UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 8

ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DEL MÓDULO ACCIONES ESPECIALES DEL SISTEMA DE INVESTIGACIÓN E INFORMACIÓN POLICIAL

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

AUTOR

MIGUEL GÓMEZ LEÓN

TUTOR

ING. ANGEL ALBERTO VAZQUEZ SÁNCHEZ

Ciudad de La Habana, 2010

"Año 52 de la Revolución"

Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber.

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaramos que somos los autores de este trabajo y autorizamos a la Facultad 8 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año 2010.

AUTOR
MIGUEL GÓMEZ LEÓN

TUTOR
ING. ANGEL ALBERTO VAZQUEZ SÁNCHEZ

AGRADECIMIENTOS

Especialmente a mis padres, por ser quienes me han guiado en la vida en todo momento. A mi hermana, por la confianza que ha puesto en mí, por servirme de ejemplo y apoyarme en cuantas decisiones he tomado. A mis demás familiares, que siempre me tienen presente.

A mi tutor, quien siempre tuvo plena disposición para ayudarme y apoyarme. A todos mis amigos, que me ofrecieron su amistad incondicional y estuvieron presentes para brindarme su ayuda…

DEDICATORIA

A mis padres y hermana, las personas más importantes en mi vida, quienes me han dado lo suficiente para alcanzar cada meta.

RESUMEN

El Cuerpo de Investigaciones Científicas, Penales y Criminalísticas de la República de Venezuela (CICPC) es una de las instituciones más importantes que lucha contra la criminalidad existente en ese país. Utilizan el Sistema Integrado de Información Policial para gestionar digitalmente la información que manejan. Sin embargo, este software se encuentra obsoleto, no satisface las necesidades actuales de dicha institución y no brinda soporte a la gestión de acciones especiales. Como parte del proyecto de modernización del CICPC surge el presente trabajo, el cual se enmarca en la investigación y desarrollo del Módulo Acciones Especiales del Sistema de Investigación e Información Policial (SIIPOL), nuevo sistema de gestión policial. El Módulo Acciones Especiales, se encarga de llevar el control sobre algunos servicios prestados por el CICPC, los cuales son ejecutados directamente por Brigadas de Acciones Especiales, estos son: custodia de personalidades y de prisioneros. Durante la realización de estos servicios pueden ocurrir diversas acciones las cuales se registran en el sistema y son denominadas “Acciones Especiales”. El objetivo del presente trabajo es analizar, diseñar e implementar un módulo que automatice los procesos de gestión de acciones especiales. En este trabajo se describen los Sistemas de Gestión de Información e Información Policial, además se recoge toda la documentación referente al proceso de análisis, diseño e implementación del Módulo Acciones Especiales. Con el desarrollo del Módulo Acciones Especiales en el nuevo sistema, se espera mejorar el proceso de gestión de información de acciones especiales en el CICPC.

ABSTRACT

The Body of Scientific, Penal and Criminal Investigations of the Republic of Venezuela (CICPC, as its acronym in Spanish) is one of the most important institutions that fight against crime in this country. They use the Integrated Police Information System to manage information digitally. However, this software is already obsolete, it does not satisfy current necessities of this institution and it does not have support to special action management. This work belongs to the project of CICPC modernization and it is focused on investigation and development of the Module Special Actions of Research and Police Information System (SIIPOL, as its acronym in Spanish), a new police management system. The Special Actions Module controls some services of CICPC, which are directly executed by Special Actions Brigades. These services include: personalities custody and prisoners custody. Actions occurred during these services are called Special Actions and they are recorded in the system. The goal of this work is analyze, design and implement a module that automates the special action management processes. In this work are described the Information Management and Police Information Systems, besides, it includes all documentation about the analysis, design and implementation process of the Special Actions Module. With the development of the Special Actions Module in the new system, the process of special action information management in CICPC will be improved.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La República Bolivariana de Venezuela no está exenta del aumento de la actividad criminal, por lo cual los cuerpos policiales venezolanos necesitan manejar un mayor número de información referente a los hechos delictivos.

El Cuerpo de Investigaciones Científicas, Penales y Criminalísticas de la República Bolivariana de Venezuela (CICPC), tiene como propósito detectar, procesar y esclarecer hechos delictivos desarrollando una correcta investigación del delito a través de su conocimiento científico y así colaborar en la lucha del Gobierno Venezolano contra la criminalidad, contribuyendo al mejoramiento y mantenimiento de la seguridad ciudadana.

El CICPC cuenta actualmente con un Sistema Integrado de Información Policial, la infraestructura que lo soporta y las técnicas con que fue desarrollado son actualmente obsoletas, por ejemplo se utilizó como lenguaje de programación Natural y como gestor de base de datos Adabas. Además, dicho sistema no es flexible, lo que dificulta realizar algún cambio o dar soporte al software. Por tales motivos la aplicación carece de muchas funcionalidades necesarias para facilitar el trabajo de su personal, como la gestión de solicitudes de custodias y por tanto, no brinda el mejor servicio.

Este organismo está constituido por una extensa estructura organizativa, formada por despachos y departamentos que interactúan continuamente entre sí. Los procesos que se llevan a cabo en esta institución de investigaciones van más allá de las investigaciones penales y criminalísticas, ya que formando parte de él, se encuentran las Brigadas de Acciones Especiales (BAE), las cuales se encargan de procesar varios servicios solicitados, como la custodia de personalidades y de prisioneros, durante los cuales pueden generarse una serie de acciones especiales. Todos los servicios realizados por estas brigadas son importantes debido a que sirven de apoyo en esclarecimiento de hechos delictivos. A pesar de su importancia no tienen soporte en el Sistema Integrado de Información Policial, es decir, el control de cada uno de los procesos desarrollados dentro de las Brigadas de Acciones Especiales se hace de forma manual, por lo cual se hacen extensas las actividades para dar respuesta a cada uno de los servicios prestados. Como consecuencia de lo planteado, se tiene que: las custodias no se atiendan en tiempo y que hechos de importancia ocurridos durante la custodia no sean registrados.

Como parte de la investigación se plantea el siguiente **problema a resolver**: ¿Cómo garantizar la gestión de información de las Brigadas de Acciones Especiales dentro del CICPC?

El **objeto de estudio** lo constituye el proceso de desarrollo de software para Sistemas de Gestión de Información y el **campo de acción** se enmarca en el proceso de desarrollo de software para Sistemas de Gestión de Información Policial orientado a los procesos de Acciones Especiales.

El **objetivo general** de dicha investigación es:

- ✓ Desarrollar un módulo que automatice los procesos de gestión de las Brigadas de Acciones Especiales del CICPC.

Para darle cumplimiento se desglosa en los siguientes **objetivos específicos**:

- ✓ Desarrollar el marco teórico orientado al desarrollo de Software de Gestión de Información, haciendo énfasis en los sistemas basados en la Gestión de Información Policial.
- ✓ Realizar el análisis y diseño del Módulo Acciones Especiales del CICPC.
- ✓ Implementar el Módulo Acciones Especiales, basándose en los elementos básicos de la Gestión de Acciones Especiales del CICPC.
- ✓ Validar el correcto funcionamiento del Módulo de Acciones Especiales.

La **idea a defender** que se plantea es: el análisis, diseño e implementación de un subsistema que automatice los procesos de Gestión de Acciones Especiales del CICPC, garantizará la gestión de las distintas solicitudes atendidas por las Brigadas de Acciones Especiales en el CICPC.

Las tareas de investigación a realizar para dar cumplimiento a los objetivos específicos son las siguientes:

1. Estudio de los conceptos sobre Sistemas de Gestión de Información y sistemas basados en la Gestión Policial.
2. Descripción del lenguaje, metodología de desarrollo y las herramientas establecidas para la solución.

3. Descripción de los procesos de Gestión de Acciones Especiales definidos para el CICPC.
4. Descripción de la propuesta de arquitectura de software para el nuevo sistema SIIPOL.
5. Análisis del modelo de casos de uso que da cumplimiento a los requisitos funcionales y no funcionales asociados al Módulo de Acciones Especiales.
6. Realización de los diagramas que conforman el modelo de análisis para los casos de uso del Módulo de Acciones Especiales.
7. Realización de los diagramas que conforman el modelo de diseño para los casos de uso del Módulo de Acciones Especiales.
8. Aplicación de los patrones para el refinamiento del modelo de diseño.
9. Implementación de la propuesta de solución.
10. Validación del correcto funcionamiento del Módulo de Acciones Especiales integrado a los demás subsistemas del SIIPOL.

El desarrollo de la siguiente investigación exige la utilización de métodos científicos, que permitan abordar la realidad y la naturaleza de los procesos desarrollados en las Brigadas de Acciones Especiales, con el fin de descubrir su esencia y relaciones. Para dar respuesta a los objetivos planteados se utilizará el método Modelación, fundamentalmente para visualizar los diferentes procesos, también para representar partes del módulo desde diferentes puntos de vista, como son el análisis, diseño, implementación y otros que son de vital importancia para el entendimiento de su funcionamiento. Además, se hace necesario el uso del Método Analítico Sintético, para realizar un análisis de la documentación referente a estas brigadas, identificar lo esencial dentro de cada proceso, determinar los rasgos que los identifican y diferencian de otros. También se analizará la documentación relacionada a los Sistemas de Gestión de Información y Gestión de Información Policial, logrando así un mejor entendimiento de estos tipos de sistemas.

La investigación quedará estructurada en 3 capítulos que agruparán los contenidos de la siguiente manera:

Capítulo 1. Fundamentación Teórica: se tratan elementos teóricos de la investigación tales como Gestión de Información, Gestión de Información Policial, metodología, lenguaje y herramientas de desarrollo que serán utilizadas para implementar la solución.

Capítulo 2. Análisis y Diseño de la solución propuesta: se analizará y diseñará la propuesta de solución, conformada por los requisitos funcionales y no funcionales del subsistema, incluyendo el diagrama y la descripción de casos de uso que se implementarán. Además, se confeccionará el modelo de análisis y posteriormente el modelo de diseño.

Capítulo 3. Implementación y validación de la solución propuesta: en este capítulo se elaborará el Modelo de Implementación, además se realizará la validación del módulo implementado.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1. INTRODUCCIÓN

Este capítulo tiene como objetivo fundamental ofrecer el marco teórico en el que se desarrolla el trabajo, además fundamenta la solución que se desarrolla en los capítulos próximos. Temas como la gestión de información y gestión de información policial serán tratados en este capítulo, debido a que están muy relacionados con el sistema al cual pertenece el módulo. De esta forma, se logrará un mejor entendimiento de los conceptos que existen sobre el tema en el cual se enmarca el problema a resolver. Además, se efectuará el estudio del ambiente de desarrollo, ofreciendo las principales características de la metodología, lenguaje y herramientas que se utilizarán en la implementación de la solución.

1.2. SISTEMAS DE GESTIÓN DE INFORMACIÓN

En la actualidad las organizaciones se desenvuelven en una sociedad cambiante y sostenida a la evolución, trayendo esto consigo que los paradigmas dominantes en años anteriores envejezcan y a la vez los métodos de análisis, unidos a las técnicas empleadas para la observación de la realidad. Lo anteriormente señalado en conjunto con el crecimiento continuo de la información, aumenta la importancia de saber qué hacer con la información tanto científica, tecnológica, como operativa y funcional de las organizaciones, surgiendo así la necesidad de que las organizaciones cuenten con novedosos Sistemas de Gestión de Información (SGI), los cuales le permitirán transformar datos en conocimiento de valor estratégico para apoyar sus operaciones organizacionales y orientar sus investigaciones. (Rivero Amador, y otros, 2007)

Analizando lo expresado anteriormente y teniendo en cuenta el desarrollo alcanzado en las tecnologías se hace evidente la necesidad de nuevas técnicas y herramientas que permitan un mejor análisis de la información, de ahí la importancia de los Sistemas de Gestión de Información ya que, “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).

Para poder utilizar la información en la toma de decisiones a diferentes niveles, es necesario gestionarla (recabar, registrar y analizar) de forma correcta y segura.

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 de 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.

Infomed expone en la Biblioteca Virtual de Ciencia de la Información como funciones de la Gestión de Información las siguientes (Infomed, 1998):

- Determinar necesidades internas de información, relativas a las funciones, actividades y procesos administrativos de la organización y a su satisfacción.
- Optimizar el flujo organizacional de la información y el nivel de la comunicación.
- Manejar eficientemente los recursos organizacionales de información, mejorar las inversiones sucesivas en los mismos y optimizar su aprovechamiento.
- Entrenar a los miembros de la organización en el manejo o la utilización de los recursos informacionales.
- Contribuir a modernizar u optimizar las actividades organizativas y los procesos administrativos relacionados con los mismos.
- Garantizar la calidad de los productos de la organización y asegurar su diseminación efectiva.
- Determinar las necesidades de información externa de la organización y satisfacerlas.

1.3. SISTEMAS DE GESTIÓN POLICIAL

El software de gestión policial tiene como objetivo automatizar los procesos en entidades policiales, además de digitalizar y centralizar toda la información recabada durante un proceso policial. Cada día es más la información a procesar en las entidades policiales, debido a que los índices de criminalidad van en aumento y es necesario agilizar todos los procesos de investigación y esclarecimiento de hechos delictivos, para brindar información actualizada y de manera rápida. Toda esta información se hace difícil de manejar manualmente, debido a esto existe la tendencia al uso de sistemas que gestionen la información policial para lograr mejores servicios en la gestión de seguridad ciudadana y gestión policial. (Díaz Cabrera, y otros, 2009)

Los Sistemas de Gestión Policial brindan un apoyo al desarrollo y esclarecimiento de las investigaciones policiales. El objetivo fundamental de dichos sistemas es: 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 del 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. (Díaz Cabrera, y otros, 2009)

La creación del nuevo Sistema de Investigación e Información Policial (SIIPOL) como Sistema de Gestión Policial, ayudará a la lucha que actualmente desarrolla el gobierno de la República Bolivariana de Venezuela contra el crimen.

1.4. MÓDULO DE ACCIONES ESPECIALES

La información se ha convertido en un recurso fundamental para la realización de las actividades de una organización y para la toma de decisiones por parte de las autoridades competentes.

Formando parte del CICPC se encuentran las Brigadas de Acciones Especiales, las cuales se encargan de llevar el control sobre varios servicios solicitados al CICPC, como la custodia de personalidades y de prisioneros. En este módulo se atienden dichas solicitudes registrándolas y asignándolas, una vez registradas son atendidas mediante un proceso en el cual se pueden registrar acciones especiales generadas durante el proceso de custodia. Los objetivos fundamentales de estas brigadas son: proteger la integridad física de una persona durante su traslado y registrar cada una de las acciones generadas durante la custodia.

1.5. METODOLOGÍA, LENGUAJES Y HERRAMIENTAS DE DESARROLLO

Acciones Especiales por ser un módulo del sistema SIIPOL, está orientado por decisiones tecnológicas ya definidas por la dirección del proyecto, por eso, no forma parte del objetivo de este trabajo, investigar acerca de las diferentes metodologías, lenguajes y herramientas de desarrollo, sino centrarse en las definidas.

1.5.1. METODOLOGÍA

Durante 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, 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 es aplicando una metodología de desarrollo de software, que son 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).

Para el desarrollo del sistema SIIPOL como metodología se eligió el Proceso Unificado de Desarrollo (RUP), creado por la empresa Rational, que intenta asegurar la producción de software de alta calidad que satisfaga las necesidades de los usuarios finales. RUP para conseguir lo que se propone se basa fundamentalmente en el orden y la documentación. (Jacobson, y otros, 2000)

Se consideró como la metodología correcta RUP por la distancia entre el equipo de desarrollo y el cliente, la magnitud de la propuesta de solución, la gran cantidad de personal con que cuenta el proyecto y el traspaso del conocimiento al cliente. RUP es una de las metodologías robustas más generales, existentes en la actualidad, unifica de forma íntegra a todo un equipo de desarrollo de software y está organizado por 9 flujos de trabajo, 6 principales y 3 de apoyo, a su vez está dividido en 4 fases (Figueroa, y otros, 2007).

