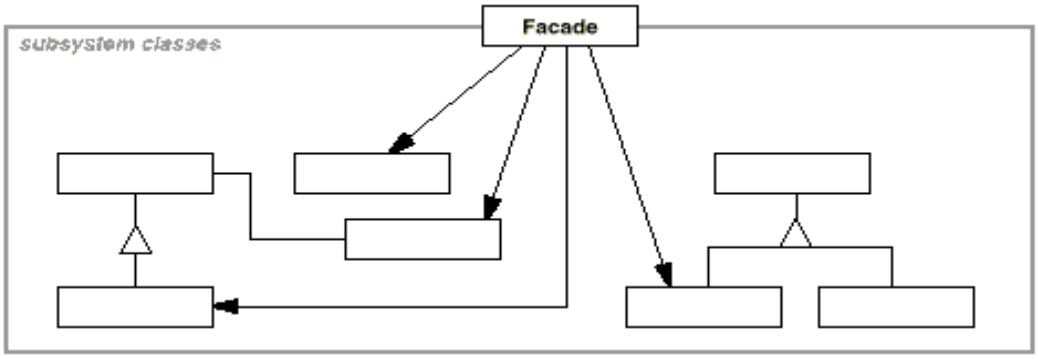

Fig 2.12 Diagrama del patrón Facade

Data Access Object (DAO): El Objeto de Acceso a Datos es un patrón que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una base de datos o un archivo. Permite abstraer y encapsular los accesos, gestionar las conexiones a la fuente de datos y obtener los datos almacenados.

En la presente investigación se utilizan los patrones N Capas, Facade y DAO debido a que son orientados por la arquitectura del Sistema SIIPOL, además de las ventajas que ofrecen.

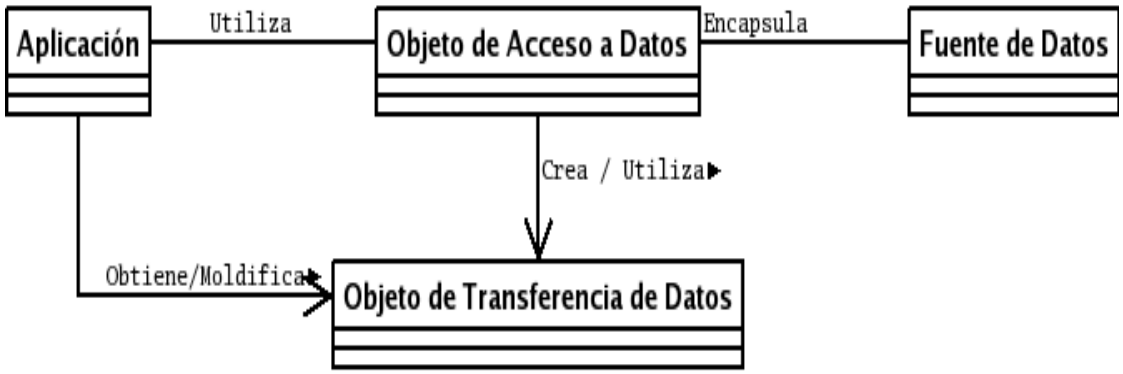

Fig 2.13 Diagrama del patrón DAO

Inyección de dependencias: Este patrón de diseño se basa en que el objeto define sus dependencias, es decir, los atributos con los cuales trabaja y provee formas de acceder a ellos, pero no los construye. Luego, el contenedor de dichos objetos es el responsable de construir estas dependencias y asignárselas (inyectárselas) a los objetos que las definen. En esto es particularmente útil el framework Spring que permite inyección de dependencias utilizando la inversión de control.

Bajo acoplamiento: Es un patrón GRASP que permite mantener las clases los más bajamente acopladas posible. A la realización de este patrón contribuye la inyección de dependencias mencionada anteriormente.

Polimorfismo: Es un patrón GRASP que permite a distintos objetos de un mismo tipo tener comportamientos diferentes. En el desarrollo del módulo Debido Proceso se utilizó principalmente para encapsular la navegación entre las interfaces de usuario dentro de los beans de respaldo responsables de suministrar datos a dichas interfaces.

Singleton: Es un patrón creacional que permite obtener siempre que se solicite un objeto de una clase determinada, obtener la misma instancia. Este patrón lo utiliza particularmente el framework JavaServer Faces a la hora de devolver instancias de un bean manejado en una misma sesión.

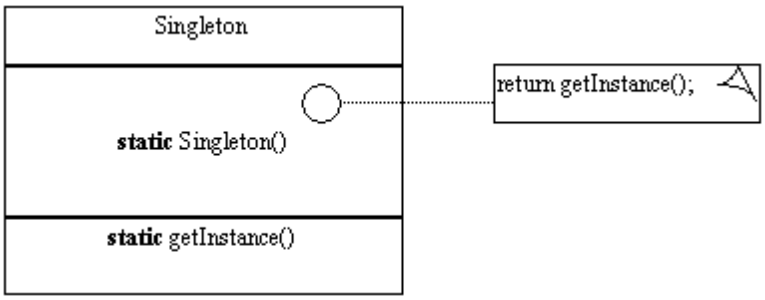

Fig 2.14 Diagrama del patrón Singleton

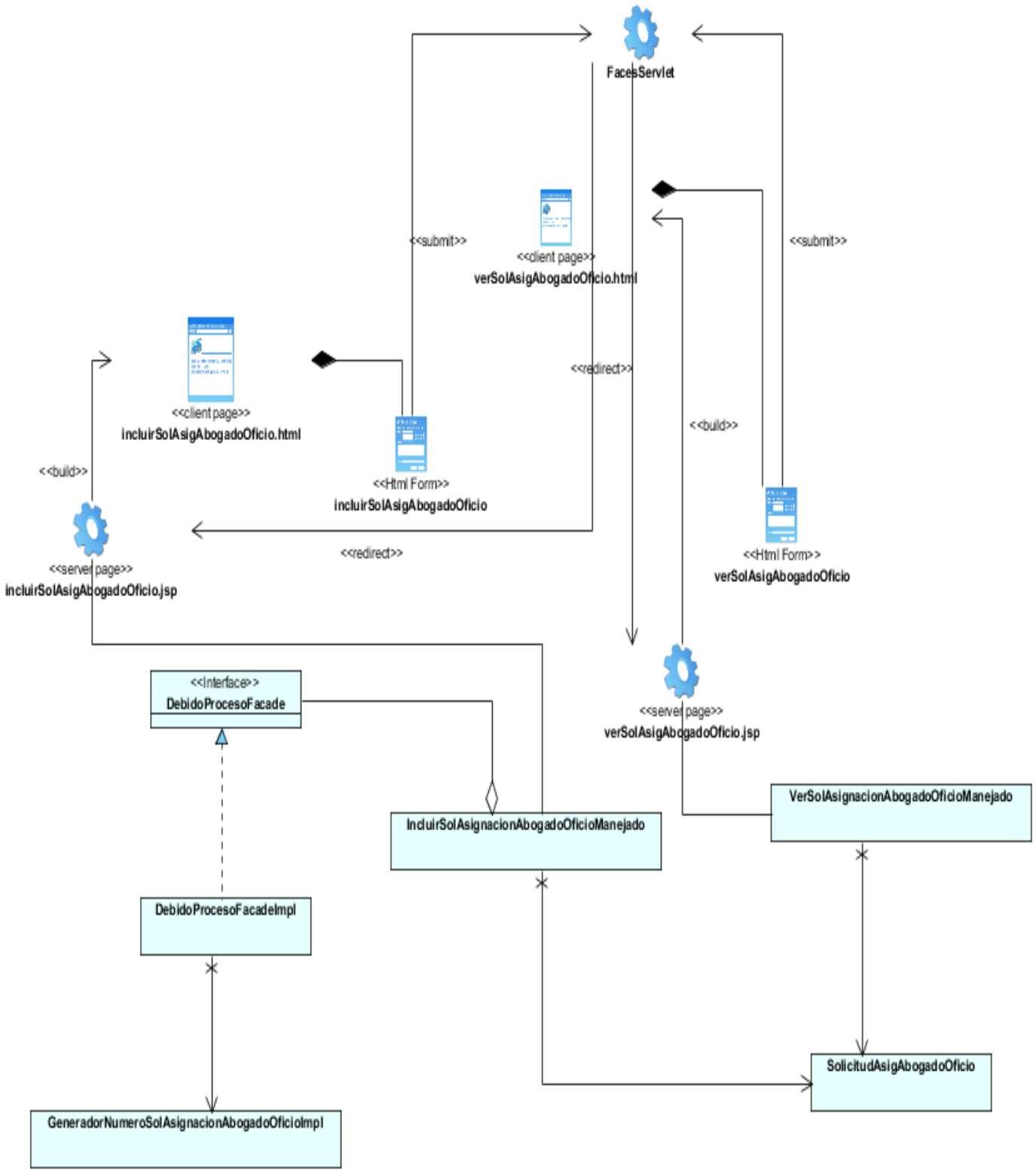

Fig 2.15 Diagrama de Clases de Diseño CU Gestionar Solicitud de Asignación de Abogado de Oficio. Capa de Presentación

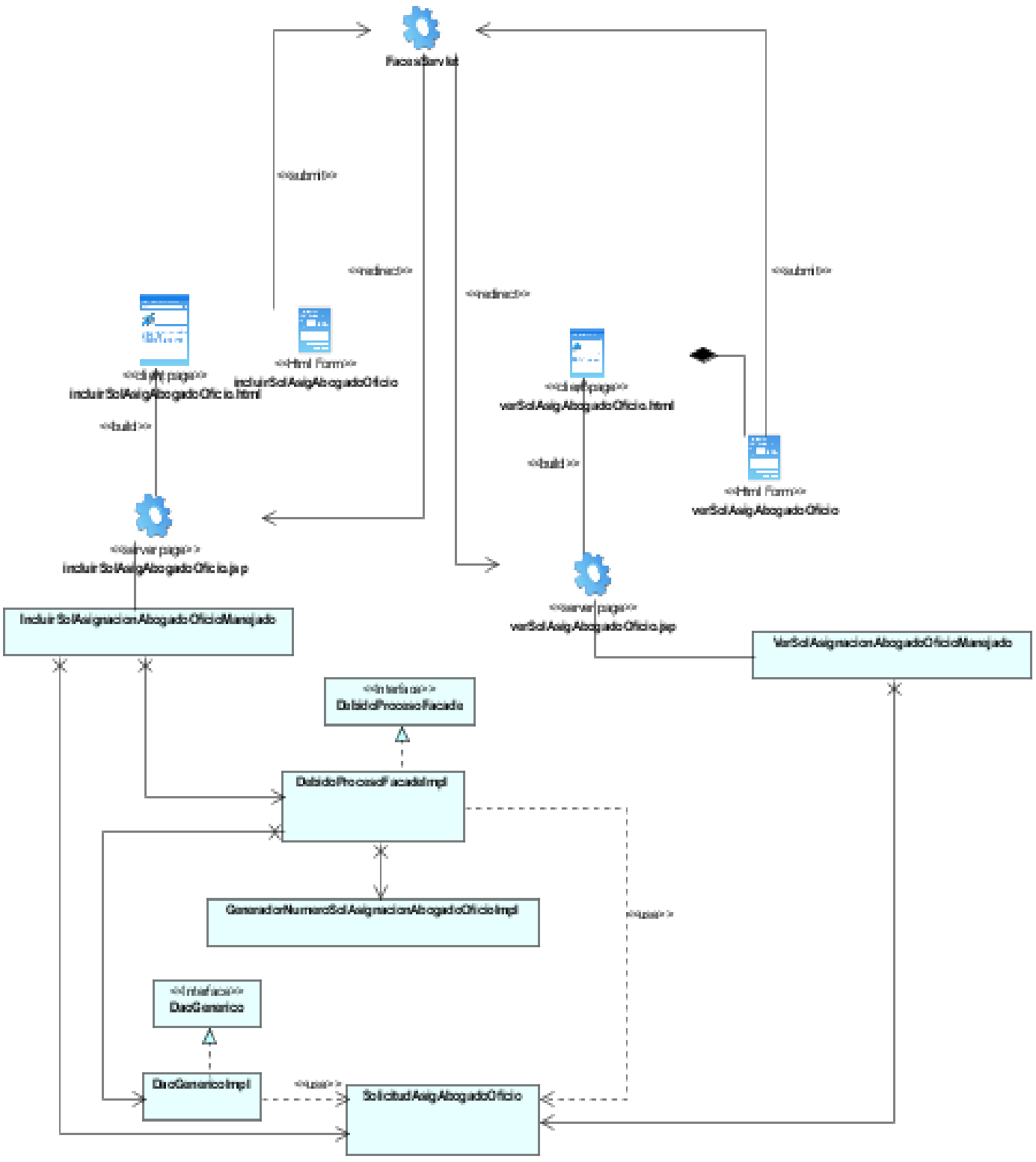

Fig 2.16 Diagrama de Clases de Diseño CU Gestionar Solicitud de Asignación de Abogado de Oficio.

A continuación se describen textualmente algunas de las clases más importantes del diseño.