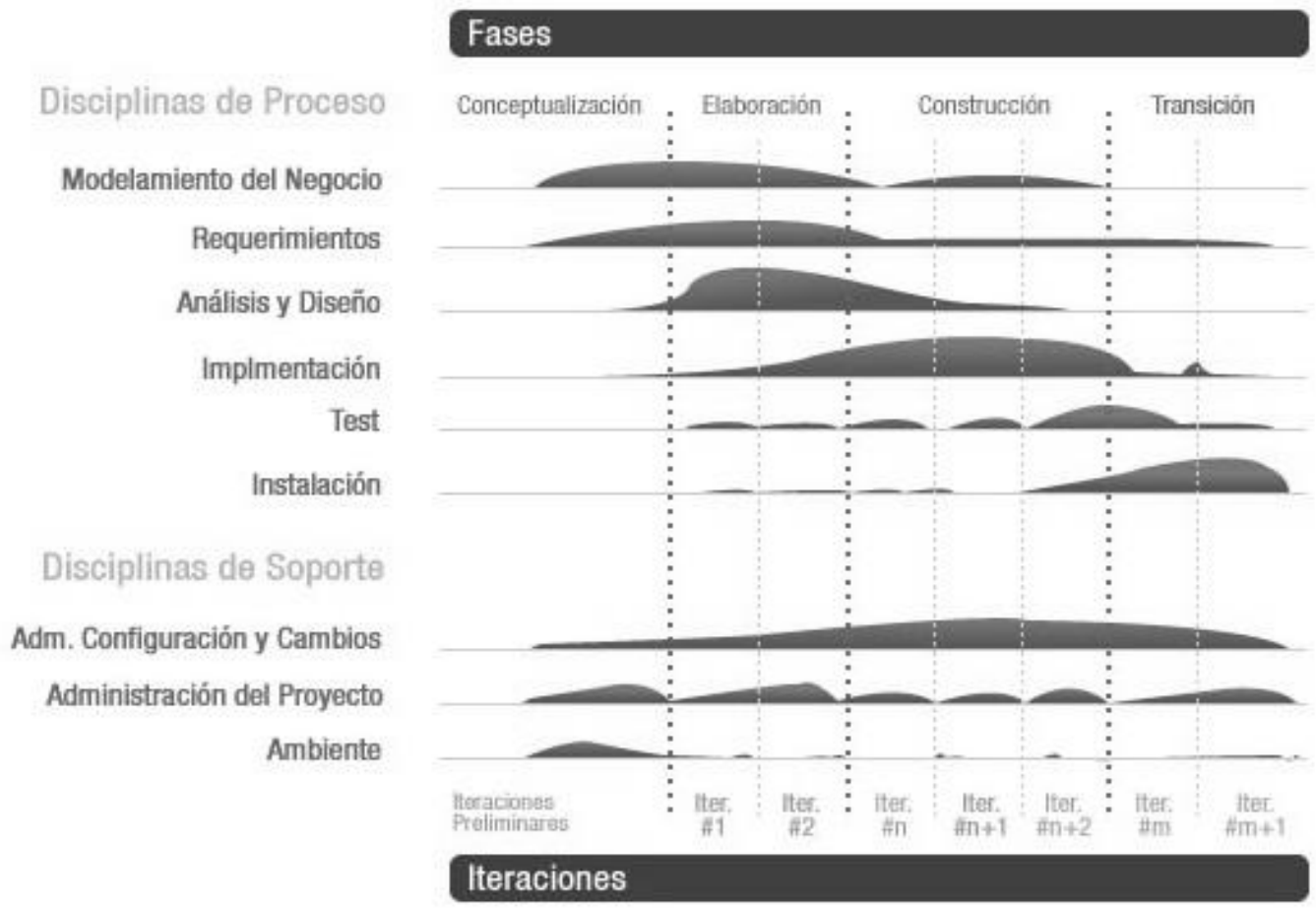


Figura 1. RUP

Fases del proceso (Jacobson, y otros, 2000)**:**

- Inicio o Conceptualización: se describe el negocio y se delimita el proyecto, describiendo sus alcances con la identificación de los casos de uso del sistema.
- Elaboración: se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales) identificados de acuerdo con el alcance definido.
- Construcción: se logra un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene una o varias versiones del producto que han pasado las pruebas. Se ponen estos entregables a consideración de un subconjunto de usuarios.
- Transición: la versión ya está lista para su instalación en las condiciones reales. Puede implicar reparación de errores.

Flujos de trabajo (Jacobson, y otros, 2000)**:**

- Modelación del negocio: describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- Requerimientos: define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- Análisis y diseño: describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
- Implementación: define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- Prueba: busca los defectos a lo largo del ciclo de vida.
- Despliegue o Instalación: produce un entregable del producto y realiza actividades como empaque, instalación, asistencia a usuarios, etc. para entregar el software a los usuarios finales.

- Administración de configuración y cambios: describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a utilización/actualización concurrente de elementos, control de versiones, etc.
- Administración del proyecto: involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- Ambiente: contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

Los elementos fundamentales del RUP son (Jacobson, y otros, 2000)**:**

- Actividades: son los procesos que se llegan a realizar en cada iteración.
- Trabajadores: son las personas o entidades involucradas en cada proceso.
- Artefactos: un artefacto puede ser un modelo o un elemento de modelo, un documento, en fin todo lo que puede ser generado en el proceso.

Sus características principales son (Jacobson, y otros, 2000)**:**

- Dirigido por Casos de Uso: los casos de uso definen lo que el usuario desea a partir de la captura de requisitos y la modelación del negocio por eso ellos guían el proceso de desarrollo, pues los modelos que se obtienen representan la realización de los mismos.
- Centrado en arquitectura: la arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, pues describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. La arquitectura se representa a través de las (4+1) vistas, en perspectivas del sistema que logran una abstracción en cada uno de los casos.

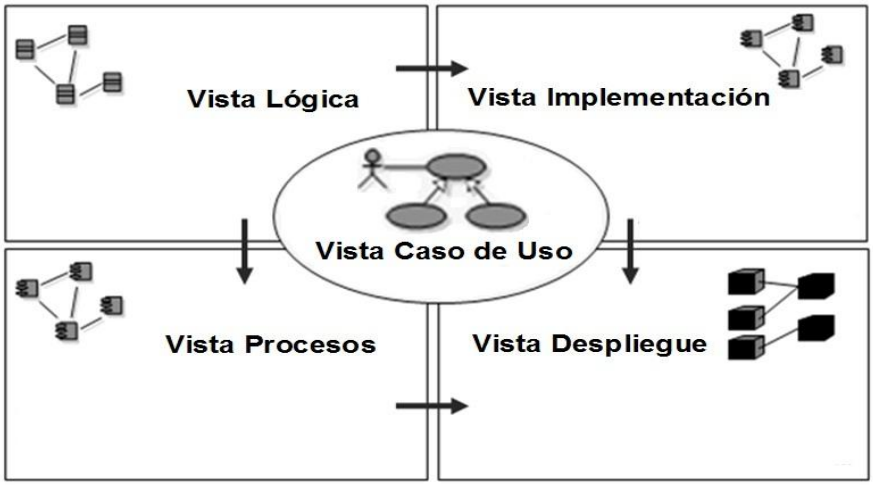


Figura 2. Vistas Arquitectónicas

- Iterativo e Incremental: propone que cada fase se desarrolle en iteraciones, de forma tal que se pueda dividir en pequeños proyectos mejorando su comprensión y desarrollo. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto.

1.5.2. LENGUAJE UNIFICADO DE MODELADO (UML)

En el proceso de desarrollo de software el modelado ejerce un papel fundamental, ya que puede comunicar cómo está estructurado el sistema desde todos los puntos de vista y detallar su comportamiento, así como también ayuda a representar aspectos conceptuales como procesos de negocios y funciones del sistema, aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. Es decir, el lenguaje de modelado es un lenguaje que se utiliza para definir un sistema de software, detallar los artefactos que se generan durante el desarrollo de software, documentar y construir, de esta forma, ofrece una amplia vista del sistema, desde la cual se logra un entendimiento entre todos los vinculados al desarrollo de software (Jacobson, y otros, 2007).

RUP utiliza como lenguaje de modelado el Lenguaje Unificado de Modelado (UML), el cual fue estandarizado por la Object Management Group u OGM (de sus siglas en inglés *Grupo de Gestión de*

Objetos) en 1997 para el desarrollo de software, pero también puede ser utilizado en otras metodologías. UML se ha convertido en el principal lenguaje de modelado, ya que fusiona notaciones de diversas técnicas para formar una herramienta compartida entre todos los ingenieros de software que trabajan en el desarrollo orientado 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 y 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.

En la actualidad UML se ha convertido en una práctica que da seguridad a la especificación de los procesos en el desarrollo del software y de gran utilidad para la comunicación y documentación.

1.5.3. HERRAMIENTAS DE MODELADO

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora) son utilizadas con el fin de automatizar los aspectos clave de todo el proceso de desarrollo de un sistema. CASE proporciona un conjunto de herramientas semiautomatizadas y automatizadas que están desarrollando la ingeniería de software en el mundo. 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, 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.
- Facilitar la reutilización de componentes software.
- Permitir un desarrollo y un refinamiento visual de las aplicaciones, mediante la utilización de gráficos.

De forma general la tecnología CASE automatiza el desarrollo del software, la documentación, la generación del código, el chequeo de errores y la gestión del proyecto, permitiendo además la reutilización del software y la estandarización de la documentación.

La herramienta CASE de modelado definida fue Visual Paradigm debido a que es una herramienta que soporta el ciclo de vida del desarrollo de software: el Análisis y Diseño Orientados a Objetos, Construcción, Pruebas y Despliegue. Visual Paradigm para UML soporta un conjunto de lenguajes, tanto para la generación de código como para la realización de ingeniería inversa en: Java, C++, CORBA IDL, PHP, XML Schema, Ada y Python. Además, apoya la generación de código C#, VB.NET, Object Definition Language (ODL), ActionScript, Delphi, Perl. La ingeniería inversa también soporta clases Java, .NET dll y .exe, JDBC, así también los archivos de mapeo Hibernate. Visual Paradigm facilita generar código Java a partir de modelos y también generar modelos a partir de código Java, y de esta forma cualquier cambio

que se realice en el código se refleja en el modelo y viceversa. 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)

1.5.4. LENGUAJE DE PROGRAMACIÓN

En la selección del lenguaje de programación para el desarrollo del SIIPOL se tuvo en cuenta varias características como la robustez, seguridad, independencia de la arquitectura, portabilidad y sencillez, con el objetivo de garantizar un sistema seguro.

El lenguaje programación escogido fue Java, el cual es desarrollado por Sun Microsystems Inc. Este lenguaje desde sus inicios fue concebido para la programación orientada a objetos, algunas de sus sintaxis son análogas a las de C y C++. Java a pesar de ser semejante a C y C++ presenta diferencias, uno de estos es que posee un modelo de objetos mucho más simple y elimina herramientas de bajo nivel, que provocan numerosos errores, como la manipulación de punteros o memoria. (Gutiérrez, 2009)

Como cualidad importante es la independencia de la plataforma, lo cual quiere decir que, programas escritos en el lenguaje Java pueden ser ejecutados en cualquier tipo de hardware o dispositivos siempre que se cuente con una máquina virtual, ejemplos de dispositivos son: computadoras, teléfonos móviles, tarjetas inteligentes, sintonizadores, impresoras, cámaras web y otros (MKM Publicaciones Informáticas, 2007). Debido a que la memoria es uno de los principales recursos que deben ser gestionados eficientemente, Java incorpora un recolector automático de memoria (garbage collector), este elimina una serie de problemas presentados en C y C++, permitiendo así que el programador no tenga la necesidad de administrar memoria dinámica de forma manual (Lopez, 2007).

Hoy en día la tecnología Java se ha convertido en la perfecta para cualquier tipo de aplicación, debido a la versatilidad y eficiencia que presenta, la portabilidad de su plataforma y la seguridad que aporta.

1.5.5. PLATAFORMA DE DESARROLLO

En el desarrollo de software cuando se desea seleccionar un lenguaje de programación se debe tener en cuenta elementos como son: el entorno de ejecución, el rendimiento, la escalabilidad, la portabilidad y la

seguridad, una vez analizados estos puntos de forma detallada se puede seleccionar el lenguaje que más se ajuste según lo que se desea.

La plataforma elegida fue Java 2 Enterprise Edition (J2EE) creada por Sun Microsystems, esta define un conjunto de estándares y especificaciones para el desarrollo de aplicaciones empresariales basado en la tecnología Java (Área de Diseño Gráfico y Multimedia - FIA, 2002). J2EE está capacitada para ejecutar aplicaciones creadas usando el Lenguaje de programación Java u otros lenguajes que compilen a bytecode[1]. La plataforma como tal es una máquina virtual encargada de la ejecución de aplicaciones, y un conjunto de librerías estándar que ofrecen funcionalidad común.

Esta plataforma basa su funcionamiento en un procesador virtual que ejecuta programas escritos en un lenguaje de programación (Java principalmente), dicho procesador es la máquina virtual de Java o Java Virtual Machine (JVM) el cual traduce el bytecode en instrucciones nativas de la plataforma destino; debido a esto se tiene como beneficio que una misma aplicación Java pueda ser ejecutada en una gran variedad de sistemas con arquitecturas distintas, siempre que se cuente con una implementación adecuada de la JVM (Área de Diseño Gráfico y Multimedia - FIA, 2002).

Algunas de las características que posee dicha plataforma son (Ciberaula, 2010):

- Portable: una vez desarrollada la aplicación, esta puede ejecutarse en cualquier plataforma para la que haya disponible una Máquina Virtual Java.
- Escalable: permite añadir nuevos componentes J2EE a una aplicación Web para soportar el aumento de clientes, sin tener que reescribir todo el código nuevamente.
- Altamente Soportada: prácticamente cualquier gran empresa de software tiene un contenedor de componentes (o servidor de aplicaciones) Web compatibles con J2EE, entre ellas IBM (Websphere), BEA (WebLogic), Apache (Tomcat), la propia Sun con su nuevo servidor de aplicaciones iPlanet, MacroMedia con (JRun), etc.
- Segura: el entorno de seguridad de la plataforma J2EE permite que se definan unas restricciones de seguridad en el momento de despliegue de la aplicación, aislando así las aplicaciones de la

complejidad de las implementaciones de seguridad, la plataforma J2EE hace portables una gran complejidad de implementaciones de seguridad.

1.5.6. 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)

Eclipse fue el entorno de desarrollo seleccionado para la implementación. Eclipse fue desarrollado originalmente por IBM[2] actualmente desarrollado por la Fundación Eclipse, una organización independiente que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

El Equipo de Desarrollo de Software en inglés Software Development Kit (SDK) de Eclipse incluye las herramientas de desarrollo de Java, ofreciendo un IDE con un compilador de Java interno y un modelo completo de los archivos fuente de Java. Esto permite técnicas avanzadas de refactorización y análisis de código.

Eclipse IDE posee una estructura modular, extensible mediante plugins[3], que le permite trabajar con cualquier tipo de recurso: gráficas, vídeo, modelos 3D, contenido web, etcétera (Eclipse Foundation, 2010). Algunos de los plugins que están disponibles son:

- Control de versiones con Subversion (Subclipse 1.6.2).
- Integración con Hibernate (Hibernate Tools 3.2.0).
- Integración con herramientas CASE como Visual Paradigm (SDE para Eclipse).

Otros lenguajes que también pueden utilizarse en Eclipse son: C/C++, PHP, Ruby, TCL o Javascript. Como IDE para Java cuenta con algunas funciones como: desarrollo de aplicaciones en grupo, unidad integrada de depuración y pruebas, compilación y construcción incremental.

1.6. FRAMEWORKS DE DESARROLLO PARA J2EE

En el ámbito del desarrollo de software se define como framework a una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, en base a la cual otro proyecto de software puede ser organizado y desarrollado. Además, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto. (Joomla Venezuela, 2010)

En general los framework son soluciones que contienen herramientas de apoyo a la construcción (ambiente de trabajo o desarrollo) y motores de ejecución (ambiente de ejecución). A continuación se dará una pequeña visión de los framework de desarrollo para J2EE que estableció la arquitectura del proyecto para el trabajo en las diferentes capas de la aplicación.

1.6.1. JAVA SERVER FACES

El framework Java Server Faces (JSF) es un marco de trabajo para construir interfaces de usuario en aplicaciones web del lado del servidor, basadas en tecnología Java, pero además se basa en el patrón Modelo Vista Controlador (MVC), utiliza JavaServer Pages (JSP[4]) como motor de plantillas. JSF está compuesto por: componentes UI[5] (User Interface Components) definidos por etiquetas y XML, eventos, validadores. Posibilita la reutilización de código, la separación de roles y la facilidad de uso de las herramientas, además tiene como meta hacer el desarrollo web más rápido y fácil. Permite almacenar automáticamente la información de los formularios, actualizando el mismo en el momento de ser mostrado al cliente, encapsula la lógica de manipulación de los eventos y la forma en que se muestran los componentes ya definidos. (Color Vivo Internet, 1999)

Este framework permite que los desarrolladores piensen en términos de componentes, eventos, backing beans y otras interacciones, en vez de peticiones, respuestas y marcas. (Sánchez Ramón, 2006)

Algunos de los componentes de JavaServer Faces son:

- Librerías de etiquetas personalizadas que permiten expresar una interfaz JavaServer Faces dentro de una página JSP.
- Un modelo de eventos en el lado del servidor.

- Beans administrados.

Utilizando JSF se puede conectar eventos generados en el cliente a código de la aplicación en el lado del servidor, mapear componentes UI a una página de datos del lado del servidor, construir una UI con componentes reutilizables y extensibles, guardar y restaurar el estado de la UI más allá de la vida de las peticiones de servidor. De forma general facilita el trabajo de construcción y mantenimiento de aplicaciones Web con UIs del lado del servidor. (Lou Torrijos, 2003)

Además, existen librerías específicas de gran importancia para JSF, una de ellas es la RichFaces, amplia librería de componentes visuales, la cual posee un framework avanzado para la integración de funcionalidades Ajax[6] en dichos componentes visuales, mediante el soporte de la librería Ajax4JSF. (Autentia, 2003)

1.6.2. SPRING FRAMEWORK

Como framework utilizado para la lógica del negocio se seleccionó Spring, framework de código abierto para el desarrollo de aplicaciones sobre la plataforma J2EE. Ofrece un amplio soporte a frameworks como Java Server Faces (JSF), se integra con frameworks de acceso a datos como Hibernate, ofreciendo un manejo seguro y eficiente de sus sesiones y manejando la configuración de la SessionFactory y las fuentes de datos JDBC en el contexto de la aplicación (Medín Piñeiro, y otros, 2006).

Este framework no solo se aplica en la plataforma Java, sino a otras como la plataforma de Microsoft con la versión Spring.Net, aunque es más popular por su influencia en el desarrollo actual de Java.

Algunas de las ventajas que proporciona Spring son (Johnson, 2005):

- Manejo de Beans con contexto de aplicación: puede organizar de forma efectiva objetos de la capa central y manejar conexiones.
- Utiliza Programación Orientada a Aspectos (AOP[7]) para el manejo de transacciones declarativas sin utilizar un contenedor EJB[8]. De esta forma, el control de transacciones se puede aplicar a cualquier POJO[9]. El control de transacciones de Spring no está atado a JTA (Java Transaction API[10]) y puede funcionar con diferentes estrategias de transacción.

- Árbol de excepciones de acceso a datos: proporciona un árbol de excepciones en lugar de SQLException. Para poder utilizar este árbol de excepciones, se debe definir un traductor de excepciones de acceso a datos dentro del fichero de configuración de Spring.
- Elimina la necesidad de usar distintos y variados tipos de ficheros de configuración.

Como característica fundamental de Spring está, la Inversión de Control (IoC), esta mediante la Inyección de Dependencia (DI) permite inyectar las dependencias de un bean en el momento de su creación utilizando un manejador externo, mediante esta técnica la IoC promueve el bajo acoplamiento de las clases. (Walls, y otros, 2005)

1.6.3. HIBERNATE

El framework utilizado para el acceso a datos fue Hibernate, el cual tiene como objetivo facilitar la persistencia de objetos Java en bases de datos relacionales y la consulta de estas bases de datos para obtener objetos (Suárez González, 2003).

Hibernate integra la estructura relacional y la objetual permitiendo que el programador trabaje desde Java de forma fácil y correcta, usando el modelo objetual, pero además se integra en cualquier tipo de aplicación por encima del contenedor de datos (Suárez González, 2003). Hibernate ofrece un lenguaje de consulta de datos llamado HQL (Hibernate Query Language) (Suárez González, 2003). Entre los beneficios que aporta está la independencia de la base de datos, el bajo acoplamiento entre negocio y persistencia, además proporciona un desarrollo rápido, cubriendo de manera sencilla y rápida la persistencia de una aplicación.

1.7. ARQUITECTURA TÉCNICA

Como propuesta de solución para el sistema se define una aplicación web con arquitectura en tres capas como se muestra en la figura 3, y basada en tecnología Java, a la que se conectarán los usuarios vía HTTPS de forma segura. Se utiliza como estilo arquitectónico n-capas, particularmente el patrón 3 capas, asociando los frameworks estudiados a cada una de las capas. La siguiente figura muestra la distribución del hardware utilizado en las implementaciones del sistema y las relaciones entre sus componentes de hardware y software, como también representa la división de las capas de la aplicación.

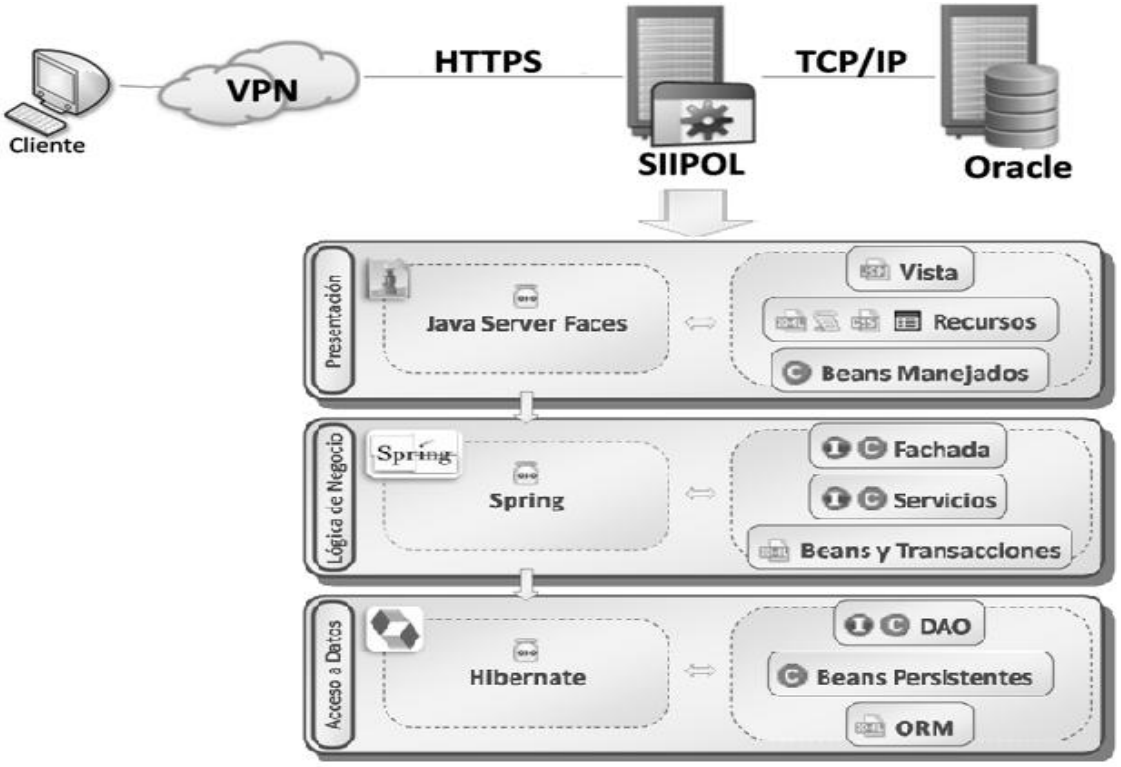


Figura 3. Vista de la arquitectura

El estilo arquitectónico en capas presenta distintas ventajas como: centralización de los aspectos de seguridad, transaccionalidad y no replicación de lógica de negocio en los clientes, además simplifica la comprensión y organización del desarrollo de sistemas complejos, reduciendo las dependencias de forma que las capas más bajas no conocen ningún detalle o interfaz de las superiores.

1.8. CONCLUSIONES

Se puede concluir, que en el desarrollo de toda investigación es esencial la fundamentación teórica, ya que, aborda conceptos necesarios para el entendimiento de la propuesta solución. Las herramientas, lenguajes y metodología descritos anteriormente, en la actualidad son temas fundamentales cuando se habla de desarrollo de software. Además, proporcionan la integración necesaria para facilitar el proceso de desarrollo del Módulo Acciones Especiales. La metodología RUP, es la indicada, ya que controlará todo el proceso de producción del software. Como herramienta Case Visual Paradigm, debido a que es una herramienta de modelado que utiliza UML y posee una buena integración con Eclipse, siendo este

último, un IDE que contiene gran variedad de herramientas y facilidades de integración. Java el lenguaje de programación, pues posee características que agilizan y facilitan el desarrollo de software. Con la utilización de todas estas herramientas integradas y como metodología RUP se logrará como resultado un módulo que cumpla con los requerimientos especificados.

CAPÍTULO 2. ANÁLISIS Y DISEÑO DE LA SOLUCIÓN PROPUESTA

2.1. INTRODUCCIÓN

Los flujos de Análisis y Diseño representan un paso fundamental en el resultado esperado. En el siguiente capítulo se mostrará el resultado obtenido a partir de la Ingeniería de Requerimientos, además serán descritos brevemente los procesos que se llevan a cabo en el Módulo de Acciones Especiales. Se mostrarán diagramas generados durante el proceso de Análisis y Diseño de los casos de uso correspondientes al módulo, los cuales conforman la parte fundamental de los modelos de análisis y diseño. Un buen análisis y diseño proporcionará rapidez y precisión a la hora de la implementación.

2.2. MODELO DE LA PROPUESTA SOLUCIÓN

En este epígrafe se comenzará a modelar la solución propuesta, para ello se describen sus requisitos, tanto funcionales como no funcionales. Además, los requisitos funcionales se modelaran en términos de casos de uso del sistema.

2.2.1. REQUISITOS DEL MÓDULO

Los requisitos son definidos durante las fases más tempranas del desarrollo de sistemas informáticos, y pueden verse como la especificación de lo que debería ser implementado (Ian Sommerville, Pete Sawyer. John Wiley & Sons, 1997). Los requisitos constituyen un tema de la Ingeniería de Software el cual tiene diferentes significados. De las definiciones que existen para requerimiento, una de las más acertadas es la que aparece en el glosario de la IEEE (Institute of Electrical and Electronics Engineers) (IEEE, 1990):

> (1) Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo.
>
> (2) Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal.
>
> (3) Una representación documentada de una condición o capacidad como en (1) o (2).

A menudo, los requerimientos de un sistema de software se clasifican en funcionales y no funcionales (Sommerville, 2005).

2.2.1.1. REQUISITOS FUNCIONALES

Los requerimientos funcionales de un sistema describen lo que el sistema debe hacer. Ian Sommerville los define en la séptima edición del libro Ingeniería de Software como (Sommerville, 2005):

- Son declaraciones de los servicios que debe proporcionar el sistema.
- Especifica la manera en que éste debe reaccionar a determinadas entradas.
- Especifica cómo debe comportarse el sistema en situaciones particulares.
- Pueden declarar explícitamente lo que el sistema no debe hacer.

Los requisitos funcionales asociados al Módulo de Acciones Especiales son los siguientes (ALBET, S.A, 2007):

RF 1. Validar la integridad de los datos introducidos por el usuario.

RF 2. Informar al usuario sobre posibles errores a la hora de la inclusión de datos.

RF 3. Mantener informado al usuario del resultado de las operaciones.

RF 4. Gestionar una solicitud de custodia de personalidades.

RF 4.1. Incluir una solicitud de custodia de personalidades.

RF 4.2. Guardar una solicitud de custodia de personalidades.

RF 4.3. Modificar los datos de una solicitud de custodia de personalidades.

RF 4.4. Ver los datos de una solicitud de custodia de personalidades.

RF 4.5. Imprimir o exportar a PDF una solicitud de custodia de personalidades.

RF 5. Gestionar una solicitud de custodia de prisionero.

RF 5.1. Incluir una solicitud de custodia de prisionero.

RF 5.2. Guardar una solicitud de custodia de prisionero.

RF 5.3. Modificar los datos de una solicitud de custodia de prisionero.

RF 5.4. Ver los datos de una solicitud de custodia de prisionero.

RF 5.5. Imprimir o exportar a PDF una solicitud de custodia de prisionero.

RF 6. Consultar una solicitud de acciones especiales.

RF 6.1. Buscar una solicitud de acción especial.

RF 6.2. Mostrar un listado de las solicitudes de acciones especiales ordenadas por un criterio.

RF 6.3. Ver los datos de una solicitud de acción especial.

RF 6.4. Imprimir o exportar a PDF una Solicitud de Custodia.

RF 7. Atender una solicitud de acciones especiales.

RF 7.1. Incluir una acción especial.

RF 7.2. Archivar una solicitud de acción especial.

RF 7.3. Ver los datos de una acción especial.

RF 7.4. Modificar una acción especial.

RF 7.1. Imprimir o exportar a PDF una acción especial.

2.2.1.2. REQUISITOS NO FUNCIONALES

Los requerimientos no funcionales son aquellos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste (Sommerville, 2005). Ian Sommerville los define en la séptima edición del libro Ingeniería de Software como (Sommerville, 2005):

- Son restricciones de los servicios o funciones ofrecidas por el sistema (fiabilidad, tiempo de respuestas, capacidad de almacenamiento, etc.).
- Definen propiedades y restricciones del sistema.

Los requisitos no funcionales más significativos del sistema al cual se integrará el Módulo de Acciones Especiales son los siguientes (ALBET, S.A, 2007):

Funcionalidad

RNF 1. Todos los mensajes de error del sistema deberán incluir una descripción textual del error.

RNF 2. El sistema permitirá el uso de reportes para presentar información al usuario.

Usabilidad

RNF 3. Los campos de texto tendrán un tamaño estándar de acuerdo con el espacio con que se cuente en el área de la página y en la medida que se llene esa área primaria agregar la barra de desplazamiento vertical.

RNF 4. Se evitará el uso de los Botones de Opción en la medida de lo posible, potenciando la utilización de cuadros de selección, a fin de economizar espacio de trabajo.

RNF 5. No se utilizarán textos extensos para las etiquetas de la interfaz de usuario.

Interfaz de usuario

RNF 6. El sistema brindará una interfaz amigable para sus usuarios. El nivel de funcionamiento del sistema deberá corresponder el nivel medio de conocimiento informático de los usuarios.

RNF 7. El sistema proporcionará claridad y buena organización de la información, permitiendo la interpretación correcta e inequívoca de esta.

RNF 8. El sistema aplicará normas de diseño que permitan la distinción visual entre los elementos de las tablas de resultados, a través del uso de colores.

RNF 9. Todos los textos y mensajes en pantalla aparecerán en idioma español. Los errores serán visibles al usuario e incluirán sugerencias de las posibles soluciones.

RNF 10. El sistema presentará los términos capitalizados, es decir, tendrán su primera letra en mayúsculas.

RNF 11. Los textos asociados a los botones de opción deberán estar redactados claramente.

RNF 12. Ante la ocurrencia de un error, el sistema señalará los campos que generan el mismo, ya sea porque contienen información incompleta, o porque se encuentran vacíos.

Fiabilidad

RNF 13. El sistema estará disponible durante 24 horas, los 7 días de la semana, los 365 días del año.

RNF 14. El sistema hará un uso eficiente de la capacidad de almacenamiento de los servidores.

Seguridad

RNF 15. El sistema manejará la seguridad de acceso y administración de usuarios: otorgamiento de privilegios y roles, asignación de perfiles.

RNF 16. El sistema implementará el uso de campos obligatorios y validaciones para garantizar la integridad de la información que se introduce por el usuario.

RNF 17. Ninguna información que se haya ingresado en el sistema será eliminada físicamente de la base de datos, independientemente de que para el sistema, este elemento ya no exista.

Restricciones de diseño

RNF 18. El sistema será una aplicación web centralizada.

RNF 19. El sistema se implementará usando la plataforma JAVA.

RNF 20. El sistema estará basado en un estilo arquitectónico en capas.

RNF 21. El sistema usará el Framework de Presentación JSF para manejar la Capa de Presentación.

RNF 22. El sistema usará el Framework Spring para manejar la Capa de Lógica de Negocio, así como para el manejo de transacciones, concurrencia y seguridad.

RNF 23. El sistema usará el Framework de Persistencia Hibernate para manejar la capa de Acceso a Datos.

2.2.2. DIAGRAMA DE CASOS DE USO

El Módulo Acciones Especiales cuenta con un total de cuatro Casos de Uso (CU) los cuales agrupan las funcionalidades referidas en el epígrafe anterior. A continuación se presenta el diagrama de casos de uso del módulo.

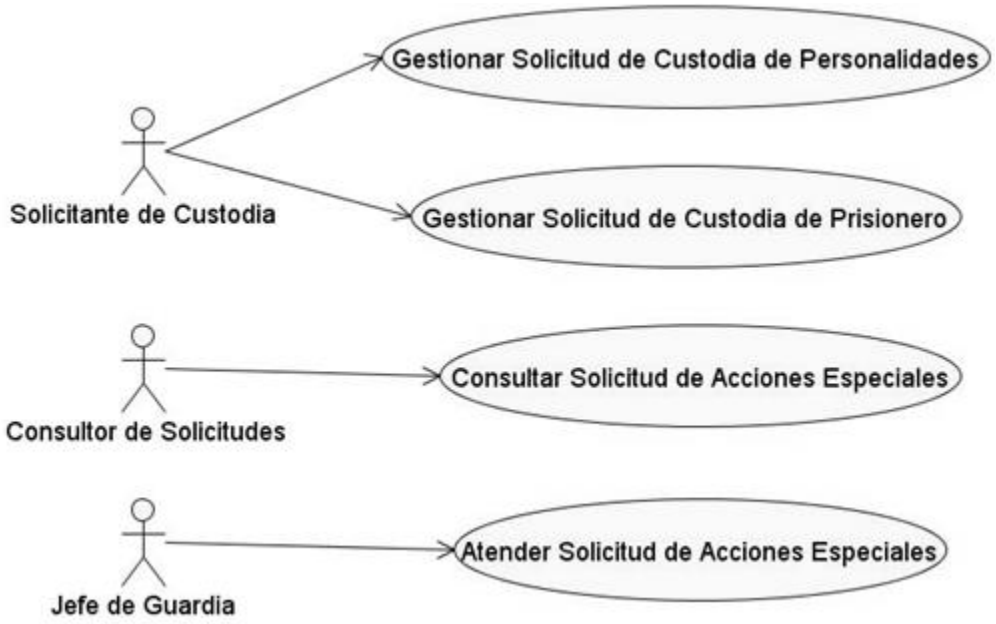


Figura 4. Diagrama de Casos de Uso

2.2.3. DESCRIPCIÓN DE LOS CU DEL SISTEMA

"Un caso de uso es una colección de escenarios con éxito y fallo relacionados, que describe a los actores utilizando un sistema para satisfacer un objetivo" (Larman, 2003). RUP proporciona una definición alternativa, aunque similar, de un caso de uso: "un conjunto de instancias de caso de uso, donde cada instancia es una secuencia de acciones que un sistema ejecuta, produciendo un resultado observable de valor para un actor particular" (Larman, 2003).