Nombre	IncluirSolAsignacionAbogadoOficioManejado
Tipo de Clase	Bean de Respaldo

Atributo	**Tipo**
solicitudAsigAbogadoOficio	SolicitudAsigAbogadoOficio
importancia	Nomenclador
descripción	String
fundamentoLegal	String
modificando	Boolean
funcionarioAsociadosExpedienteDisciplinario	List<SelectableItemDTO<Funcionario>>
debidoProcesoFacade	DebidoProcesoFacade
correspondenciaFacade	CorrespondenciaFacade
genericRedirect	GenericRedirect

Para cada Responsabilidad	
Nombre	configuracionesIniciales()
Descripción	Método encargado de establecer las configuraciones iniciales cuando se crea el bean y cuando el usuario manda a limpiar todos los campos.
Nombre	limpiar(ActionEvent e)
Descripción	Método encargado de borrar todos los campos.
Nombre	asociarExpedienteDisciplinario(ActionEvent e)
Descripción	Método encargado de asociar un expediente disciplinario a la solicitud.
Nombre	construirSolicitud()
Descripción	Método encargado de construir la solicitud una vez que se han especificado todos los datos necesarios.
Nombre	incluirSolicitud(ActionEvent e)
Descripción	Método encargado de almacenar la solicitud en la base de datos.
Nombre	vistaPrevia(ActionEvent e)
Descripción	Método encargado de mostrar la vista previa de la solicitud.
Nombre	cancelar(ActionEvent e)
Descripción	Método encargado de cancelar la realización de la solicitud.
Nombre	isModificando()
Descripción	Método para saber si se está modificando o incluyendo la solicitud.
Nombre	setModificando(boolean modificando)
Descripción	Método encargado de notificar a la clase que la solicitud va a ser

	modificada.
Nombre	getSolicitudAsigAbogadoOficio()
Descripción	Método encargado de retornar la solicitud.
Nombre	setSolicitudAsigAbogadoOficio(SolicitudAsigAbogadoOficio solicitudAsigAbogadoOficio)
Descripción	Método encargado de establecer la solicitud a la clase.

2.4.1.3 Realizaciones de Casos de Uso

Los diagramas de contrato entre paquetes representan a nivel de paquetes las realizaciones de los casos de uso. A continuación se ilustran escenarios de casos de uso representativos del subsistema Debido Proceso.

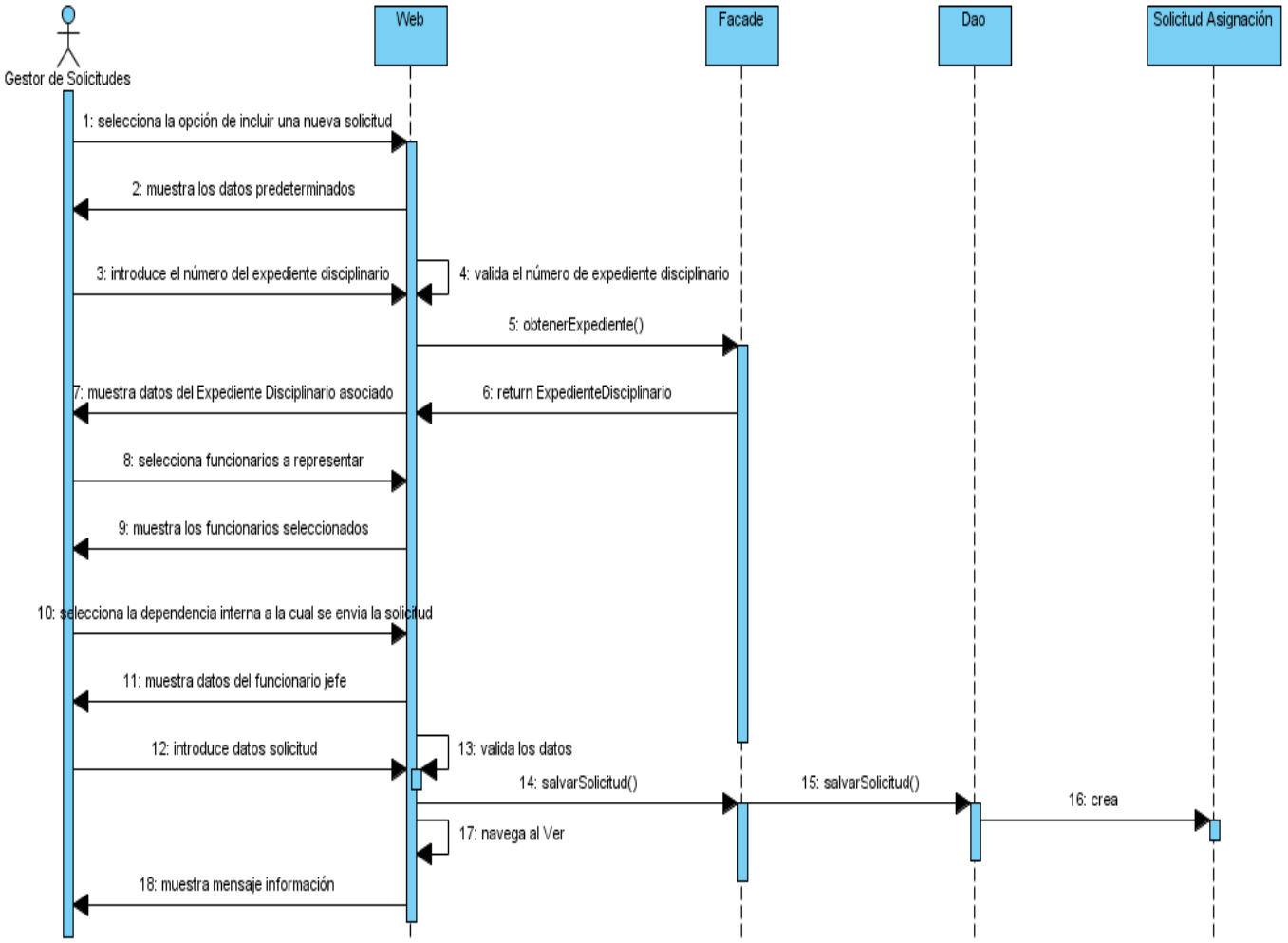

Fig 2.17 Diagrama de contrato entre paquetes CU Gestionar Solicitud Asignación Abogado Oficio, escenario Incluir.

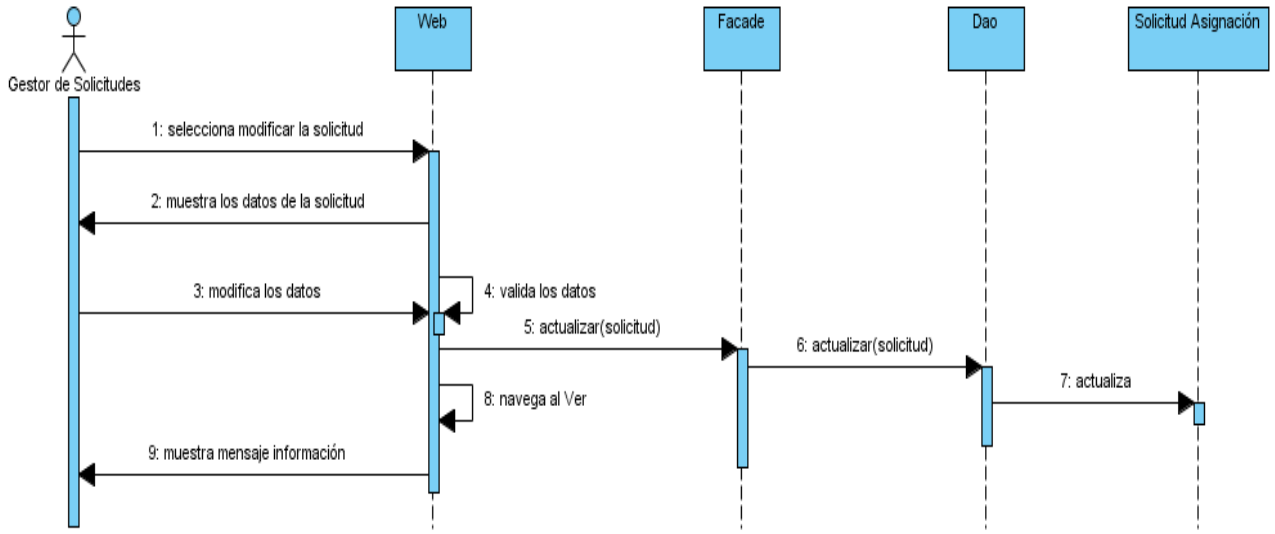

Fig 2.18 Diagrama de contrato entre paquetes CU Gestionar Solicitud Asignación Abogado Oficio, escenario Modificar.

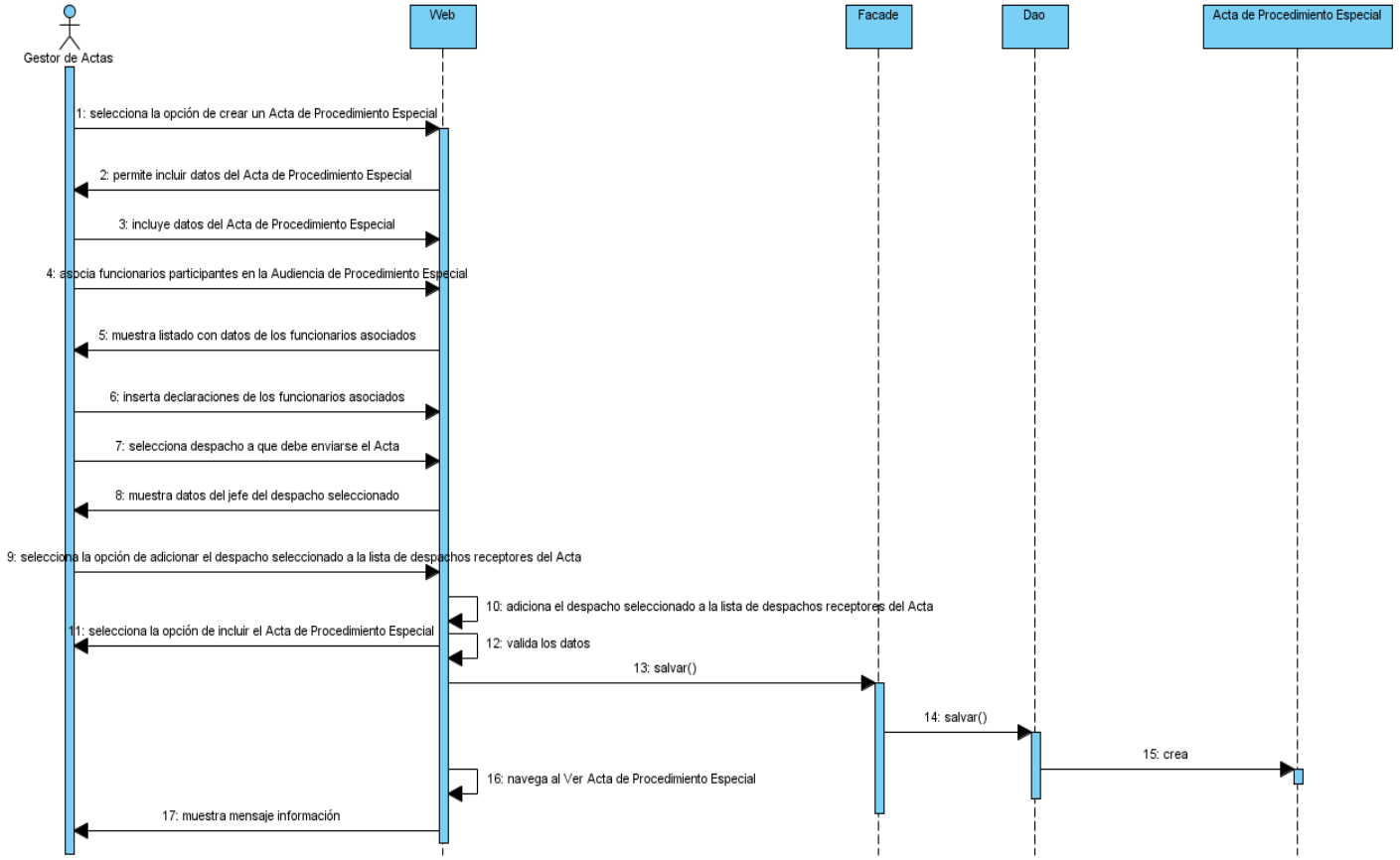

Fig 2.19 Diagrama de contrato entre paquetes CU Gestionar Acta de Procedimiento Especial, escenario Incluir.

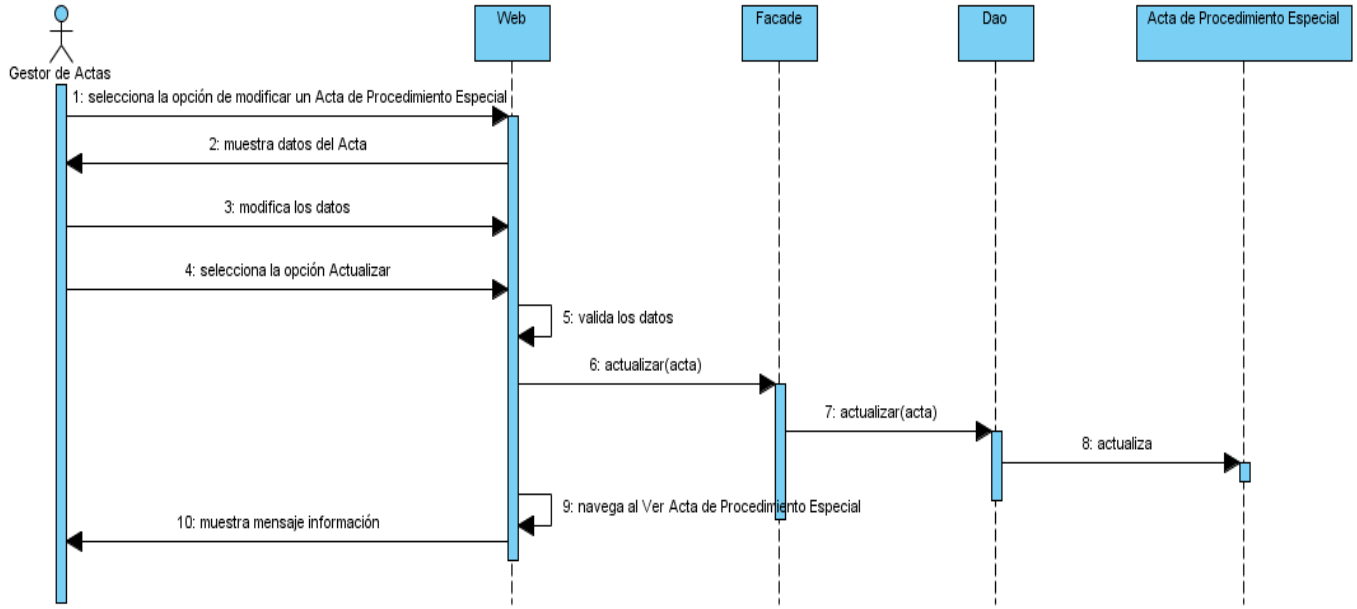

Fig 2.20 Diagrama de contrato entre paquetes CU Gestionar Acta de Procedimiento Especial, escenario Modificar.

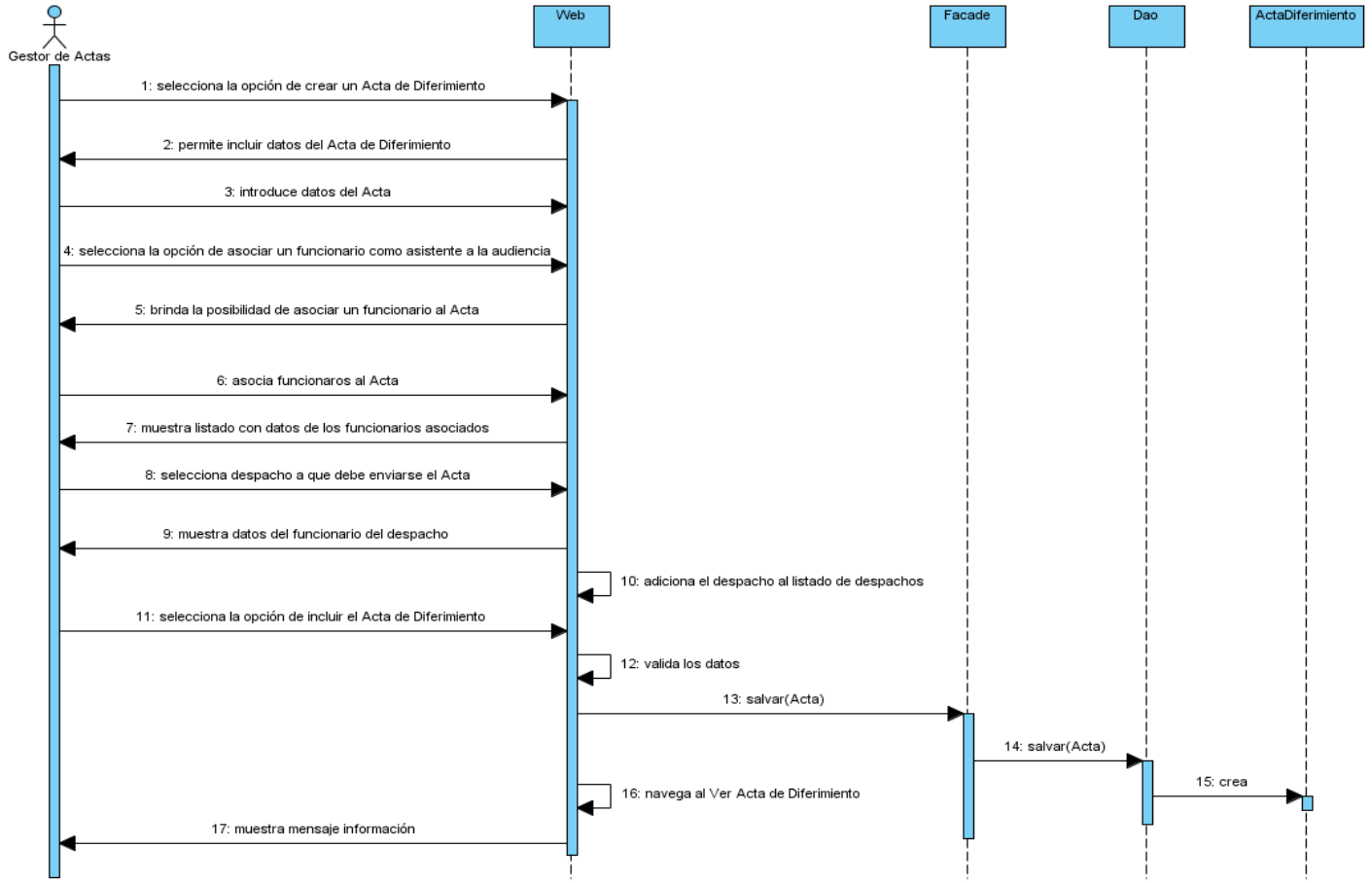

Fig 2.21 Diagrama de contrato entre paquetes CU Gestionar Acta de Diferimiento, escenario Incluir.

2.5 Modelo de Datos

El Modelo de Datos describe las representaciones lógica y física de los datos persistentes utilizados por la aplicación. (Rational Corporation, 2003) A continuación se muestran el diagrama de clases persistentes y el diagrama entidad relación del subsistema Debido Proceso.

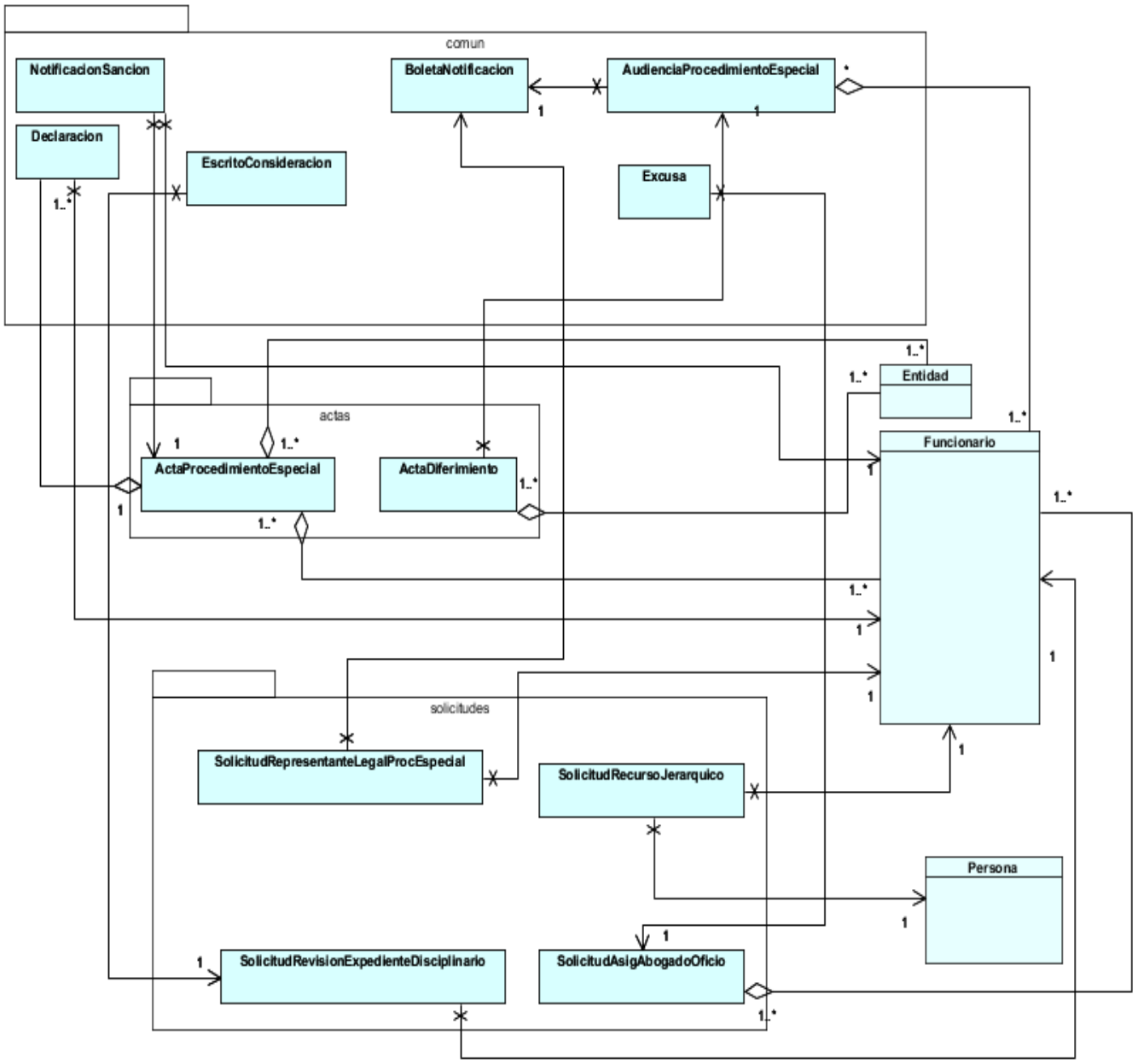

Fig 2.22 Diagrama de clases persistentes del subsistema Debido Proceso

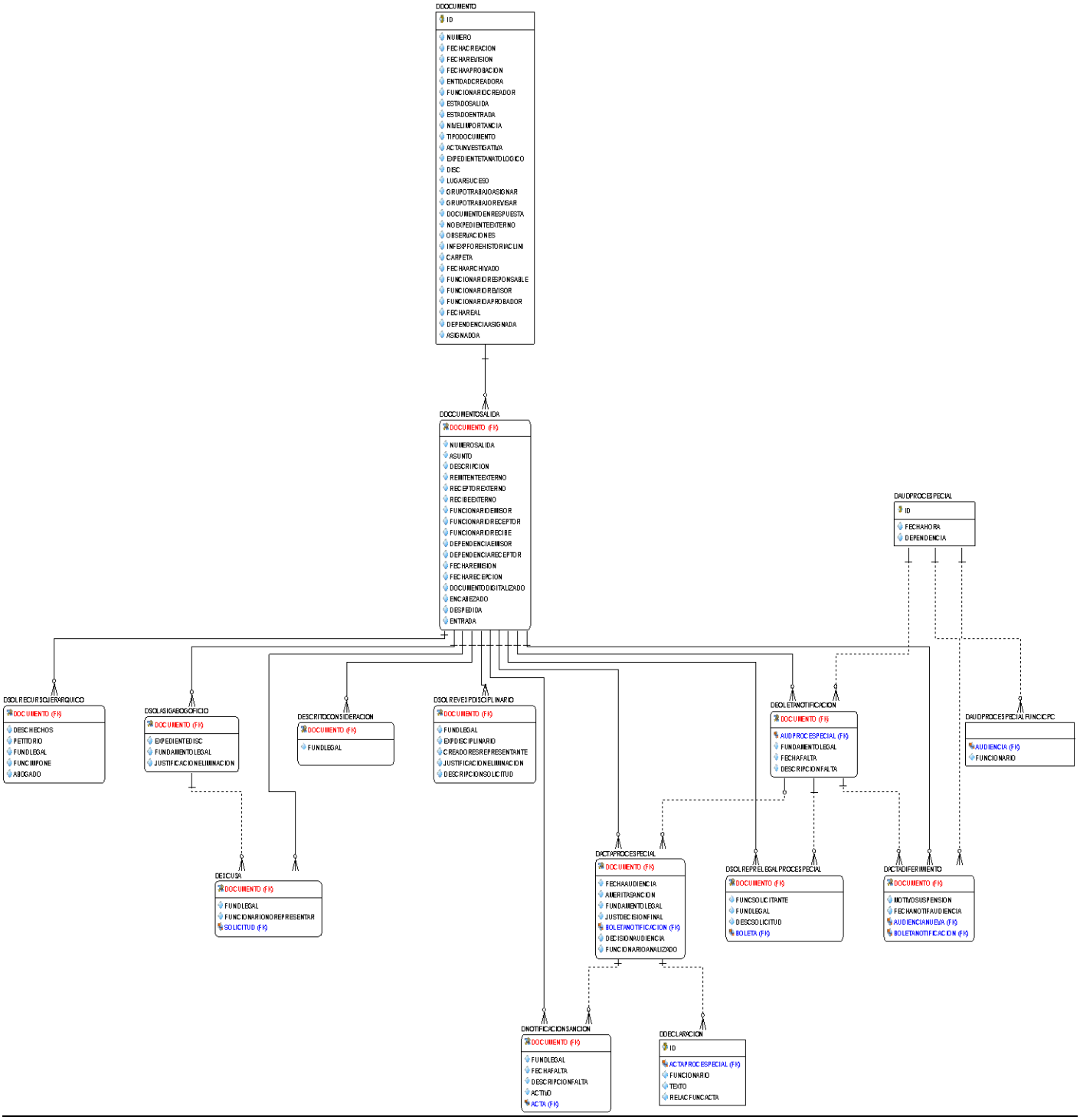

Fig 2.23 Diagrama entidad relación del subsistema Debido Proceso

2.6 Conclusiones

A lo largo del presente capítulo se expuso el análisis y el diseño del subsistema Debido Proceso. Se presentaron las principales clases del diseño, resúmenes de los casos de uso que guiarán la implementación, fueron abordados los patrones de diseño empleados y se generó el diagrama entidad relación con las tablas que permitirán persistir los datos generados por el subsistema. Los modelos anteriormente expuestos sirven como base del Modelo de Implementación.