Las descripciones se realizarán a través de casos de uso a nivel resumen y su objetivo es proponer de manera general las prestaciones del módulo y garantizar la construcción de un software que cumpla con

las expectativas, necesidades y realidades de la organización en estudio. A continuación se presenta la descripción de los CU del Módulo Acciones Especiales:

Nombre del CU	Gestionar Solicitud de Custodia de Personalidades.
Propósito	Incluir, ver y modificar una Solicitud de Custodia de Personalidades.
Actor (es)	Solicitante de Custodia
Descripción	El caso de uso inicia cundo el actor accede a la opción que le permite realizar alguna acción sobre una Solicitud de Custodia de Personalidades. El actor podrá incluir, ver y modificar una Solicitud de Custodia de Personalidades. Si selecciona la opción de incluir una Solicitud de Custodia de Personalidades se le mostrará de manera predeterminada un resumen de los datos del ente solicitante y le permitirá introducir los datos de la Solicitud de Custodia de Personalidades como son: hora de traslado, descripción, dirección del origen de traslado, dirección del destino de traslado, persona a la que se le va a realizar la custodia entre otros datos. Si selecciona la opción de ver los datos de la solicitud el sistema muestra los datos de la solicitud seleccionada, permitiendo realizar diferentes acciones sobre ella como son: modificar si se tienen los permiso requeridos, e imprimir/exportar a PDF. Si selecciona la opción de modificar, muestra los datos de la solicitud y brinda la posibilidad de cambiar los valores que originaron la no aceptación de la solicitud.
Referencia	RF 1, RF 2, RF 3, RF 4

Nombre del CU	Gestionar Solicitud de Custodia de Prisionero.

Propósito	Incluir, ver y modificar una Solicitud de Custodia de Prisionero.
Actor (es)	Solicitante de Custodia.
Descripción	El caso de uso inicia cundo el actor accede a la opción que le permite realizar alguna acción sobre una Solicitud de Custodia de Prisionero. El actor podrá incluir, ver y modificar una Solicitud de Custodia de Prisionero. Si selecciona la opción de incluir una Solicitud de Custodia de Prisionero se le mostrará de manera predeterminada un resumen de los datos del ente solicitante y le permitirá introducir los datos de la Solicitud de Custodia de Prisionero como son: hora y origen de traslado, hora y destino de traslado, fecha de traslado, descripción, persona a custodiar entre otros. Si selecciona la opción de ver los datos de la solicitud el sistema muestra los datos de la solicitud seleccionada, permitiendo realizar diferentes acciones sobre ella como son: modificar si se tienen los permiso requeridos, e imprimir/exportar a PDF. Si selecciona la opción de modificar, muestra los datos de la solicitud y brinda la posibilidad de cambiar los valores que originaron la no aceptación de la solicitud.
Referencia	RF 1, RF 2, RF 3, RF 5

Nombre del CU	Consultar Solicitud de Acciones Especiales.
Propósito	Buscar y listar de manera ordenada un resumen de los datos de las Solicitudes de Acciones Especiales coincidentes con uno o varios criterios de búsqueda a partir de la Agenda de Trabajo.
Actor (es)	Consultor de Solicitudes.

Descripción	El caso de uso se inicia cuando el actor selecciona la opción de consultar una Solicitud de Acción Especial desde la Agenda de Trabajo. El sistema brinda la posibilidad de introducir los datos elementales de búsqueda para realizar la consulta como son: número de comunicación de la Solicitud de Acciones Especiales, ente solicitante, fecha de recepción, importancia, y tipo de servicio (Custodia de Prisionero o Custodia de Personalidades). El actor introduce los datos y el sistema busca los resultados y muestra un listado de posibles coincidencias, permitiendo acceder al contenido de una solicitud o atender una solicitud seleccionada. El sistema permite imprimir los resultados obtenidos terminando así el caso de uso.
Referencia	RF 1, RF 2, RF 3, RF 6

Nombre del CU	Atender Solicitud de Acciones Especiales.
Propósito	Atender, darle respuesta a una Solicitud de Custodia.
Actor (es)	Jefe de Guardia
Descripción	El caso de uso se inicia cuando el actor selecciona la opción que le permite atender una Solicitud Custodia. El sistema brinda la posibilidad de atender la solicitud previamente seleccionada introduciendo los siguientes datos: hora en la que comenzó la custodia, hora en la que se terminó la custodia, descripción e incidencias; luego de introducir los datos pertinentes el sistema procesa la información y cambia el estado de la solicitud. El caso de uso termina.
Referencia	RF 1, RF 2, RF 3, RF 7

2.3. MODELO DE ANÁLISIS

El modelo de análisis ofrece una especificación más precisa de los requisitos que la que se tiene como resultado de la captura de requisitos, incluyendo al modelo de casos de uso; este modelo estructura los requisitos de modo que facilita su comprensión, preparación, modificación y mantenimiento. Además, se describe utilizando el lenguaje de los desarrolladores, introduce un mayor formalismo y es utilizado para razonar sobre los funcionamientos internos del sistema, también puede considerarse como una primera aproximación al modelo de diseño, y es por tanto una entrada fundamental cuando se da forma al sistema en el diseño y en la implementación. (Jacobson, y otros, 2000)

Entre las características fundamentales del modelo de análisis están (Jacobson, y otros, 2000):

- Estructurado por clases y paquetes estereotipados; proporciona la estructura a la vista interna.
- Utilizado por los desarrolladores para comprender cómo debería darse forma al sistema, es decir, cómo debería ser diseñado e implementado.
- Define realizaciones de casos de uso, y cada una de ellas representa el análisis de un caso de uso del modelo de casos de uso.

Parte fundamental del modelo de análisis son las clases de análisis, las cuales representan una abstracción de una o varias clases y/o subsistemas del diseño del sistema, y se clasifican en tres tipos fundamentales (Jacobson, y otros, 2000):

- Clases de Interfaz: modelan la interacción entre el sistema y sus actores.
- Clases entidad: modelan información que posee larga vida y que es a menudo persistente.
- Clases de control: coordinan la realización de uno o unos pocos casos de uso coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso.

2.3.1. DIAGRAMAS DE CLASES DE ANÁLISIS (DCA)

Uno de los artefactos principales del flujo de análisis son los diagramas de clases de análisis, los cuales muestran las clases y sus relaciones en la realización de los distintos casos de uso (Jacobson, y otros, 2000).

A continuación se representarán los diagramas de clases de análisis de cada uno de los casos de uso correspondientes al Módulo de Acciones Especiales.

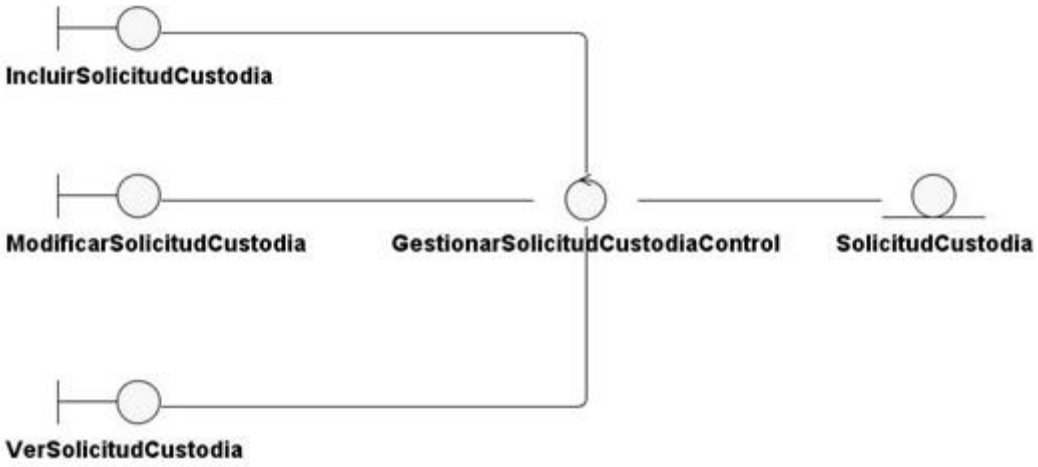


Figura 5. DCA CU Gestionar Solicitud de Custodia de Personalidad y Prisionero

Figura 6. DCA CU Consultar Solicitudes de Acciones Especiales

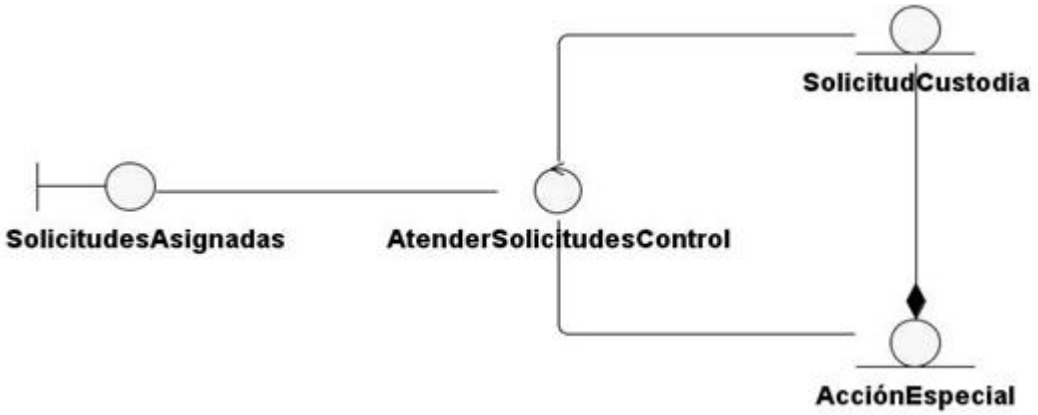


Figura 7. DCA CU Atender Solicitud de Acciones Especiales

2.3.2. DIAGRAMAS DE COLABORACIÓN DEL ANÁLISIS (DCOA)

En un caso de uso la secuencia de acciones comienza cuando un actor invoca al caso de uso mediante el envío de un mensaje al sistema, un objeto interfaz recibirá este mensaje del actor, el objeto interfaz enviará a su vez un mensaje a algún otro objeto, y de esta forma los objetos interactúan para llevar a cabo el caso de uso. En el análisis, dicho proceso se muestra con lo que se nombra diagrama de colaboración, siendo el objetivo fundamental identificar requisitos y responsabilidades sobre los objetos, y no identificar secuencias de interacción detalladas y ordenadas cronológicamente. (Jacobson, y otros, 2000)

Los diagramas de colaboración de análisis de cada uno de los casos de uso correspondientes al Módulo de Acciones Especiales son los siguientes.