CAPÍTULO 3. Implementación y pruebas de la propuesta de solución.

3.1 Introducción

A partir del Análisis y el Diseño de la propuesta de solución se pasa a la implementación del subsistema.

El Modelo de Implementación representa la composición física de la implementación en términos de Subsistemas de Implementación y Elementos de Implementación, los que comprenden carpetas, archivos, datos, código fuente y archivos ejecutables. (Rational Corporation, 2003) Fundamentalmente se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos.

En el presente capítulo se expone el modelo de implementación del subsistema Debido Proceso y luego se pasa a explicar en qué consisten las pruebas utilizadas para la validación de la solución y los resultados de las mismas. Esta parte reviste una gran importancia ya que son los resultados positivos de los distintos tipos de pruebas los que certifican que la solución debe satisfacer las necesidades del cliente y, los defectos encontrados, permiten refinar la implementación del Subsistema, propiciando esto una mayor calidad del mismo.

3.2 Diagrama de Subsistemas de Implementación

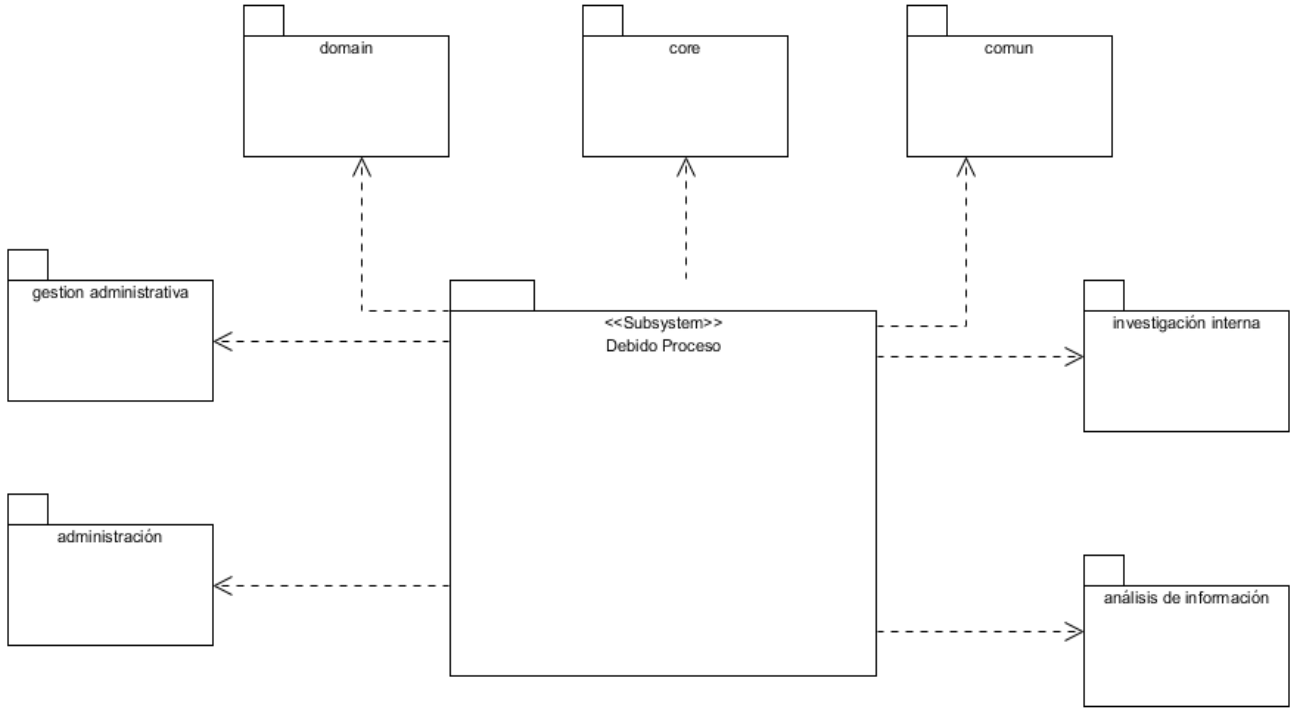

Fig 3.1 Diagrama de Subsistemas de Implementación

3.3 Diagramas de componentes.

Los Diagramas de Componentes ilustran las piezas de software que conforman un sistema. Poseen un nivel de abstracción mayor que los Diagramas de Clases porque normalmente un componente se compone de más de una clase.

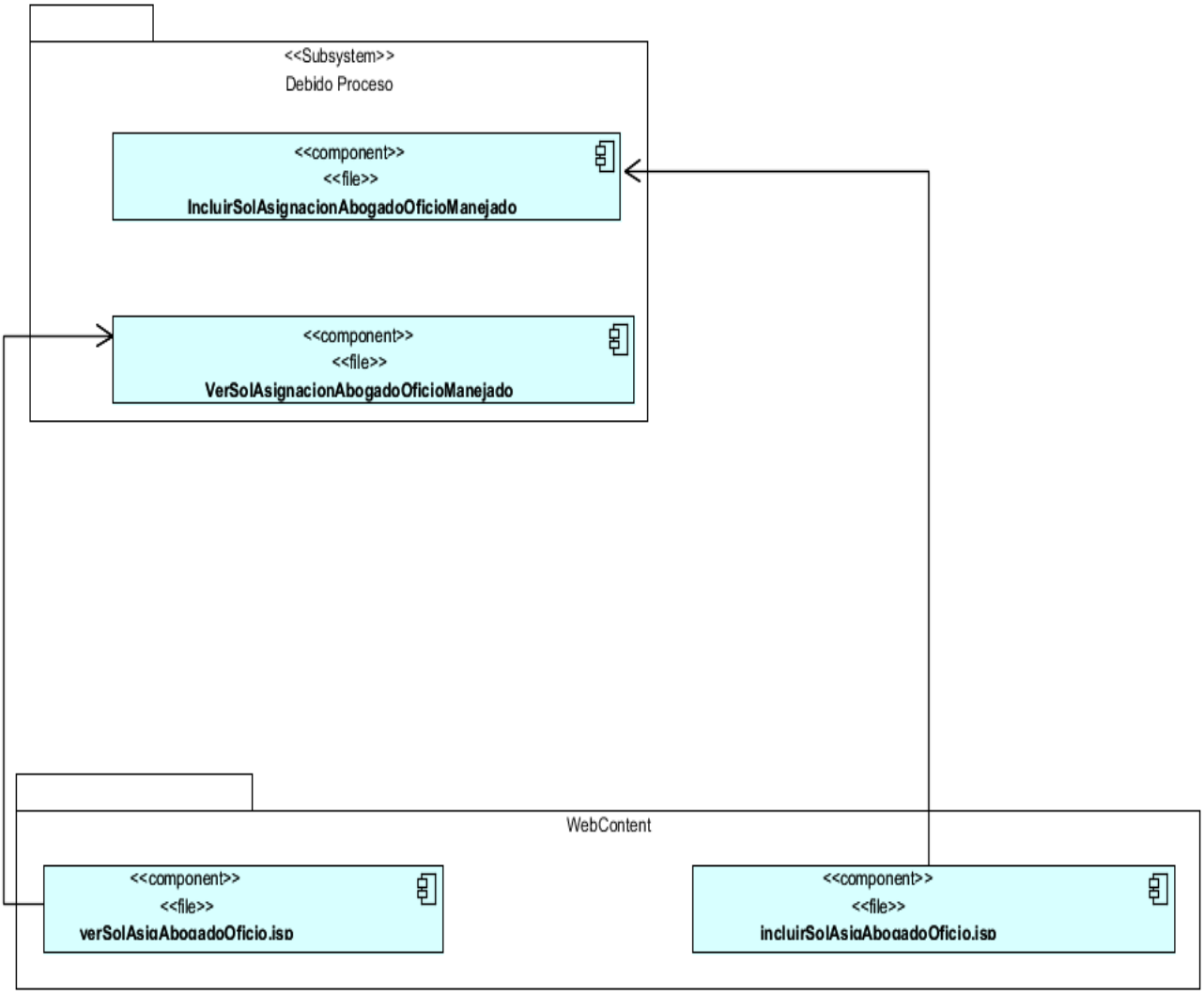

Fig 3.2 Diagrama de componentes del CU Gestionar Solicitud de Asignación de Abogado de Oficio, subsistema Debido Proceso, capa de Presentación.

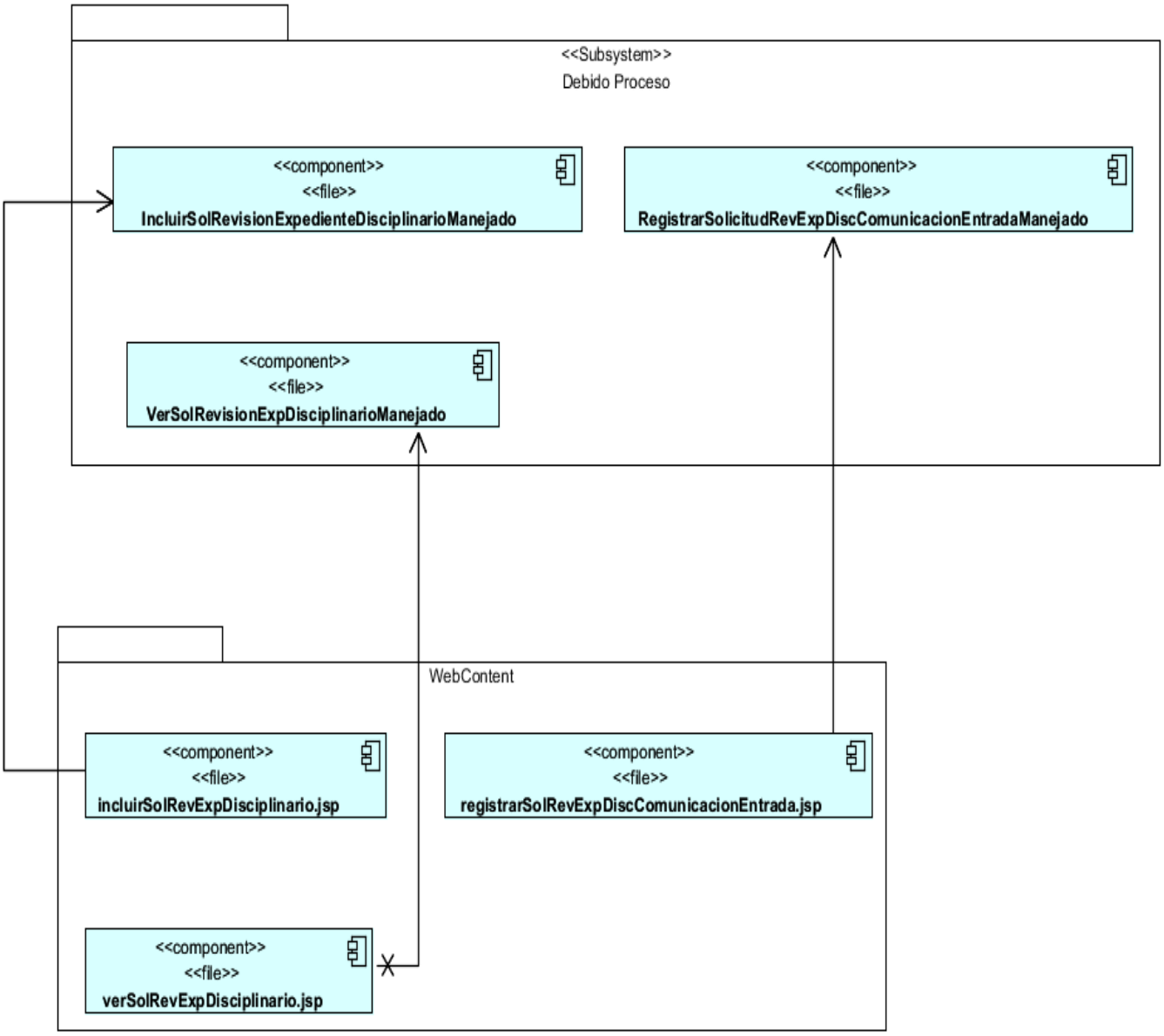

Fig 3.3 Diagrama de componentes del CU Gestionar Solicitud de Revisión de Expediente Disciplinario, subsistema Debido Proceso, capa de Presentación.

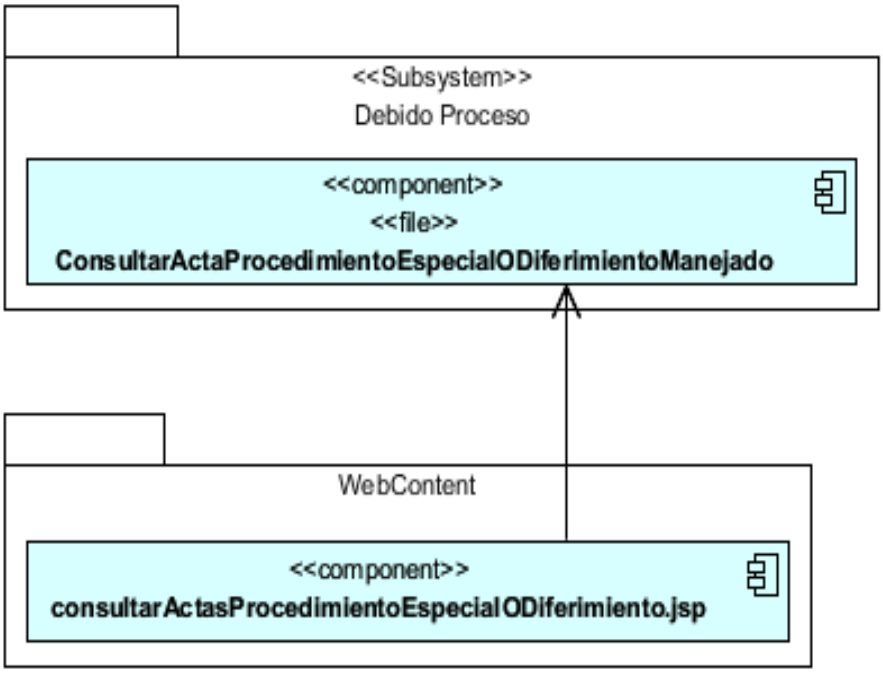

Fig 3.4 Diagrama de componentes del CU Consultar Actas de Procedimiento Especial o Diferimiento, subsistema Debido Proceso, capa de Presentación.

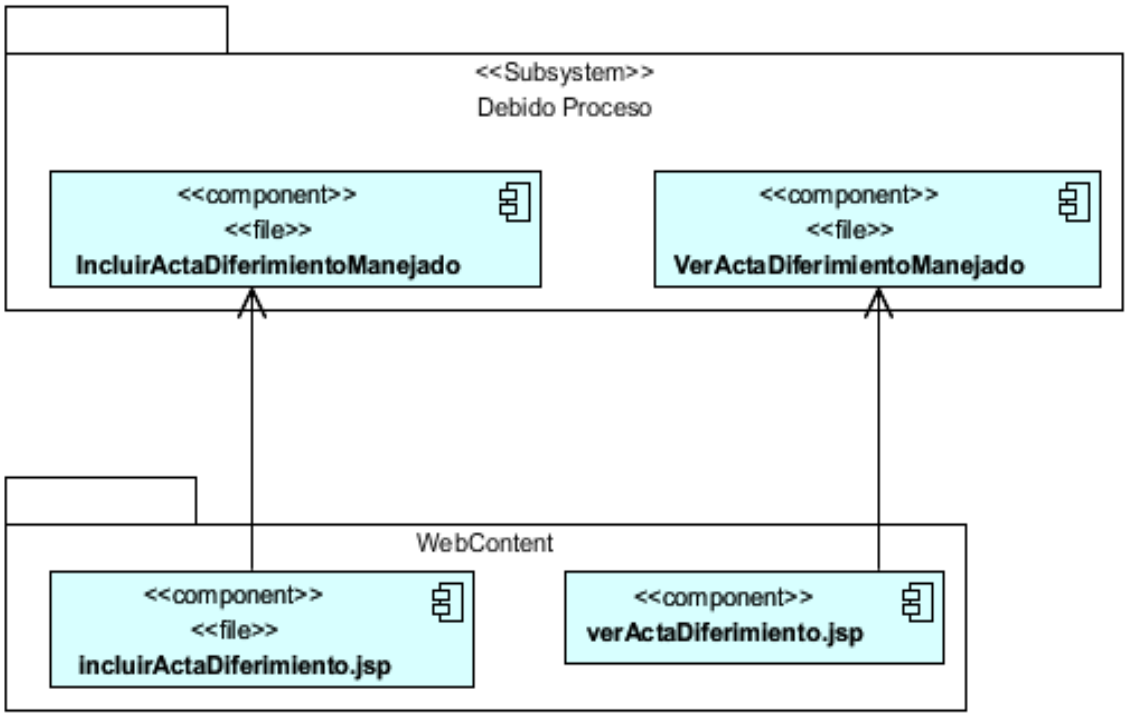

Fig 3.5 Diagrama de componentes del CU Gestionar Acta de Diferimiento, subsistema Debido Proceso, capa de Presentación.

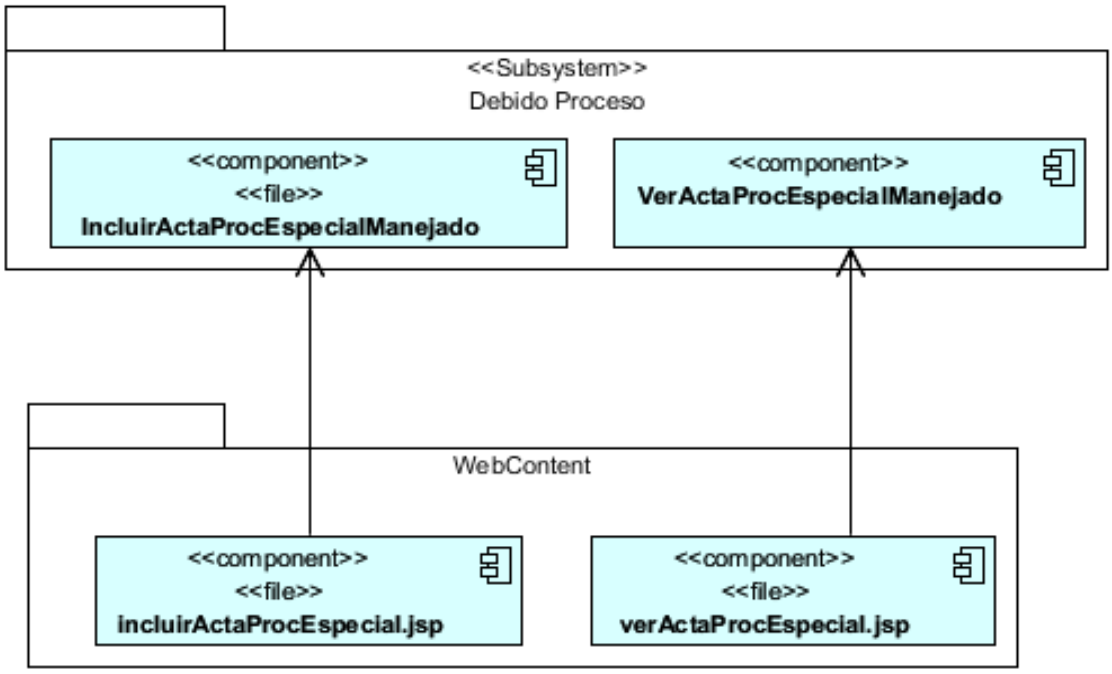

Fig 3.6 Diagrama de componentes del CU Gestionar Acta de Procedimiento Especial, subsistema Debido Proceso, capa de Presentación.

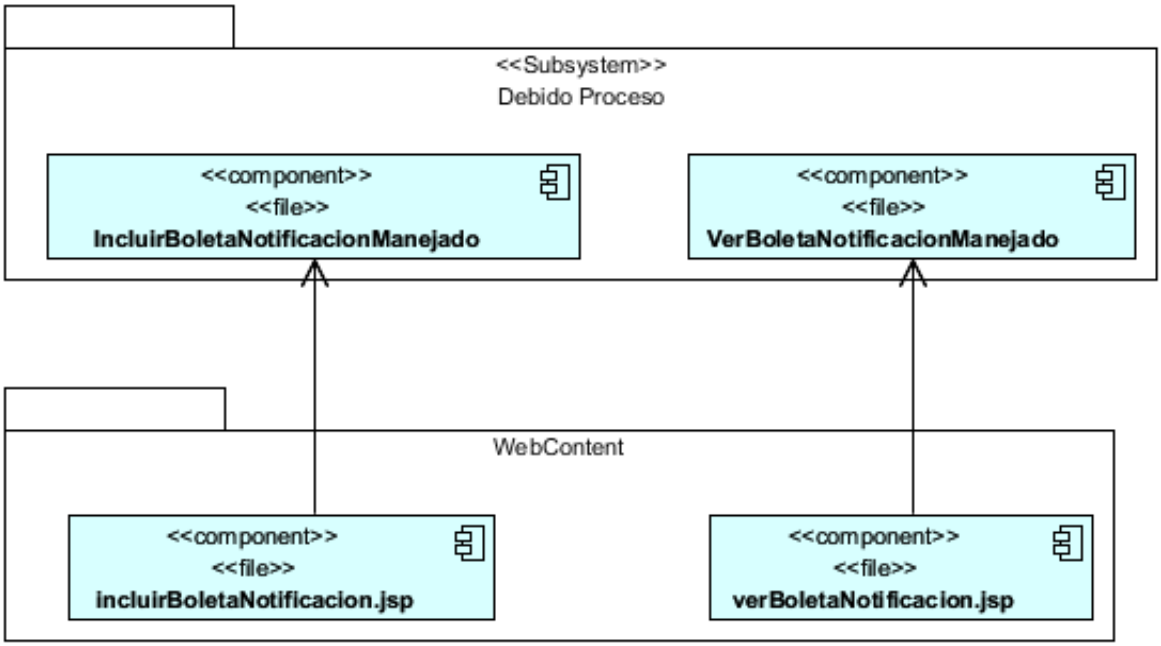

Fig 3.7 Diagrama de componentes del CU Gestionar Boleta de Notificación, subsistema Debido Proceso, capa de Presentación.

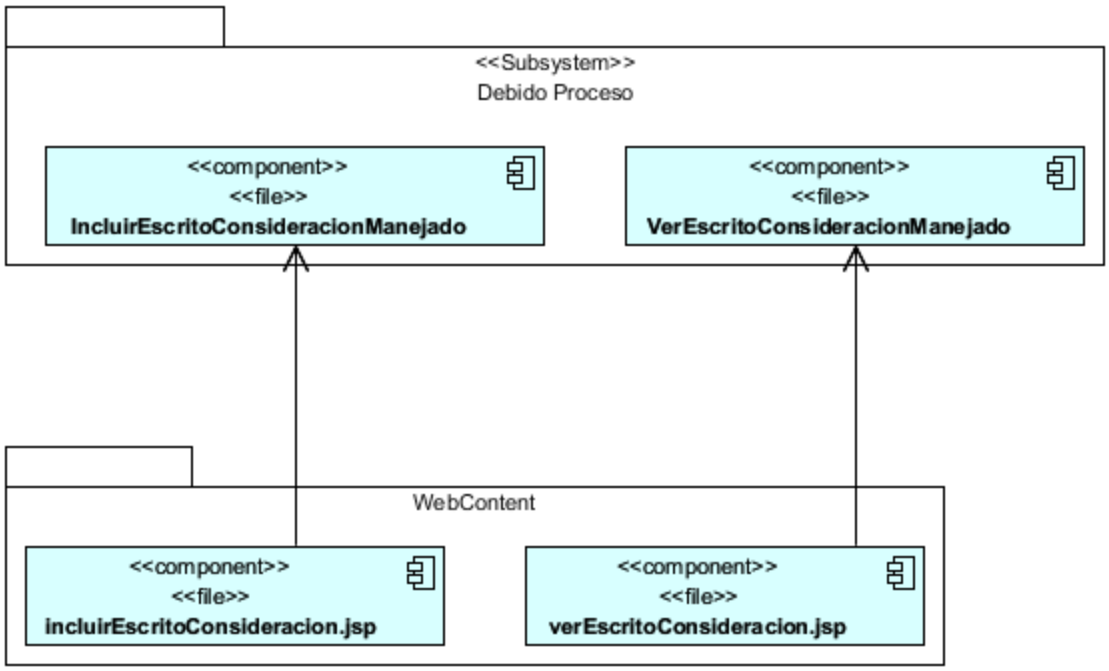

Fig 3.8 Diagrama de componentes del CU Gestionar Escrito de Consideración, subsistema Debido Proceso, capa de Presentación.

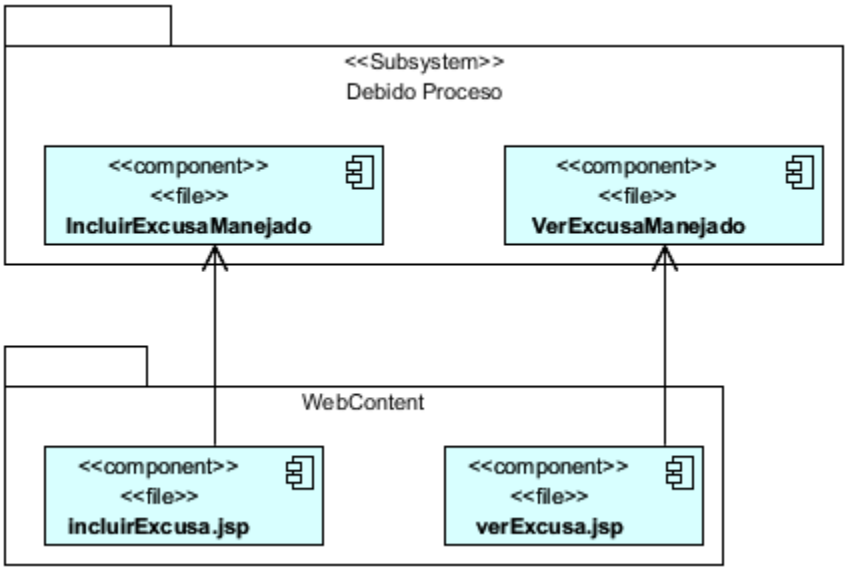

Fig 3.9 Diagrama de componentes del CU Gestionar Excusa, subsistema Debido Proceso, capa de Presentación.