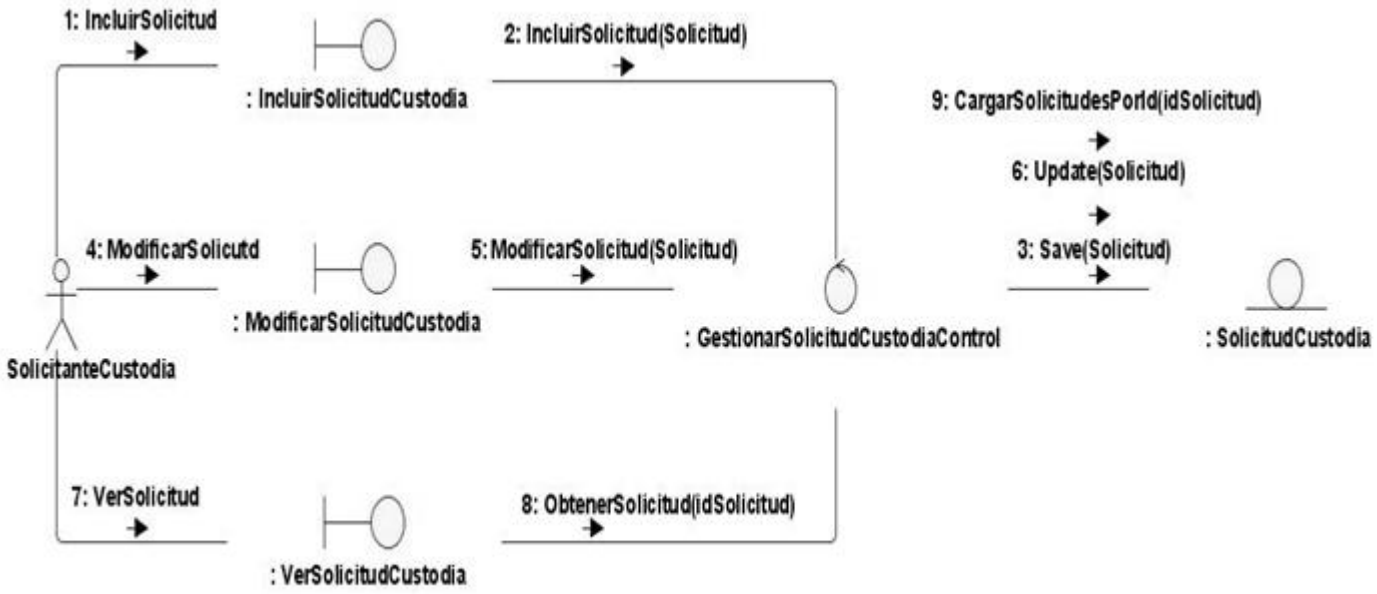


Figura 8. DCOA CU Gestionar Solicitud de Custodia de Personalidad y Prisionero

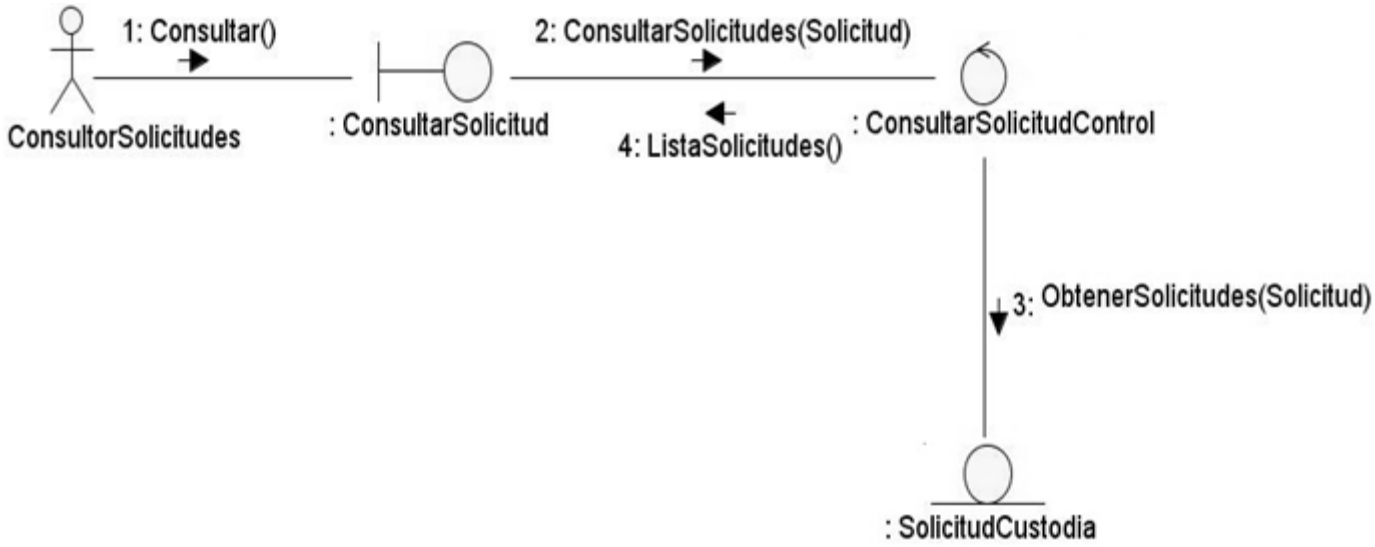


Figura 9. DCOA CU Consultar Solicitudes de Acciones Especiales

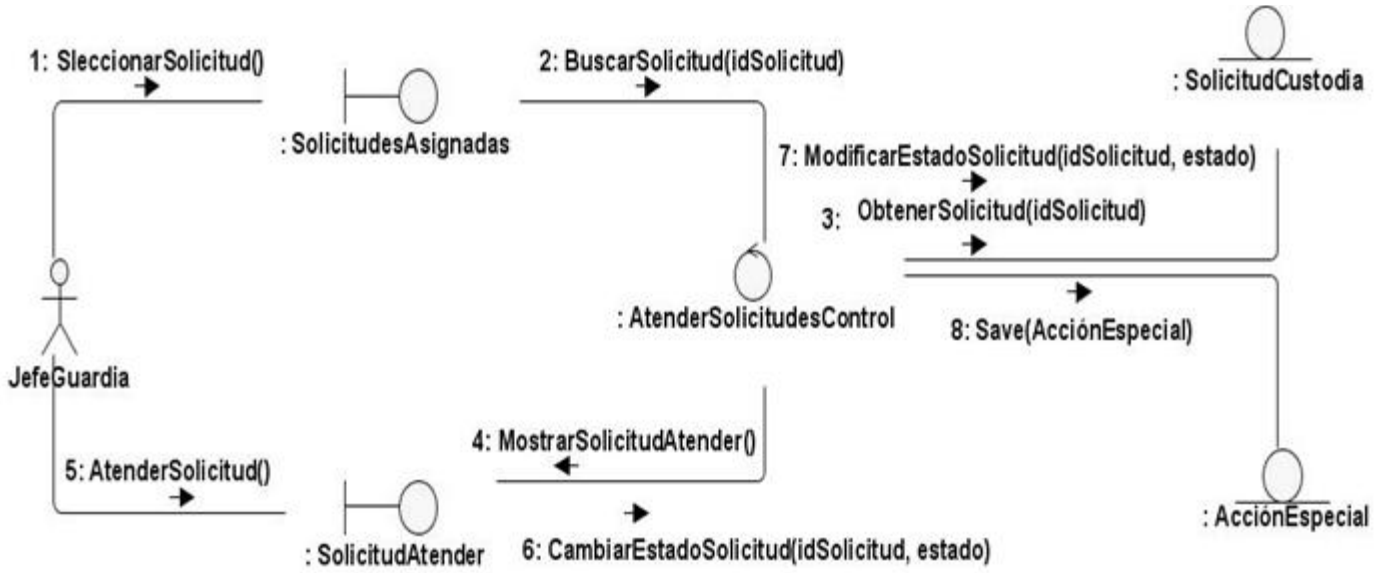


Figura 10. DCOA CU Atender Solicitud de Acciones Especiales

2.4. MODELO DE DISEÑO

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación tienen impacto en el sistema a considerar. Además, sirve como abstracción de la implementación del sistema y es, de este modo, utilizada como una entrada fundamental de las actividades de implementación. (Jacobson, y otros, 2000)

Algunos de los tipos de clases del diseño son (ALBET, S.A, 2007):

- Beanes manejados: son simples clases Java que mapean los valores de los formularios JSF a sus propiedades, estas clases son manejadas por el framework en el sentido de que este las crea.
- Clases fachadas: son las clases encargadas de proporcionar un punto de entrada a las funcionalidades que brinda un módulo. No se incluirá lógica de negocio en la fachada, la fachada delegará en las clases de servicio y las clases del dominio.
- Clases de servicio: son las clases donde descansa la implementación de la lógica necesaria para coordinar las acciones ejecutadas por las clases del modelo. Generalmente los métodos de las clases de servicio se corresponden con el flujo del caso de uso que implementa. Estas clases también encapsulan todo el código necesario para dar respuesta a las peticiones de la fachada.
- Clases DAO: estas clases tienen como misión encapsular toda la lógica asociada a la persistencia de objetos. Las clases DAO generalmente usan las clases del framework de persistencia para hacer todas las operaciones sobre las clases persistentes (insertar, actualizar y borrar, recuperar).
- Clases del dominio: estas clases son el núcleo de la capa de negocio, expresan los conceptos del negocio, las relaciones que se establecen entre los distintos elementos del negocio, así como su ciclo de vida. Estas clases mantendrán el estado de los elementos del negocio así como implementarán el comportamiento asociado a estos.
- Clases validadoras: Estas clases están fuertemente acopladas al framework JSF, ya que implementan interfaces base del mismo. Las validaciones aseguran que un dato introducido en un formulario JSF tenga un valor correcto.

- Clases convertidoras: Estas clases están fuertemente acopladas al framework JSF ya que implementan interfaces base del mismo. Los convertidores aseguran que un tipo de datos introducido en un formulario JSF sea el correcto.

2.4.1. DIAGRAMAS DE CLASES DEL DISEÑO (DCD)

El diagrama de clases del diseño es el utilizado para representar clases del diseño y sus objetos, como también los subsistemas que contienen dichas clases y las relaciones de cada uno de estos elementos participantes en la realización de uno o varios casos de uso. (Jacobson, y otros, 2000)

En la realización de los diagramas de clases del diseño se utilizaron diferentes patrones de diseños, los más significativos son los siguientes:

- Patrón Fachada, consiste en proveer una interfaz única a un conjunto de interfaces en un subsistema, o sea, define una interfaz a alto nivel que hace más fácil el uso del subsistema (Gamma, y otros, 1994).
- Patrón DAO (Objetos de Acceso a Datos), consiste en utilizar un objeto como medio de acceso a base de datos, para abstraer y encapsular todo el acceso a la fuente de datos. El DAO administra las conexiones con la fuente de datos, además recupera y almacena información. (Oracle Corporation, 2002)
- Patrón Interface, consiste en que, instancias de clases proveen datos y servicios a instancias de otras clases. Se utiliza cuando se desea que una clase que hace uso de los servicios proporcionados por otras clases, permanezca independiente de estas. (Grand, 2002)
- Patrón de asignación de responsabilidad Bajo Acoplamiento, el cual se encarga de asignar las responsabilidades de modo que se mantenga un bajo acoplamiento, o sea, el modo de dar soporte a poca dependencia y a una mayor reutilización (Larman, 1999).

Los diagramas de clases del diseño de cada uno de los casos de uso correspondientes al Módulo de Acciones Especiales son los siguientes.

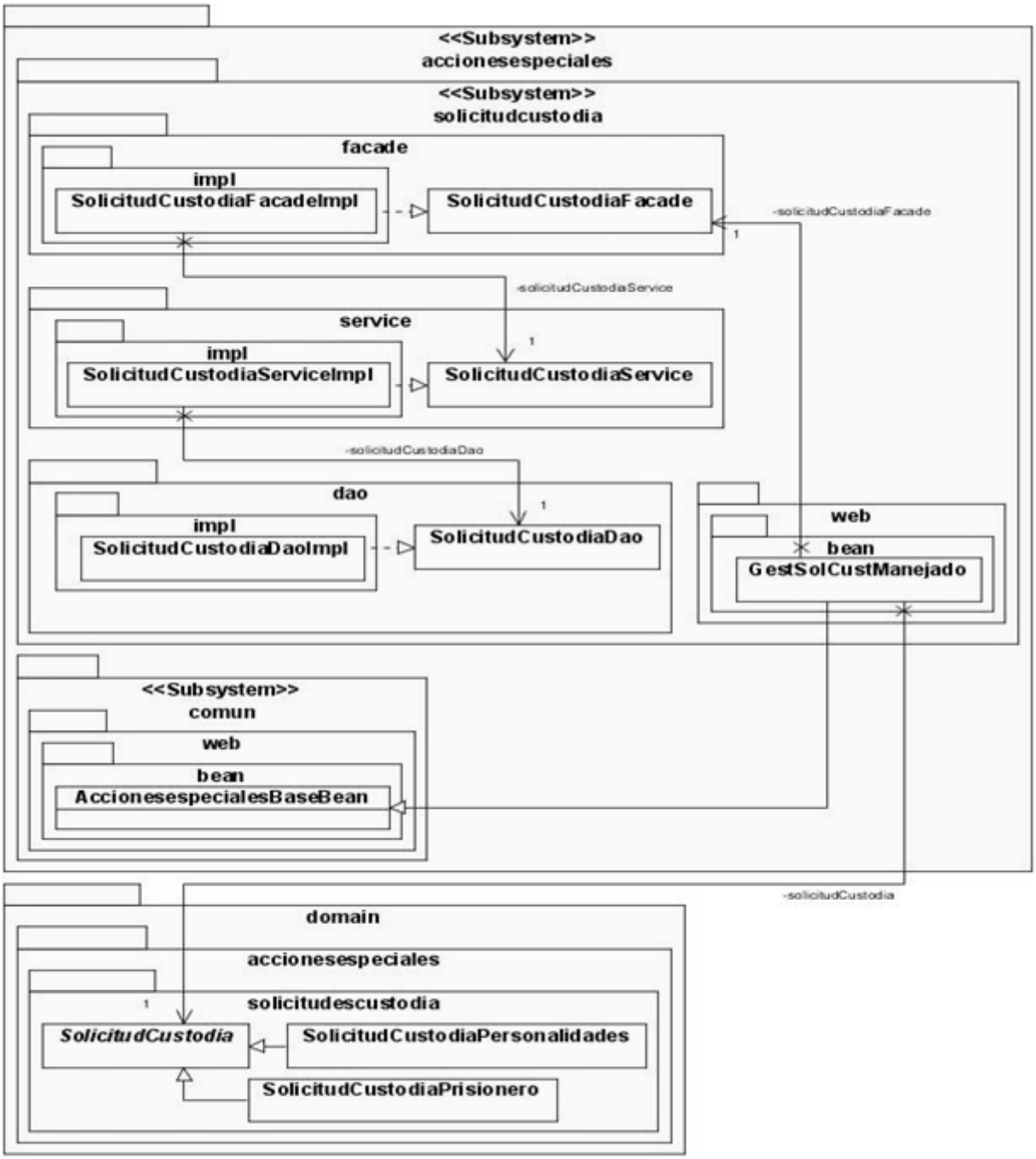


Figura 11. DCD CU Gestionar Solicitud de Custodia de Personalidad y Prisionero

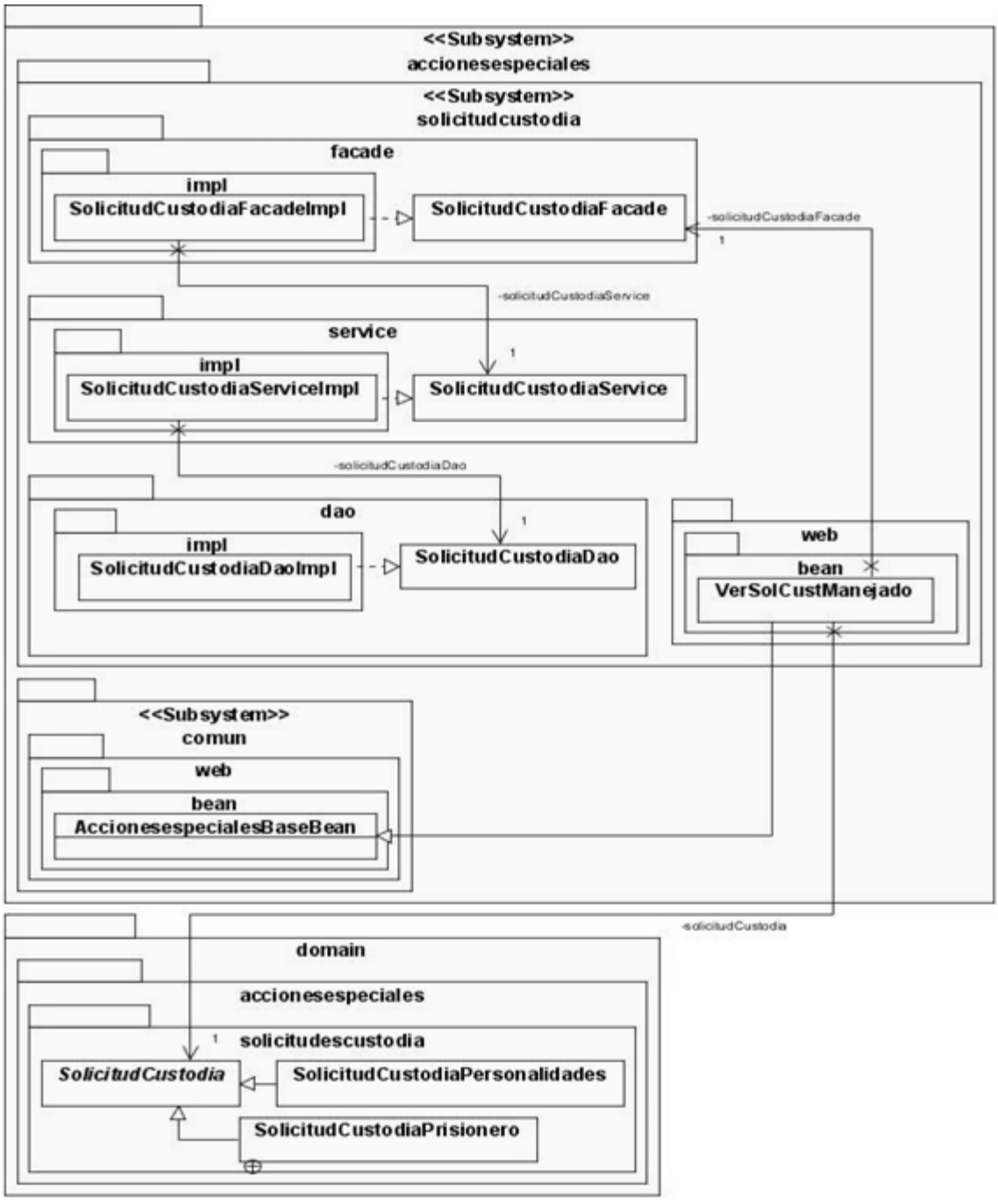


Figura 12. DCD CU Consultar Solicitudes de Acciones Especiales

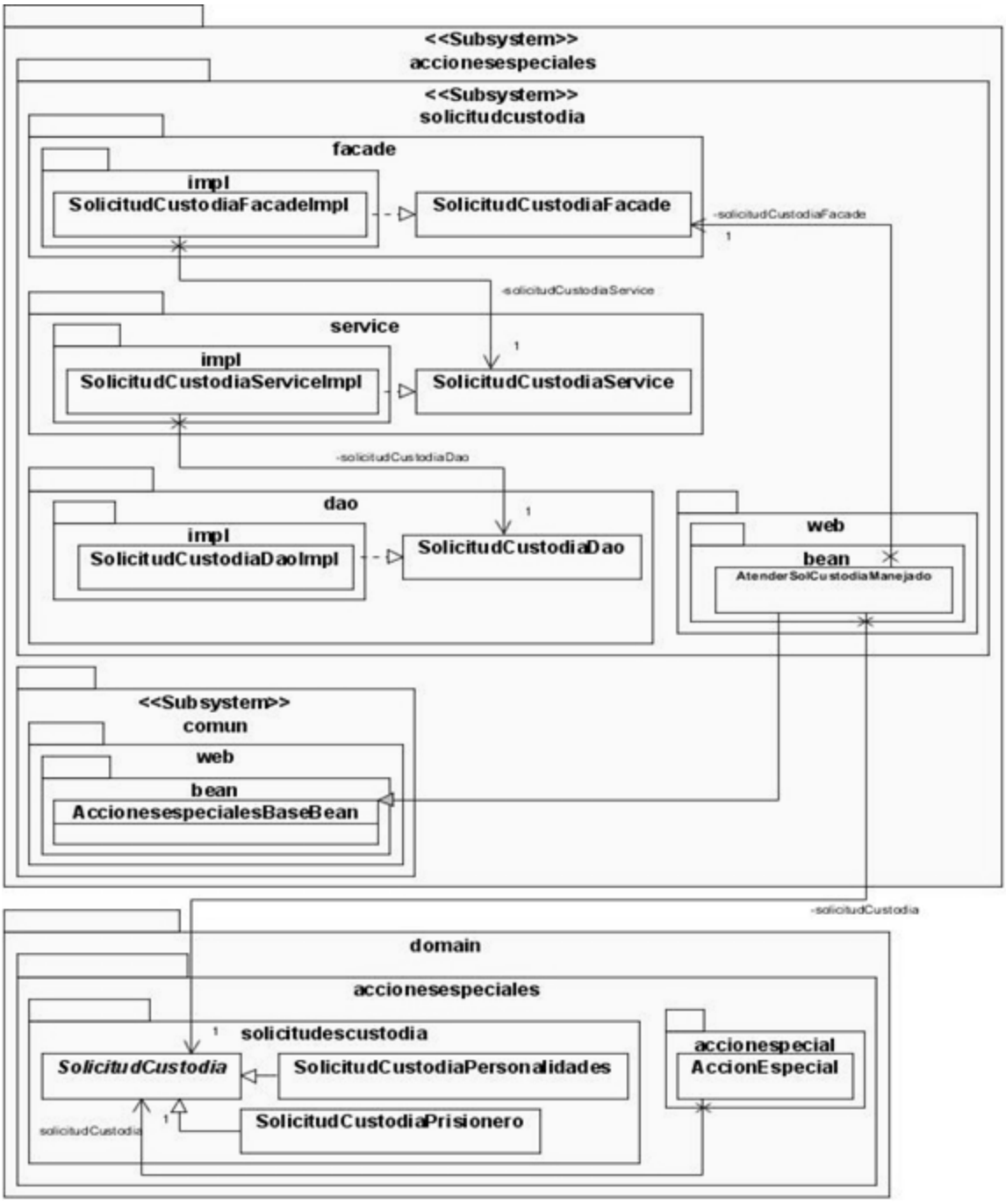


Figura 13. DCD CU Atender Solicitud de Acciones Especiales

A continuación se describen las principales clases que presentan mayor importancia en el diseño de la solución.

Nombre: SolicitudCustodiaServiceImpl		
Tipo de clase: Servicio		
Descripción: Servicio encargado de manejar la lógica de negocio tanto de la solicitud de custodia como de la acción especial.		
Para cada atributo:		
Nombre	*Tipo*	*Descripción*
solicitudCustodiaDao	SolicitudCustodiaDao	Objeto de acceso a datos tanto para las solicitudes de custodia como para las acciones especiales.
agendaTrabajoFacade	AgendaTrabajoFacade	Fachada para acceder al módulo Gestión Administrativa.
Para cada responsabilidad:		
Nombre	*Tipo*	*Descripción*
actualizarSolicitudCustodia	void	Método para actualizar una solicitud de custodia.
actualizarAccionEspecial	void	Método para actualizar una Acción Especial.
salvarAccionEspecial	void	Método para salvar una Acción Especial.
cargarSolicitudCustodiaPorId	SolicitudCustodia	Método que devuelve una solicitud de custodia dado el id de dicha solicitud.
salvarSolicitudCustodia	void	Método para salvar una solicitud de custodia.

Nombre: GestSolCustManejado
Tipo de clase: Bean manejado
Descripción: Clase encargada de manejar la lógica por detrás de las páginas de presentación asociadas a la gestión de solicitudes de custodia.

Para cada atributo:		
Nombre	*Tipo*	*Descripción*
solicitudCustodia	SolicitudCustodia	Objeto de la clase solicitud de custodia, contenedor de los datos de la solicitud de custodia.
solicitudCustodiaFacade	SolicitudCustodiaFacade	Fachada para acceder a los métodos del Módulo Acciones Especiales.
administracionFacade	AdministracionFacade	Fachada para acceder a los métodos del Módulo Administración.
gestionAdministrativaFacade	GestionAdministrativaFacade	Fachada para acceder a los métodos del Módulo Gestión Administrativa.
Para cada responsabilidad:		
Nombre	*Tipo*	Descripción
incluirPersona	void	Método para incluir la persona a la cual se le realizará la custodia.
salvarSolicitudCustodia	void	Método para salvar una solicitud de custodia.
actualizarSolicitud	void	Método para actualizar una solicitud de custodia.
modificarsolicitud	void	Método para modificar una solicitud de custodia.

Nombre: AtenderSolCustodiaManejado
Tipo de clase: Bean manejado
Descripción: Clase encargada de manejar la lógica por detrás de las páginas de presentación asociadas al proceso de atender una solicitud de custodia.
Para cada atributo:

Nombre	*Tipo*	*Descripción*
solicitudCustodia	SolicitudCustodia	Objeto de la clase solicitud de custodia contenedor de los datos de la solicitud de custodia.
accionEspecial	AccionEspecial	Objeto de la clase solicitud de acción especial contenedor de los datos de la acción especial que se creará.
solicitudCustodiaFacade	SolicitudCustodiaFacade	Fachada para acceder a los métodos del Módulo Acciones Especiales.
gestionAdministrativaFacade	GestionAdministrativaFacade	Fachada para acceder a los métodos del Módulo Gestión Administrativa.
Para cada responsabilidad:		
Nombre	*Tipo*	*Descripción*
archivarSolicitud	void	Método para archivar una solicitud de custodia.
actualizar	void	Método para actualizar una Acción Especial.
atenderSolicitud	void	Método para atender una solicitud de custodia.

Nombre: AccionEspecial		
Tipo de clase: Dominio		
Descripción: Clase encargada de contener los datos asociados con una Acción Especial.		
Para cada atributo:		
Nombre	*Tipo*	*Descripción*
incidencias	String	Atributo contenedor de las incidencias de una acción especial.
descripcion	String	Atributo contenedor de la descripción de una acción especial.
horaRealInicioCustodia	Date	Atributo contenedor de la hora de inicio de una acción especial.
horaRealFinCustodia	Date	Atributo contenedor de la hora de fin de una acción especial.

Nombre: SolicitudCustodia		
Tipo de clase: Dominio		
Descripción: Clase encargada de contener los datos asociados con una solicitud de custodia.		
Para cada atributo:		
Nombre	*Tipo*	*Descripción*
horaSolicitadaInicioCustodia	Date	Atributo contenedor de la hora de inicio de la solicitud de custodia.
fechaInicioCustodia	Date	Atributo contenedor de la fecha de inicio de la solicitud de custodia.
personaCustodiada	Persona	Objeto de la clase persona contenedor de los datos de la persona a custodiar.
baseLegal	Set<Articulo>	Lista contenedora de los artículos que conforman la base legal de una solicitud de custodia.

2.4.2. DIAGRAMA DE CLASES DEL DOMINIO

Primeramente se hace necesario especificar que, un diagrama de clases persistentes muestra todas las clases capaces de mantener su valor en el espacio y en el tiempo. (Jacobson, y otros, 2000)

El diagrama de clases del dominio representa solamente las clases del dominio del subsistema y su interacción directa con las demás clases del dominio de otros subsistemas. Este diagrama es similar a un diagrama de clases persistentes, aunque no se ha querido categorizar como tal, teniendo en cuenta que debido a refactorizaciones que se han realizado, existen dentro de los dominios algunas micro-arquitecturas, clases no persistentes o clases que persisten dentro de otras clases. Debido a esto no se considera llamarle diagrama de clases persistentes, aunque al final no hay mucha diferencia en la práctica. (ALBET, S.A, 2009)

El diagrama de clase del domino asociado al Módulo de Acciones Especiales es el siguiente.

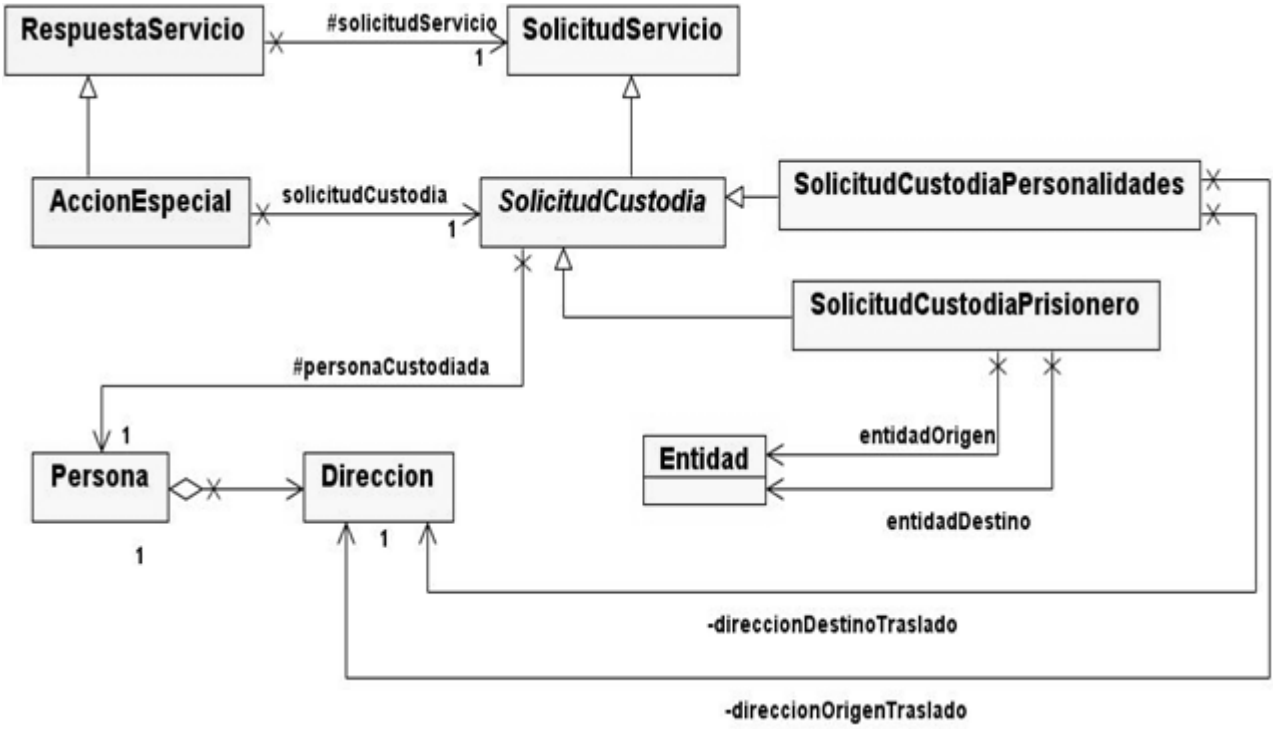


Figura 14. Diagrama de clases del dominio

2.4.3. DIAGRAMAS DE CONTRATO ENTRE PAQUETES (DCEP)

Un diagrama de contrato entre paquetes es un diagrama de secuencia que muestra el paso de mensajes entre las diferentes carpetas, sin mostrar todo el flujo de eventos que ocurre dentro de las mismas cuando llega una determinada petición. Es un diagrama de secuencia de alto nivel, su objetivo es brindar a los programadores, sin incurrir en demasiado nivel de detalle, lo necesario para la realización de los casos de uso. Este artefacto ahorra tiempo de diseño, ya que su realización no requiere demasiadas especificaciones, y simplifica de manera considerable los diagramas necesarios para las realizaciones. (ALBET, S.A, 2009)

El diagrama de contrato entre paquetes surge por la necesidad de simplificar en cuestiones de tiempo y complejidad el proceso de diseño de cada uno de los casos de uso del sistema, por lo cual la dirección del

proyecto decidió realizar los diagramas de secuencia mediante la interacción entre subsistemas a lo cual denominaron “Diagrama de contrato entre paquetes”, convirtiéndose en un entregable del proyecto.

Los diagramas de contrato entre paquetes asociados a los diferentes casos de uso del Módulo de Acciones Especiales son los siguientes.