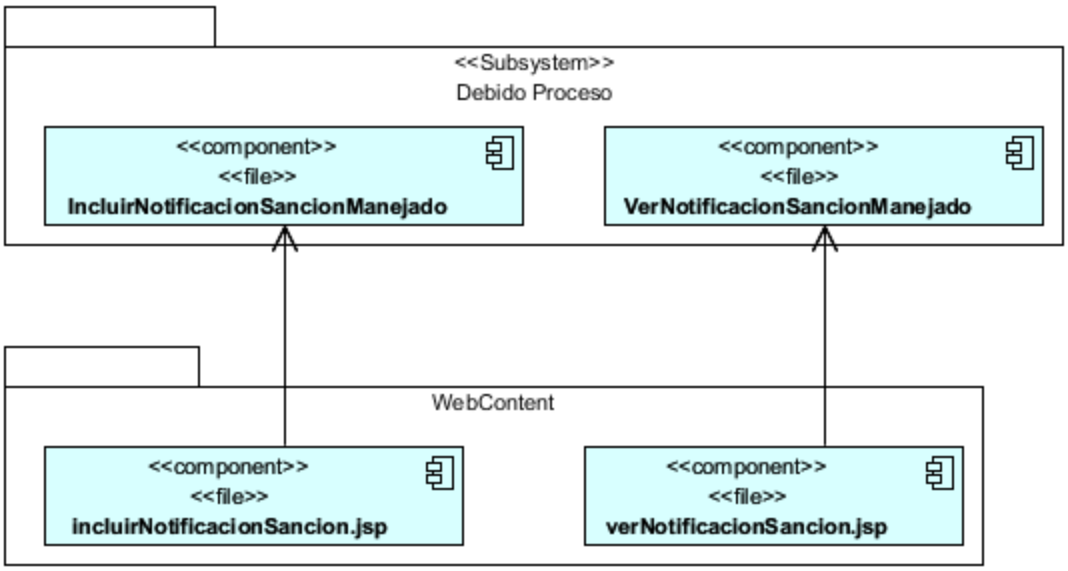

Fig 3.10 Diagrama de componentes del CU Gestionar Notificación de Sanción, subsistema Debido Proceso, capa de Presentación.

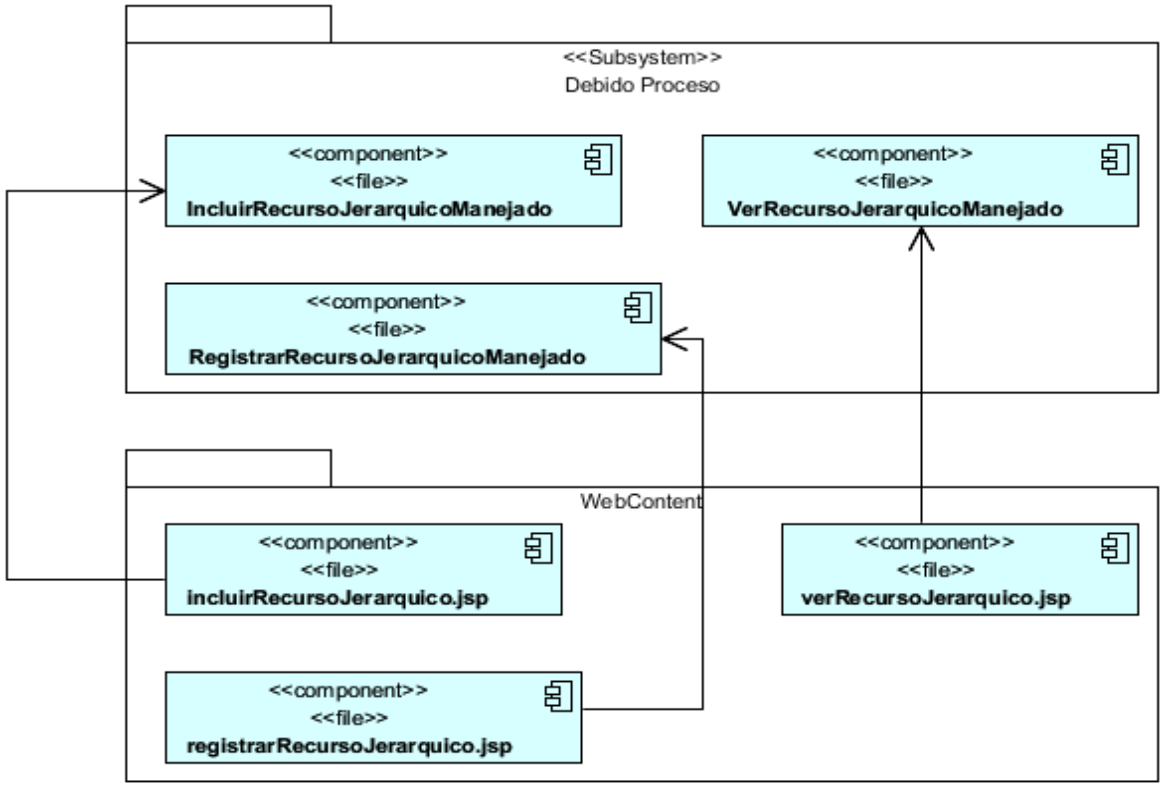

Fig 3.11 Diagrama de componentes del CU Gestionar Recurso Jerárquico, subsistema Debido Proceso, capa de Presentación.

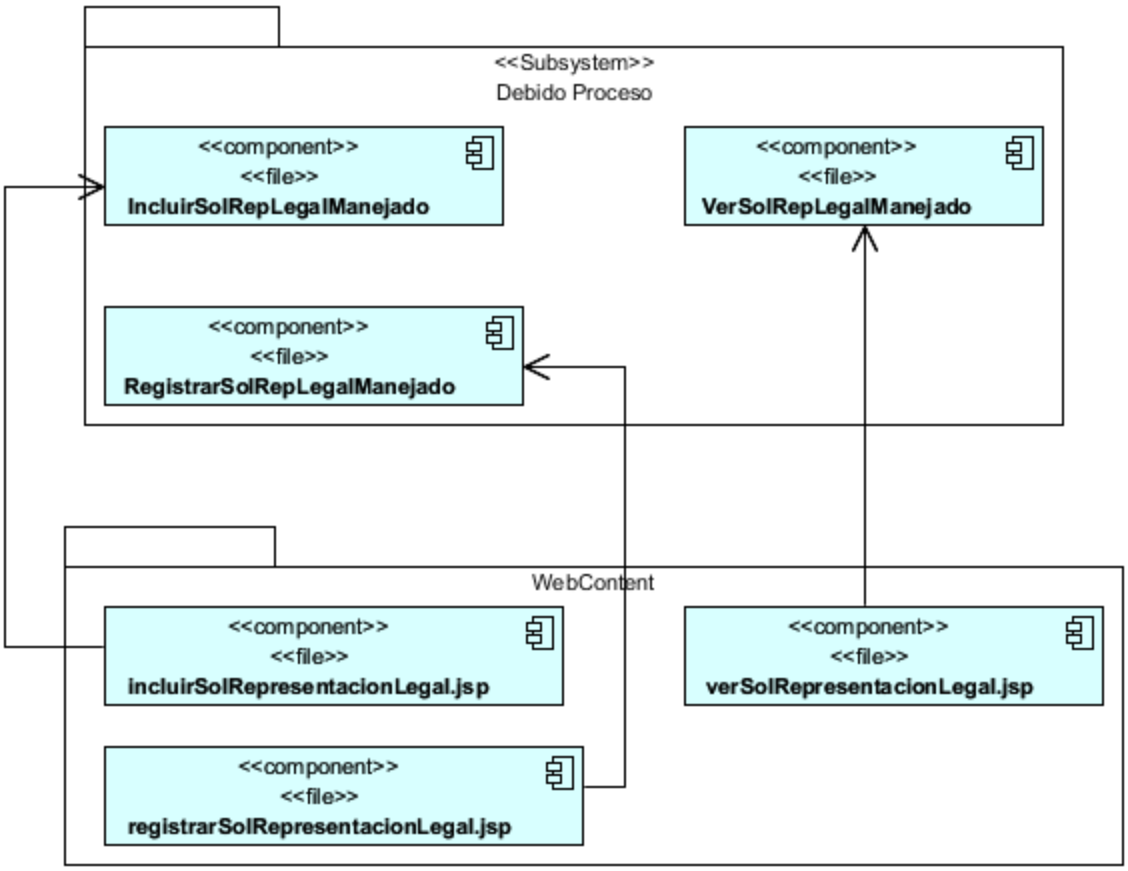

Fig 3.12 Diagrama de componentes del CU Gestionar Solicitud de Representación Legal en Procedimiento Especial, subsistema Debido Proceso, capa de Presentación.

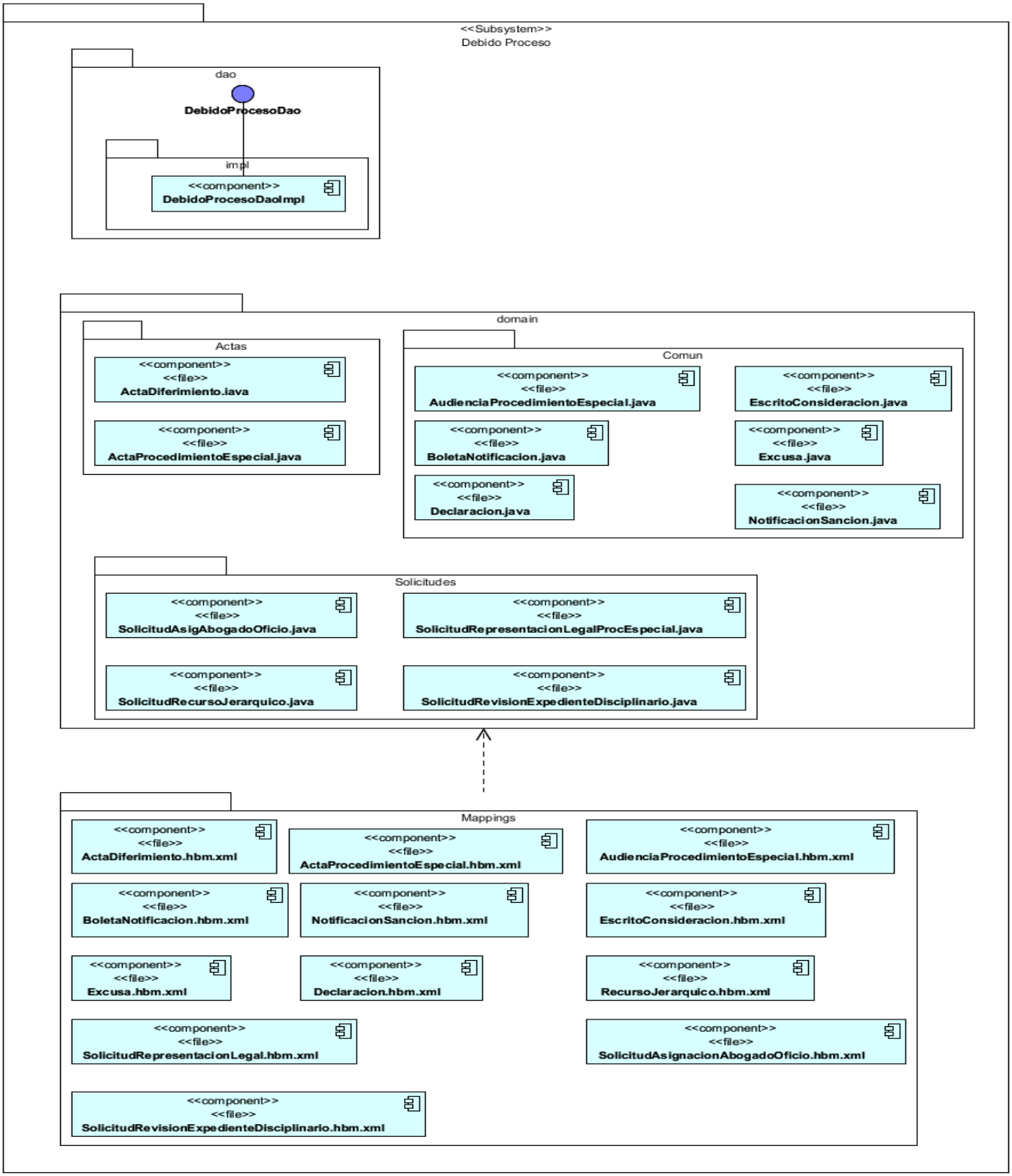

Fig 3.13 Diagrama de componentes del subsistema Debido Proceso, capa de Acceso a Datos.

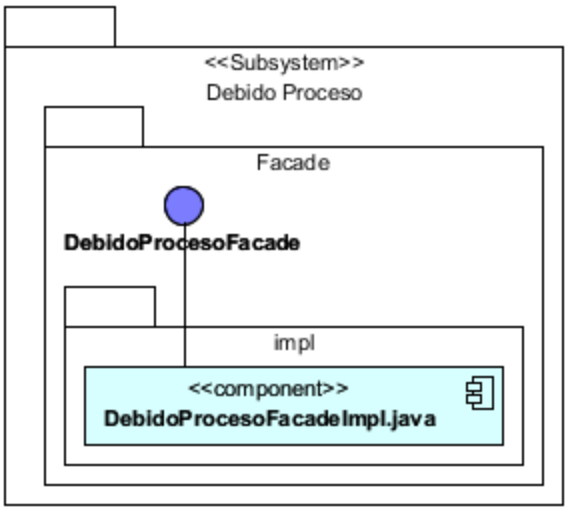

Fig 3.14 Diagrama de componentes del subsistema Debido Proceso, vista Lógica de Negocio.

3.4 Estándar de codificación.

Los estándares de codificación utilizados en la implementación del subsistema Debido Proceso, se encuentran definidos en un documento llamado Guía de Estilo de Código para el proyecto CICPC. El mismo se elaboró por la arquitectura del proyecto y está acorde a las Convenciones de Código Java, ya que no se debe desligar el estilo de código del lenguaje de programación.

3.5 Certificación de la propuesta de solución

A lo largo de la historia los proyectos de software han sufrido problemas de calidad, tanto en el proceso de desarrollo como en los productos en sí. La deficiente gestión de configuración de software produce demoras en los plazos de entrega de los productos y en el nivel en que los mismos satisfacen las expectativas del cliente.

La certificación del software es de vital importancia ya que proporciona un alto grado de confianza y seguridad en el producto, a la vez que permite hacer estimaciones realistas sobre el comportamiento del producto una vez que se encuentre en funcionamiento. Certificar el software es un requisito que debe ser cumplido con rigor.

Las pruebas son un conjunto de actividades en las cuales un sistema o componente es ejecutado bajo unas condiciones o requerimientos específicos, donde los resultados son observados y registrados para dar una evaluación de algún aspecto del sistema o componente que es evaluado y determinar la calidad del mismo.

Para lograr una correcta certificación del software se hace necesario una especificación documentada de los requisitos para que sea posible chequear su cumplimiento, a la vez que permita la elaboración de Casos de Prueba que conduzcan el desarrollo de las mismas.

Un Caso de Prueba es un conjunto de entradas de pruebas, condiciones de ejecuciones y resultados esperados, desarrollados para cumplir un objetivo en particular o una función esperada. No solo tienen que ser escritos para entradas válidas y esperadas, sino también condiciones no válidas e inesperadas.

Objetivos de las Pruebas:

- Encontrar y documentar los defectos que puedan afectar la calidad del software.
- Validar que el software trabaje como fue diseñado.
- Validar y probar los requisitos que debe cumplir el software.
- Validar que los requisitos fueron implementados correctamente.

La certificación de la implementación del subsistema Debido Proceso se rige por iteraciones de pruebas a diferentes niveles, de las cuales se expondrán los resultados obtenidos.

3.5.1 Tipos de Pruebas.

Para la certificación de productos de software existen diferentes tipos de pruebas. Las pruebas de Caja Blanca son aquellas que se realizan sobre las funciones internas de un determinado software o porción del mismo.

Mediante los métodos de prueba de la caja blanca, el ingeniero de software puede obtener casos de prueba que garanticen que:

- ✓ Se ejerciten por lo menos una vez todos los caminos independientes para cada módulo.
- ✓ Se ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsa.
- ✓ Se ejerciten las estructuras internas de datos para asegurar su validez.
- ✓ Se obtengan los resultados esperados luego de ejecutar una funcionalidad dada.

Se denomina caja negra a aquel elemento que es estudiado desde el punto de vista de las entradas que recibe y las salidas que devuelve, sin importar su funcionamiento interno. Existe un tipo de pruebas que se denominan de Caja Negra en las cuales se analizan porciones de software atendiendo a sus funcionalidades, es decir, a qué hace sin importar como lo realiza.

Por lo tanto, de una caja negra deben estar muy bien definidas sus entradas y salidas; pero no resulta necesario los detalles internos de su funcionamiento.

En las pruebas de Caja Negra los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene.

Los casos de uso brindan datos de entrada muy útiles en el diseño de pruebas de Caja Negra. Las pruebas de caja negra además:

- ✓ Verifican las especificaciones funcionales y no consideran la estructura interna del programa.
- ✓ Es hecha sin el conocimiento interno del producto.
- ✓ No validan funciones ocultas (por ejemplo funciones implementadas pero no descritas en las especificaciones funcionales del diseño) por tanto los errores asociados a ellas no serán encontrados.

Dadas las características de las pruebas antes expuestas para la certificación del subsistema Debido Proceso se utilizaron pruebas de Caja Blanca para la lógica interna y de Caja Negra para las funcionalidades específicas descritas en las especificaciones de casos de uso.

3.5.2 Niveles de Pruebas.

Pruebas Unitarias: Comienzan con la prueba de cada módulo. Una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código, lo cual sirve para asegurar que cada uno de estos funcione correctamente por separado. El objetivo de las pruebas unitarias es aislar cada parte del programa y mostrar que las partes individuales son correctas. Proporcionan un contrato escrito que el fragmento de código debe satisfacer. Estas pruebas aisladas proporcionan cinco ventajas básicas, fomentan el cambio, simplifica la integración, documenta el código, separa la interfaz del código y hace que los errores estén más acotados y sean fáciles de localizar. (Granados Hondares, y otros, 2009)

Pruebas de Integración: A partir del esquema del diseño, los módulos probados se vuelven a probar combinados para probar sus interfaces. Pruebas integrales o pruebas de integración son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias. Únicamente se refieren a la prueba o pruebas de todos los elementos unitarios que componen un proceso, hecha en conjunto, de una sola vez. Consiste en realizar

pruebas para verificar que un gran conjunto de partes de software funcionan juntos. (Granados Hondares, y otros, 2009)

Prueba del Sistema. El software ensamblado totalmente con cualquier componente hardware que requiera, se prueba para comprobar que se cumplen los requisitos funcionales. Cualquier pieza de software completo, desarrollado o adquirido, puede verse como un sistema que debe probarse, ya sea para decidir acerca de su aceptación, para analizar defectos globales o para estudiar aspectos específicos de su comportamiento, tales como seguridad o rendimiento. A este tipo de pruebas donde se estudia el producto completo se les llama Pruebas de Sistema. (Granados Hondares, y otros, 2009)

Pruebas de Aceptación. El cliente comprueba que el software funciona según sus expectativas. (Granados Hondares, y otros, 2009)

Para evaluar la solución desarrollada se planificaron varias iteraciones de pruebas, cuyos resultados se exponen a continuación.

Las pruebas se dividieron de la siguiente forma:

Primeramente se desarrollaron pruebas cruzadas por parte de los desarrolladores del subsistema, los cuales probaron los casos de uso implementados por sus compañeros detectando no conformidades (NC). Estas pruebas al ser realizadas por los desarrolladores en el rol de probadores, tienen un beneficio extra, y es que el conocimiento del funcionamiento interno del software que tienen estas personas les permite aportar valiosa información al desarrollo y detectar errores altamente complejos antes de poner el Subsistema en funcionamiento.

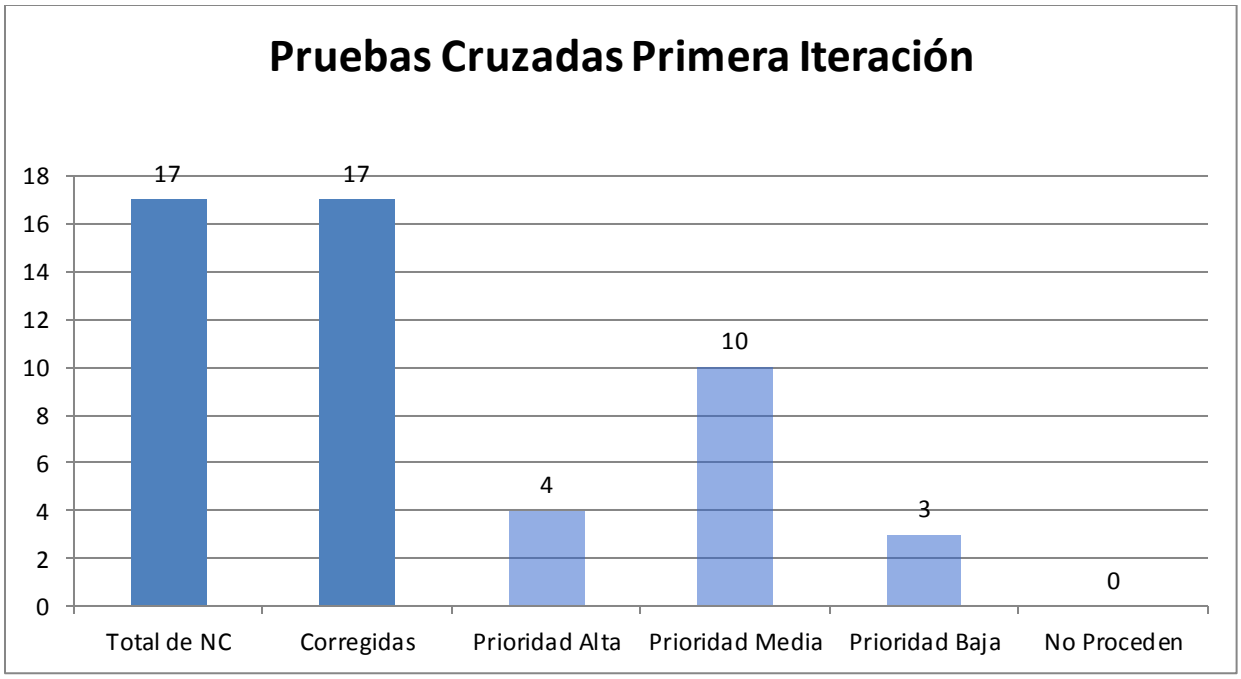

Fig 3.15 No conformidades detectadas en las pruebas cruzadas por el equipo de desarrollo del subsistema.

Luego de ser corregidas las NC detectadas en la primera iteración de pruebas cruzadas se procedió a realizar una segunda iteración pero, en este caso, fueron desarrolladores de otros subsistemas los que realizaron las pruebas del subsistema objeto de la presente investigación. Esta iteración arrojó los siguientes resultados.

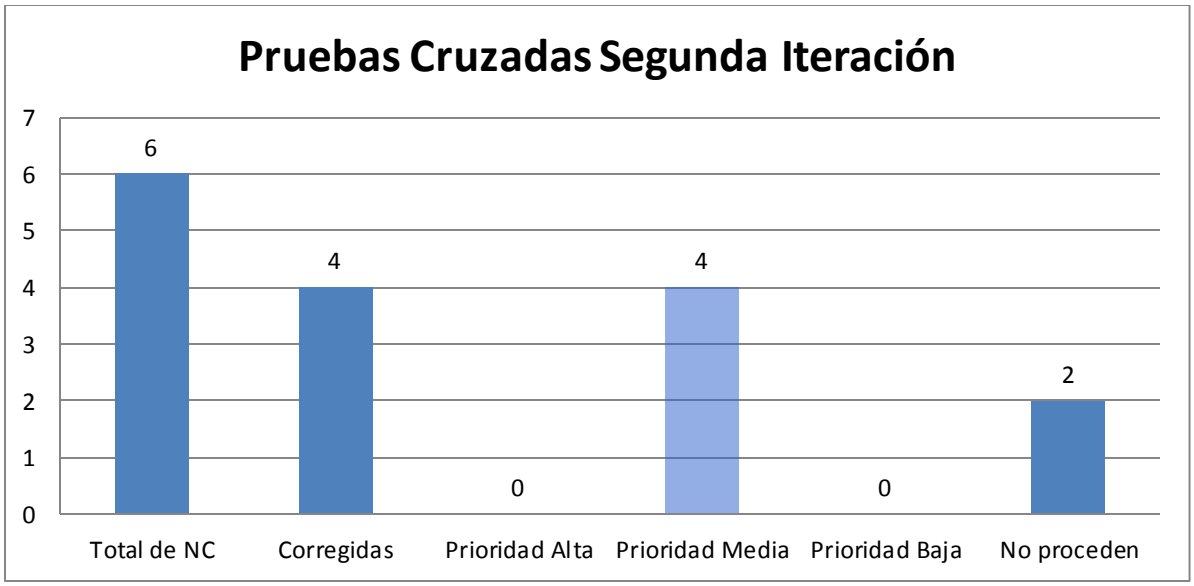

Fig 3.16 No conformidades detectadas en la segunda iteración de pruebas cruzadas por desarrolladores de otros subsistemas.

Posterior a esto se realizó una tercera iteración de pruebas cruzadas. En esta ocasión se probaron más a fondo los flujos completos del subsistema. Dichas pruebas arrojaron los siguientes resultados.

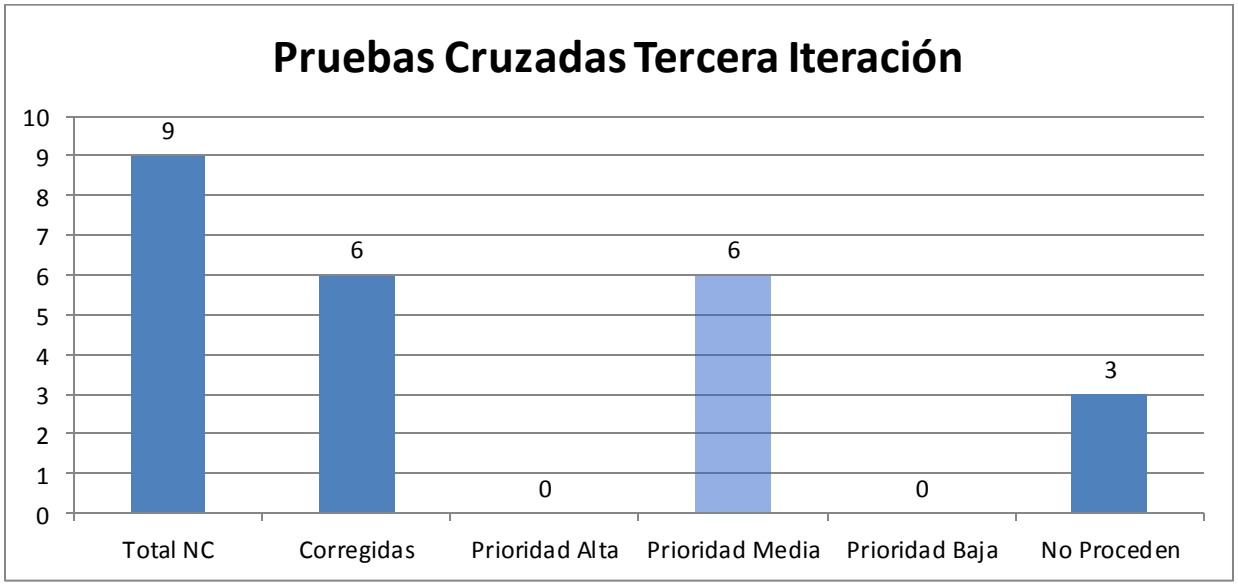

Fig 3.17 No conformidades detectadas en la tercera iteración de pruebas cruzadas por desarrolladores de otros subsistemas.