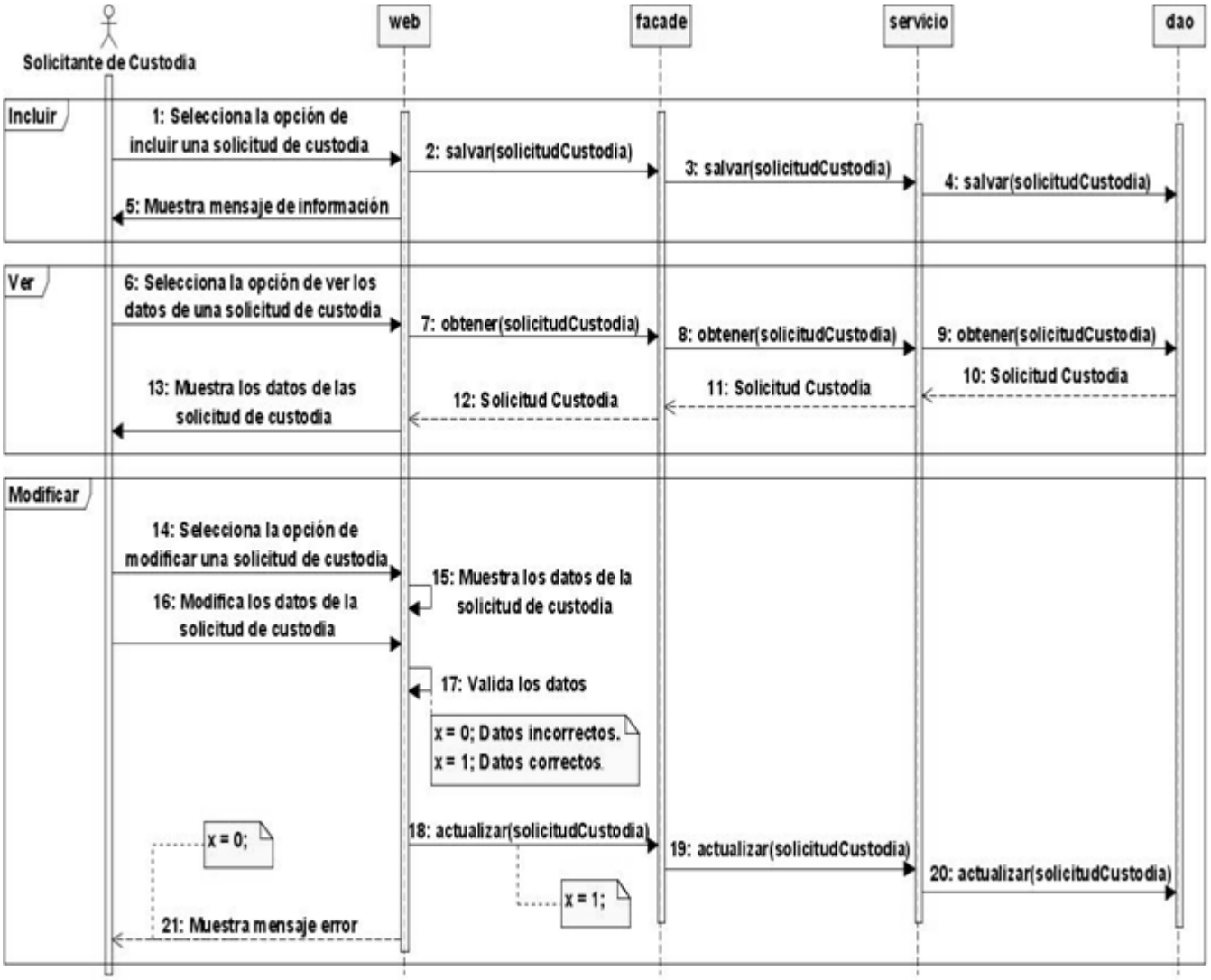


Figura 15. DCEP CU Gestionar Solicitud de Custodia de Personalidad y Prisionero

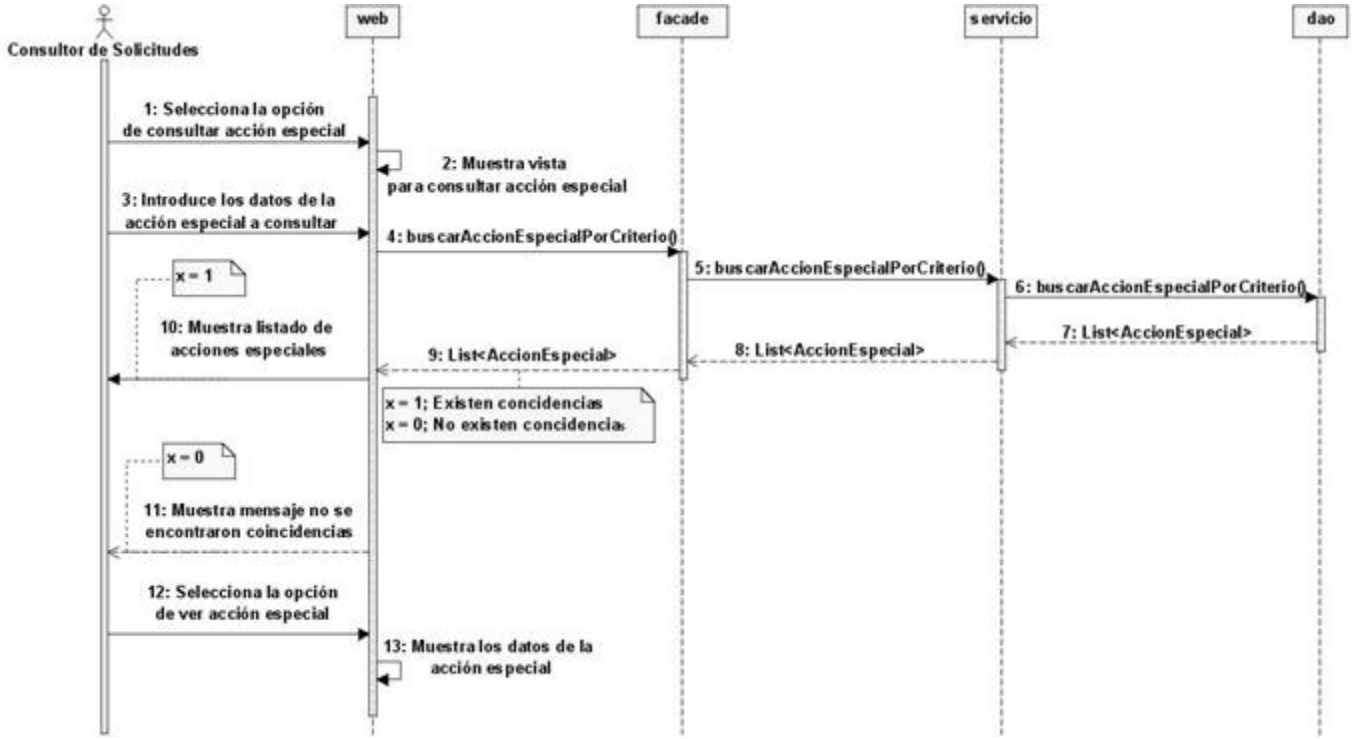


Figura 16. DCEP CU Consultar Solicitudes de Acciones Especiales

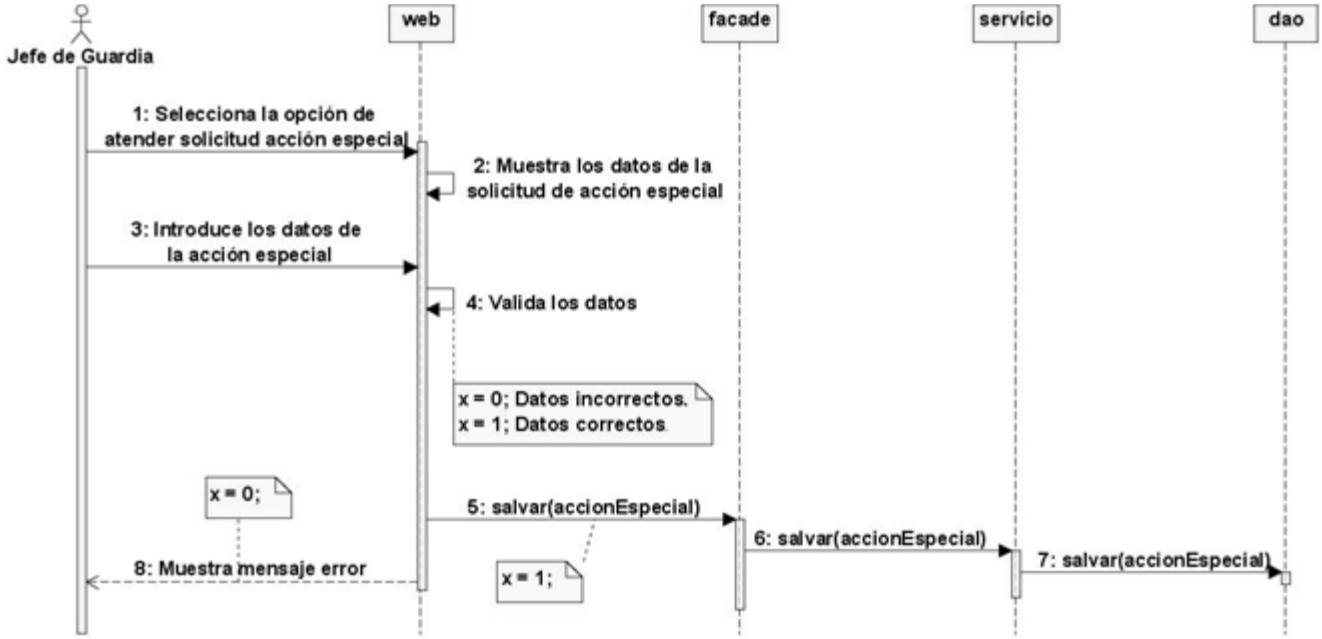


Figura 17. DCEP CU Atender Solicitud de Acciones Especiales

2.5. MODELO DE DESPLIEGUE

El modelo de despliegue es un modelo de objeto que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. El modelo de despliegue se utiliza como entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal en su diseño. (Jacobson, y otros, 2000)

Del modelo de despliegue se puede decir que (Jacobson, y otros, 2000):

- Cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo hardware similar.
- Los nodos poseen relaciones que representan medios de comunicación entre ellos.
- El modelo de despliegue puede describir diferentes configuraciones de red.

2.5.1. DIAGRAMA DE DESPLIEGUE

El diagrama de despliegue es el tipo de diagrama utilizado normalmente para representar el modelo de despliegue (Rational Software Corporation, 2003). A continuación se muestra el diagrama de despliegue que representa cómo será desplegado el sistema del cual forma parte el Módulo de Acciones Especiales.

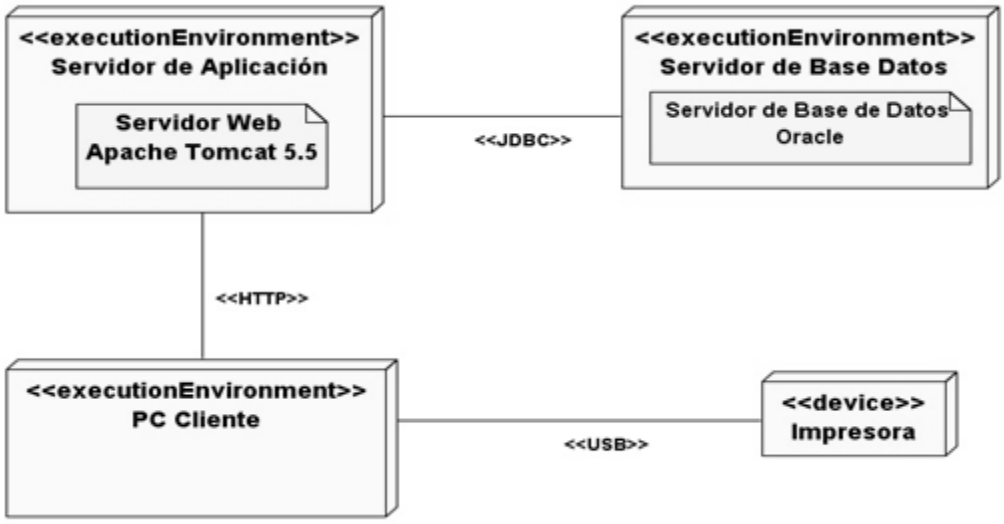


Figura 18. Diagrama de despliegue

2.6. CONCLUSIONES

En este capítulo se dio cumplimiento a los requisitos funcionales y no funcionales. Una vez realizado un correcto análisis y diseño se concluye en que, puede desarrollarse de forma cómoda el modelo de implementación de la solución propuesta, debido a que proporcionan todos los artefactos necesarios como son: los diagramas de clases del diseño, diagrama de clases del dominio y diagramas de contrato entre paquetes.

CAPÍTULO 3. IMPLEMENTACIÓN Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

3.1. INTRODUCCIÓN

Este capítulo es orientado a la implementación de la propuesta solución. Se obtendrá el modelo de implementación, el cual estará formado por los diagramas de subsistema de implementación y componentes, estos exponen los diferentes subsistemas de implementación junto a los componentes que los forman. Además, se verificará el resultado de la implementación, se hará uso de los artefactos generados durante el flujo de trabajo de pruebas, exponiendo resultados obtenidos por diferentes tipos de pruebas realizadas al módulo.

3.2. MODELO DE IMPLEMENTACIÓN

El modelo de implementación describe cómo los elementos del modelo de diseño y las clases se implementan en términos de componentes, como ficheros de código fuente, ejecutables, etc. El modelo de implementación describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros. (Jacobson, y otros, 2000)

3.2.1. DIAGRAMA DE SUBSISTEMAS DE IMPLEMENTACIÓN

Los subsistemas de implementación proporcionan una forma de organizar los artefactos del modelo de implementación en trozos más manejables. Un subsistema puede estar formado por componentes, interfaces y otros subsistemas (recursivamente). Además, un subsistema puede implementar -y así proporcionar - las interfaces que representan la funcionalidad que exportan en forma de operaciones. (Jacobson, y otros, 2000)

Un subsistema de implementación se manifiesta a través de un mecanismo de empaquetamiento concreto en un entorno de implementación determinado. Por tanto, la semántica de la noción de subsistema de implementación se refinará ligeramente cuando se manifieste en un entorno de implementación determinado.

El diagrama de subsistema de implementación correspondiente al Módulo de Acciones Especiales es el siguiente.

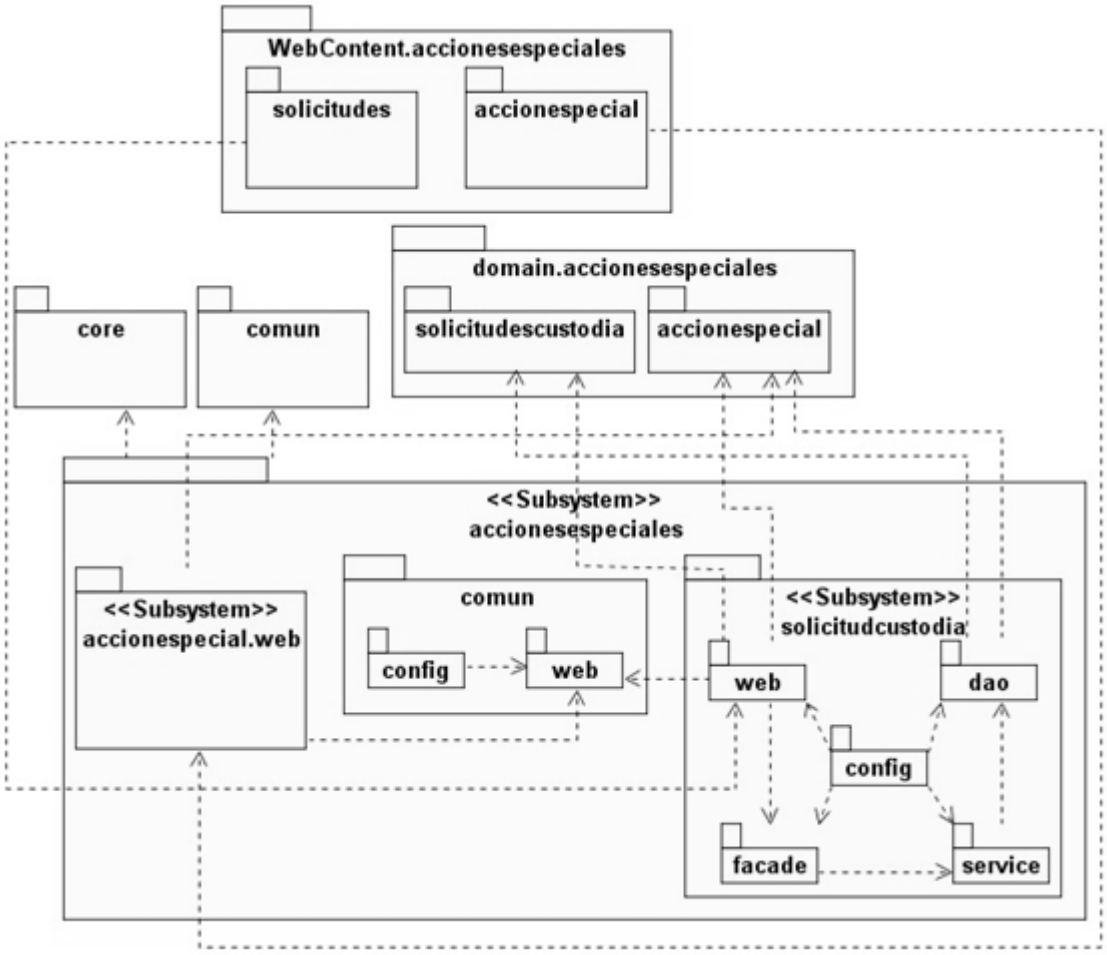


Figura 19. Diagrama de subsistemas de implementación

3.2.2. DIAGRAMAS DE COMPONENTES

"Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño" (Jacobson, y otros, 2000). De ahí que los componentes tengan relaciones de trazas con los elementos del modelo que implementan. (Jacobson, y otros, 2000)

Los diagramas de componentes muestran la organización y dependencias entre un conjunto de componentes (Ferré Grau, y otros, 2004).

Los diagramas de componentes correspondientes al Módulo de Acciones Especiales son los siguientes.