Al desarrollarse la cuarta iteración de pruebas cruzadas se pudo constatar una mayor estabilidad en el funcionamiento del subsistema, evidenciado esto por las siguientes estadísticas.

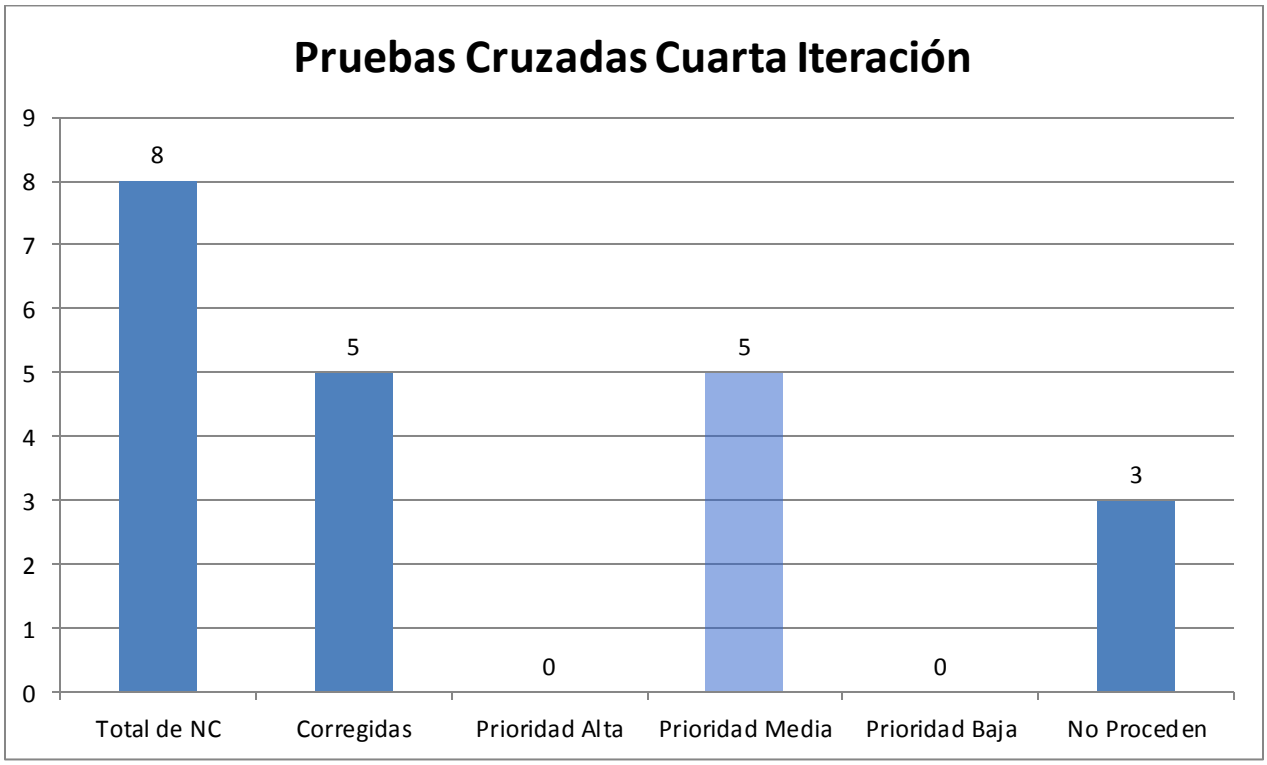

Fig 3.18 No conformidades detectadas en la cuarta iteración de pruebas cruzadas.

De las cinco no conformidades que fueron corregidas, sólo tres obedecieron a problemas de funcionalidad del subsistema, demostrando esto la alta estabilidad de la propuesta de solución.

3.6 Conclusiones

El desarrollo de un Modelo de Implementación perteneciente al subsistema Debido Proceso, permitió implementar la solución con un mayor nivel de seguridad al tener ya definidos los componentes que habrían de ser programados y cómo deberían integrarse los mismos. Además, el tener documentada la estructura de la solución de software permite que otras personas ajenas al código puedan darle soporte en el futuro con mayor facilidad.

La certificación de la solución es un requisito de obligatorio cumplimiento para garantizar la calidad del producto, por esto es la atención esmerada que se le presta a dicha actividad. Las pruebas se realizaron de forma incremental, permitiendo la detección de no conformidades y la corrección de las mismas por parte del equipo de desarrollo, obteniéndose así un producto de mayor calidad, lo cual solidifica las bases y brinda confianza y experiencia para las versiones posteriores.

Conclusiones

El estudio del estado en que se encuentra la Gestión Policial en el mundo contribuyó de manera importante en el desarrollo del subsistema Debido Proceso.

El análisis de los requisitos recopilados sobre las necesidades de automatización de determinados procesos pertenecientes a la Dirección de Inspectoría del Debido Proceso, posibilitó que el equipo de desarrollo tuviese un mejor enfoque a la hora de implementar el Subsistema en cuestión.

La utilización de la metodología seleccionada posibilitó documentar ampliamente la solución, lo que brindó guías en el proceso de desarrollo y agilizó la implementación del mismo.

Las herramientas y definiciones arquitectónicas utilizadas permitieron un desarrollo claro y fluido de la solución construido sobre bases sólidas y un entorno bien definido.

Los patrones de diseño utilizados posibilitaron el entendimiento entre los distintos miembros del equipo de desarrollo, a la vez que permitieron el desarrollo de componentes reutilizables en el desarrollo del sistema de software.

Las pruebas realizadas posibilitaron examinar cada parte de los componentes del subsistema, garantizando su correcto funcionamiento, así como la completa integración del mismo con el Sistema de Investigación e Información Policial.

Con el desarrollo de la solución de software se automatizaron un alto porcentaje de los procesos llevados a cabo por la Dirección de Inspectoría del Debido Proceso perteneciente al Cuerpo de Investigaciones Científicas, Penales y Criminalísticas de la República Bolivariana de Venezuela.

Bibliografía

Allamaraju, Subrahmanyam, y otros. *Programación Java Server con J2EE Edición 1.3.*

Apache Software Foundation. Apache MyFaces Project. [En línea] [Citado el: 14 de Enero de 2011.] http://myfaces.apache.org/.

Barreiro Noa, Dr. Alfredo. 2003. *LA CONTABILIDAD Y EL SISTEMA DE INFORMACIÓN EN LA EMPRESA CUBANA.* Las Tunas : s.n., 2003.

Berroa Álvarez, Geldri. 2009. *Análisis, Diseño e implementación del sub-módulo Dotación de Equipos Policiales perteneciente al módulo Registro y Control de SIIPOL.* 2009.

Booch, Grady, Rumbaugh, Jim y Jacobson, Ivar. 1999. *El Lenguaje Unificado de Modelado.* s.l. : Addison Wesley, 1999. ISBN: 84-7829-028-1.

Centro Nacional de Información de Ciencias Médicas. Biblioteca Virtual de Ciencia de la Información. *http://cis.sld.cu/indice.html.* [En línea] [Citado el: 14 de Enero de 2011.] http://cis.sld.cu/E/monografias/gestioncap1.html.

Collado Cabeza, Eduardo y Díaz Berenguer, Angi. 2003. Madritel. *http://web.madritel.es/personales3/edcollado/index.html.* [En línea] 2003. [Citado el: 19 de Enero de 2011.] http://web.madritel.es/personales3/edcollado/ingsw/tema2/2-6.htm.

Computer Language Company Inc. YourDictionary.com. [En línea] [Citado el: 18 de Enero de 2011.] http://computer.yourdictionary.com/sql.

Diaz Cabrera, Maylin y Lopez Espinosa, Kenny. 2009. *TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS.* La Habana : s.n., 2009.

Eclipse Software Foundation. The Eclipse Foundation open source community website. [En línea] [Citado el: 14 de Enero de 2011.] http://www.eclipse.org/org/.

Gallardo, David, Burnette, Ed y McGovern, Robert. 2003. *Eclipse in Action: A Guide for Web Developers.* 2003.

Granados Hondares, Yaimara y Maure Díaz, Yeleyne. 2009. *Análisis, Diseño e Implementación del módulo Experticias Criminalísticas v2.0 del Sistema de Investigación e Información Policial.* Ciudad de La Habana : s.n., 2009.

Guevara Turruelles, José Carlos. 2009. *Análisis Diseño e Implementación de la Capa de Presentación de los Sub-Módulos Persona y Vehículo pertenecientes al Subsistema de Análisis de Información del SIIPOL.* Ciudad de La Habana : s.n., 2009.

Guibert Estrada, Lisandra. 2008. *Ingeniería de Requerimientos Aplicada al Proceso de Registro y Control de Equipos .* 2008.

Hernández Vega, Luis Enrique y Rodríguez Font, Reinaldo Javier. 2009. *Análisis, diseño e implementación del submódulo Solicitudes perteneciente al módulo Investigación Criminalística del Sistema de Investigación e Información Policial.* Ciudad de La Habana : s.n., 2009.

Herrera Pereda, Reiniel. 2009. *EasyPol: Propuesta de Componentes Genéricos Reutilizables para Sistemas de Investigación Policial.* Ciudad de La Habana : s.n., 2009.

Hurtado Bustamante, Diana Paola, Gutiérrez Valencia, Diana Patricia y Suárez Valencia, Juan Pablo. Escuela de Ingeniería de Sistemas y Computación. Universidad del Valle. [En línea] [Citado el: 16 de Febrero de 2011.] http://eisc.univalle.edu.co/materias/Material...G01/presentacion.ppt.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de software.* 2000.

JBoss Organization. *Hibernate Reference Documentation version 3.2.0.ga.*

JSF 2.0 Expert Group. JavaServer Faces.org. [En línea] [Citado el: 14 de Enero de 2011.] http://www.javaserverfaces.org/.

Kuchana, Partha. 2004. *Software Architecture Design Patterns in Java.* 2004. 0-8493-2142-5.

Larman, Craig. 2002. *UML y Patrones.* 2002.

LASPROVINCIAS.ES. lasprovincias.es. [En línea] [Citado el: 08 de Junio de 2011.] http://www.lasprovincias.es/20100615/comunitatvalenciana/valencia/policia-local-presenta-sistema-201006151607.html.

Letelier, Patricio. 2003. *Proyecto Docente e Investigador.* s.l. : DSIC, 2003.

Mann, Kito D. 2005. *JavaServer Faces in Action.* Greenwich : Manning Publications Co., 2005. 1-93239-11-7.

Peralta. 2003. *Sistemas de Información.* 2003.

Pressman, Roger S. *Ingeniería del Software: un enfoque práctico.* s.l. : McGraw - Hill.

Quintana, Claudia Jiménez. 2002. *Indicadores de Alineamiento entre Procesos de Negocios y Sistemas.* Universidad de Concepción : s.n., 2002.

Quiroga. 2002. Gestión de información, gestión del conocimiento y gestión de la calidad en las organizaciones. [En línea] 2002. [Citado el: 18 de Enero de 2011.] http://www.bvs.sld.cu/revistas/aci/vol10_5_02/aci04502.htm.

Rational Corporation. 2003. Rational Unified Process Help. [En línea] 2003.

Segura. 2005. Crean sistemas de control de armas y municiones para policías. [En línea] 2005. [Citado el: 19 de Enero de 2011.] http://www.comunidadesegura.org/?q=es/node/24469.

SpringSource Organization. Springsource community. [En línea] [Citado el: 17 de Enero de 2011.] http://www.springsource.org/about.

Unión Europea. Europa. Síntesis de la legislación de la UE. [En línea] [Citado el: 8 de junio de 2011.] http://europa.eu/legislation_summaries/other/l33183_es.htm.

Universidad de Oviedo. 2003. Sitio Web de la E.U de Ingeniería Técnica Informática de Oviedo. *http://petra.euitio.uniovi.es/.* [En línea] 2003. [Citado el: 14 de enero de 2011.] http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%20de%20Desarrollo%20Integrado.pdf.

Visual Paradigm Organización. 2009. Sitio Web oficial Visual-Paradigm. *Sitio Web oficial Visual-Paradigm.* [En línea] 2009. [Citado el: 10 de Enero de 2011.] http://www.visual-paradigm.com/product/vpuml/.

Wayne Bartle, Philip Francis. 2009. Potenciación comunitaria. *www.scn.org.* [En línea] 2009. [Citado el: 18 de Enero de 2011.] http://www.scn.org/mpfc/modules/mon-miss.htm.

Wiley Publishing, Inc. [En línea] [Citado el: 18 de Enero de 2011.] http://computer.yourdictionary.com/xml.