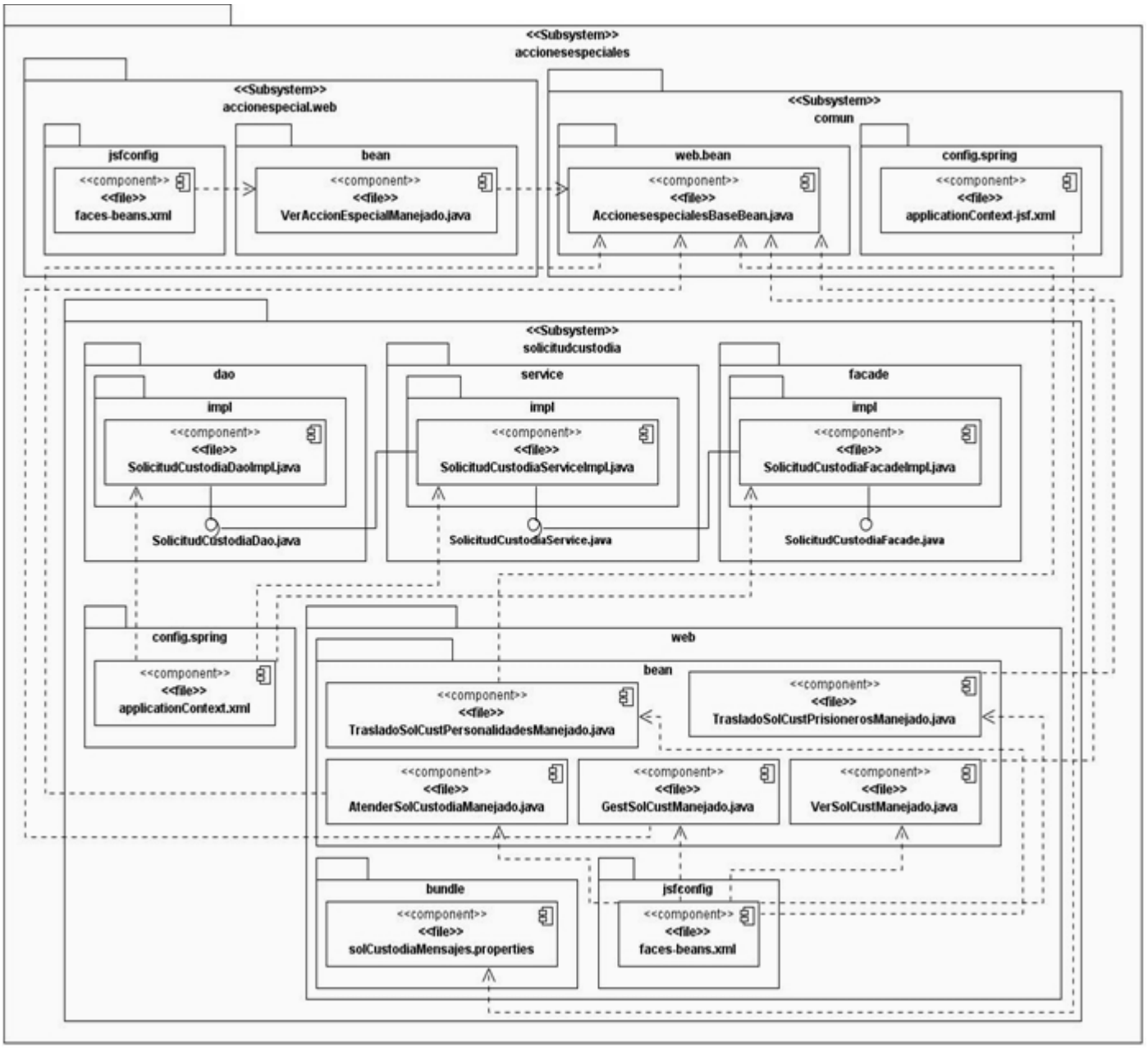


Figura 20. Diagrama de componentes del subsistema Acciones Especiales

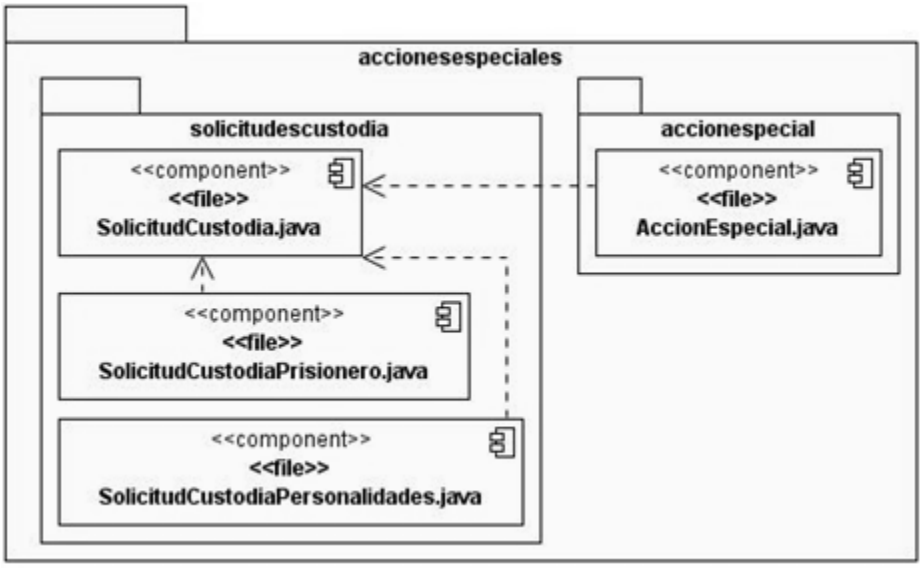


Figura 21. Diagrama de componentes del Dominio de Acciones Especiales

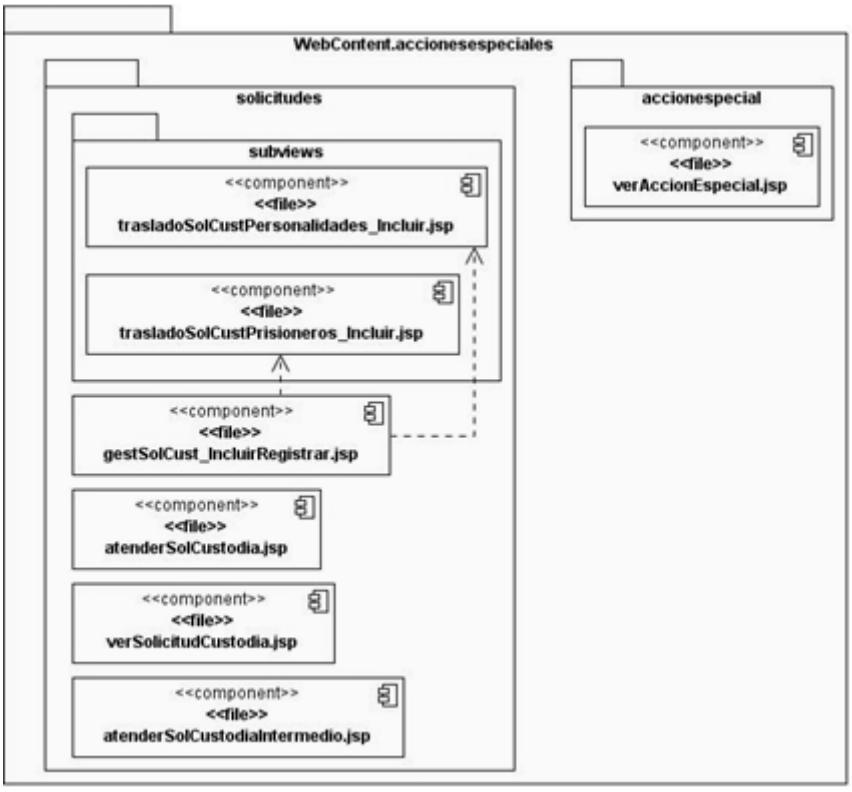


Figura 22. Diagrama de componentes del WebContent de Acciones Especiales

3.3. VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

Una vez generado el código fuente, el software debe ser probado para revelar y corregir el máximo de errores posible antes de su entrega al cliente, haciéndolo con la menor cantidad de tiempo y esfuerzo posible (Pressman, 1997). “Las pruebas del software son un elemento para la garantía de la calidad del software y representan una revisión final de las especificaciones, del diseño y de la codificación” (Pressman, 1997).

Como objetivo fundamental de la prueba según Pressman en la quinta edición del libro “Ingeniería de Software un enfoque práctico” se tiene el siguiente: descubrir la máxima cantidad de errores no detectados hasta entonces. Aun así, es necesario señalar que las pruebas no pueden asegurar la ausencia de errores, solo puede demostrar que existen defectos en el software. (Pressman, 1997)

En la actualidad es decisivo verificar y evaluar la calidad de lo implementado, para minimizar el costo de su reparación y la aceptación por parte del cliente final.

El desarrollo de las pruebas se basó en dos métodos de pruebas (Caja Blanca y Caja Negra), además fue enfocado a solo 3 niveles, los cuales, son los fundamentales y aportan resultados significativos para la validación del módulo.

3.3.1. MÉTODOS DE PRUEBAS

Desde años anteriores, han surgido y desarrollado una variedad de métodos para efectuar pruebas de software. Los más significativos en este contexto son los de prueba de caja blanca y caja negra; las primeras, pruebas orientadas a la estructura y las segundas al comportamiento del software. (Velasco Elizondo, 2001)

En la actualidad, la mayor parte de las técnicas de prueba se basan fundamentalmente en estos dos métodos, los cuales han tomado un lugar muy importante en el desarrollo de sistemas de cualquier tipo, sin dichas pruebas un sistema desarrollado carecería de garantías y credibilidad en sus resultados.

Pruebas de caja blanca: la prueba de caja blanca, denominada a veces prueba de caja de cristal, es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que: garanticen que se ejercita por lo menos una vez todos los caminos

independientes de cada módulo; ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa; ejecuten todos los bucles en sus límites y con sus límites operacionales y ejerciten las estructuras internas de datos para asegurar su validez. (Pressman, 1997)

Pruebas de caja negra: Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales, permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Este tipo de prueba intenta descubrir diferentes errores que los métodos de caja blanca no logran identificar. Las categorías de los errores encontrados por este tipo de prueba son: funciones incorrectas o ausentes, errores de interfaz, estructuras de datos, rendimiento, inicialización y terminación. (Pressman, 1997)

3.3.2. NIVEL DE PRUEBAS DE UNIDAD

Es el nivel donde se prueba el componente de software mínimo, o el módulo. Cada unidad (componente básico) del software se prueba para verificar que el diseño detallado para la unidad se ha puesto en ejecución correctamente. En un ambiente orientado al objeto, esto está generalmente en el nivel de la clase. (Carpeta, 1999)

Las pruebas de unidad tienen como objetivo fundamental verificar la funcionalidad y estructura de cada componente individualmente una vez que ha sido codificado. La prueba de unidad hace un uso intenso de los métodos de prueba de caja blanca, debido a que se analiza la lógica del módulo usando uno o más de estos. (José Barrios, y otros, 2007)

Estas pruebas fueron efectuadas por el programador. Después de realizada la implementación de cada funcionalidad, esta era probada de forma individual. La mejor forma de ilustrar la aplicación de estas pruebas, es a través de la clase JUnit[11] utilizada.

A continuación se muestra la clase JUnit del Módulo Acciones Especiales.

Nombre: SolicitudCustodiaFacadeJUnit	
Tipo de clase: JUnit	
Para cada atributo:	
Nombre	*Tipo*
solicitudCustodiaFacade	SolicitudCustodiaFacade
Para cada responsabilidad:	
Nombre	*Descripción*
testSalvarSolicitudCustodia	Método para probar la funcionalidad que permite salvar una solicitud de custodia.
testCargarSolicitudCustodiaPorId	Método para probar la funcionalidad que permite cargar una solicitud de custodia dado el id de esta.
testActualizarSolicitudCustodia	Método para probar la funcionalidad que permite actualizar una solicitud de custodia.
testSalvarAccionEspecial	Método para probar la funcionalidad que permite salvar una Acción especial.
testActualizarAccionEspecial	Método para probar la funcionalidad que permite actualizar una Acción especial.

Con estas pruebas se logró un correcto funcionamiento y un aumento en la calidad de lo implementado, logrando un código sin errores, listo para integrarse con el sistema y ser probado como un todo.

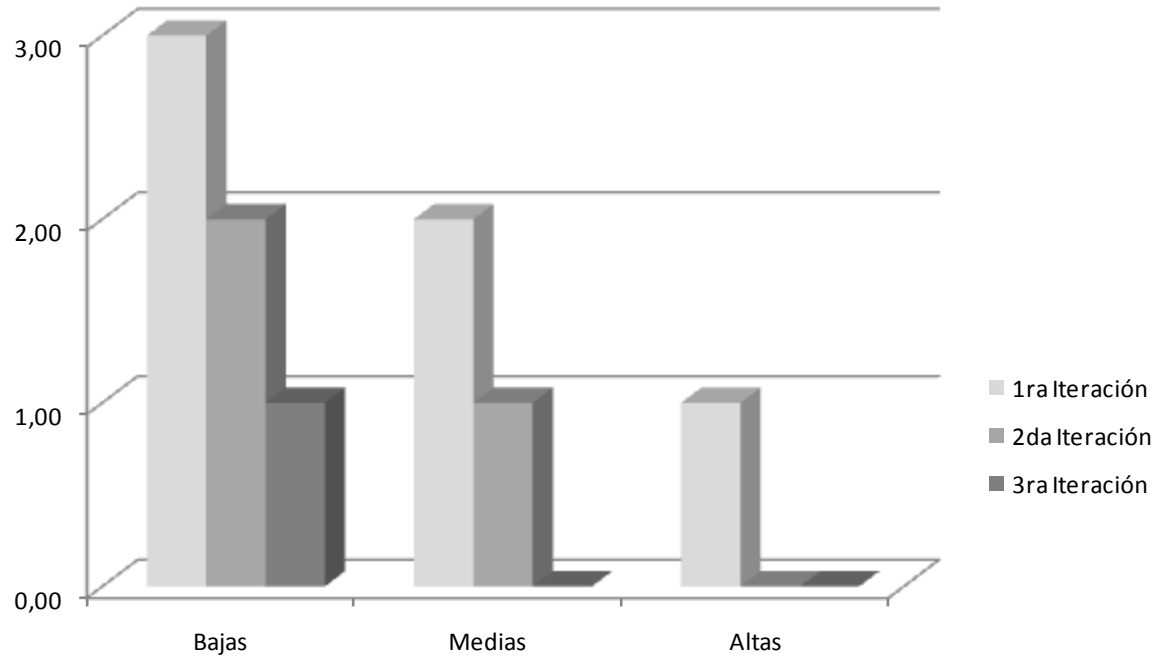
3.3.3. NIVEL DE PRUEBAS DE SISTEMA

La prueba de sistema es la realizada sobre un software completamente integrado, que proporciona una seguridad final de que el software satisface todos los requisitos especificados. Durante la validación se usan exclusivamente técnicas de prueba de caja negra. (IEEE, 1990)

Estas pruebas fueron realizadas por Calidad UCI, entidad externa al equipo de trabajo de SIIPOL y se encarga de certificar la calidad del software terminado. Para la realización de las pruebas se

confeccionaron casos de pruebas, los cuales no son más que: “un conjunto de entradas, condiciones de ejecución y resultados esperados diseñados para un objetivo particular” (IEEE, 1990). Los casos de pruebas fueron diseñados previamente por el equipo de calidad y utilizados fundamentalmente con el objetivo de determinar si los requisitos especificados fueron cumplidos.

Estas pruebas fueron realizadas en 3 iteraciones, los resultados obtenidos se ilustran en la siguiente gráfica:



Iteración	Total	Bajas	Medias	Altas
Pruebas de Liberación 3ra Etapa 1ra Iteración	6	3	2	1
Pruebas de Liberación 3ra Etapa 2da Iteración	3	2	1	0
Pruebas de Liberación 3ra Etapa 3ra Iteración	1	1	0	0

Haciendo un análisis del gráfico, se observa como la cantidad de no conformidades disminuye a medida que se avanza a una iteración siguiente, demostrando la corrección de los errores y la calidad de lo desarrollado.

3.3.4. NIVEL DE PRUEBAS DE ACEPTACIÓN

Las pruebas de aceptación validan que un sistema cumple con el funcionamiento esperado, además permite al usuario de dicho sistema que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento. (José Barrios, y otros, 2007)

"Las pruebas de aceptación son definidas por el usuario del sistema y preparadas por el equipo de desarrollo, aunque la ejecución y aprobación final corresponden al usuario". (José Barrios, y otros, 2007)

Durante la validación del sistema se utilizan métodos de pruebas de caja negra, los cuales demuestran la conformidad con los requisitos funcionales especificados por el usuario, teniendo en cuenta también, los requisitos no funcionales relacionados con el rendimiento, seguridad de acceso al sistema, a los datos y procesos, así como a los distintos recursos del sistema. (José Barrios, y otros, 2007)

En este nivel, el cliente interactuó con el módulo, aparentando un ambiente real. Como resultado, no se detectaron no conformidades y se obtuvo una petición de cambio.

3.4. CONCLUSIONES

Se puede concluir en que, los diagramas generados en el modelo de implementación, visualizan los diferentes subsistemas de implementación de los que se compone el módulo, además muestran cada uno de los diferentes componentes que lo conforman y sus relaciones, logrando un mejor entendimiento del sistema desde el punto de vista de implementación. Además, los resultados obtenidos en las pruebas realizadas, ejemplifican la calidad del trabajo efectuado en la implementación del Módulo Acciones Especiales, también muestran pleno cumplimiento de los requisitos funcionales y no funcionales.

CONCLUSIONES GENERALES

Con el desarrollo del presente trabajo se obtuvieron conocimientos teóricos sobre los Sistemas de Gestión de Información e Información Policial, conceptos fundamentales para la interpretación de la propuesta solución. Se describieron las herramientas, lenguajes y metodología empleados en la solución, los cuales proporcionan un desarrollo cómodo, debido a que poseen características que agilizan y facilitan el proceso de desarrollo de software, con el uso de estos elementos se alcanzará como resultado, un módulo que solucionará la problemática planteada. Además, se le dio cumplimiento a los requisitos funcionales y no funcionales. También se demostró que, el análisis y diseño constituyen un paso fundamental para un desarrollo correcto de la implementación, logrando así, que el sistema tenga la menor cantidad de errores y de esta forma alcanzar una alta calidad en el módulo de gestión de información obtenido.

RECOMENDACIONES

Continuar con la realización de las Pruebas de Calidad al software, para entregar una aplicación más robusta y estable.

Realizar una refactorización en todas las capas de la aplicación para lograr una mayor optimización en el rendimiento del sistema.

Realizar un estudio para determinar si pueden ser adicionadas nuevas funcionalidades al Módulo Acciones Especiales.

BIBLIOGRAFÍA

2009. Adapting. *Adapting.* [En línea] 10 de 3 de 2009. http://www.gestiondocumental.biz/AdaptingNews/usuario/muestranoticia.asp?IdProducto=&IdCanalEVC=&IdCategoria=0&IdGrupo=1&IdNoticia=20.

ALBET, S.A. 2009. *Proyecto de Modernización del CICPC. ARTEFACTOS A GENERAR. Estilos de Modelado.* La Habana : s.n., 2009.

—. 2007. *Proyecto de Modernización del CICPC. Documento de descripción de la Arquitectura de Software.* Habana : ALBET, S.A, 2007.

—. 2007. *Proyecto de Modernización del CICPC. ESPECIFICACIONES DE CASOS DE USO.* 2007.

—. 2007. *Proyecto de Modernización del CICPC. Requerimientos suplementarios.* 2007.

Albiol, Francesc Rosés. 2003. *Introducción a Hibernate.* 2003.

ALEGSA. 2010. ALEGSA.COM. *Diccionario de informática.* [En línea] 2010. [Citado el: 8 de enero de 2010.] http://www.alegsa.com.ar/Dic/interfaz.php.

—. 2010. ALEGSA.COM. *Diccionario de informática.* [En línea] 2010. [Citado el: 2010 de abril de 18.] http://www.alegsa.com.ar/Dic/bytecode.php.

Área de Diseño Gráfico y Multimedia - FIA. 2002. NOTIFIA. *http://www.usmp.edu.pe/.* [En línea] 2002. [Citado el: 7 de marzo de 2010.] http://www.usmp.edu.pe/publicaciones/boletin/fia/info40/j2ee.html.

Asteasuain, Fernando y Ezequiel Contreras, Bernardo. 2002. *PROGRAMACIÓN ORIENTADA A ASPECTOS.* 2002.

Autentia. 2003. Adictos al Trabajo. Formación y desarrollo. *adictosaltrabajo.com.* [En línea] 2003. [Citado el: 22 de marzo de 2010.] http://www.adictosaltrabajo.com/tutoriales/richFacesJsfIntro.php.

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

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

Carpeta, Roberto V. 1999. *Sistemas orientados al objeto de prueba: Objetos, patrones, y herramientas.* s.l. : Addison-Wesley, 1999. 0-201-80938-9.

Ciberaula. 2010. Ciberaula. *Curso de Java para Internet (J2EE).* [En línea] 2010. [Citado el: 1 de abril de 2010.] http://www.ciberaula.com/curso/java2/que_es/.

Collado Cabeza, Eduardo y Díaz Berenguer, Angi. 2003. Madritel. *http://web.madritel.es/personales3/edcollado/index.html.* [En línea] 2003. [Citado el: 19 de febrero de 2010.] http://web.madritel.es/personales3/edcollado/ingsw/tema2/2-6.htm.

Color Vivo Internet. 1999. Java en Castellano. *Java en Castellano.* [En línea] 1999. [Citado el: 3 de febrero de 2010.] http://www.programacion.com/java/tutorial/jsf_intro.

Díaz 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 Foundation. 2010. Eclipse. *Eclipse.* [En línea] 2010. [Citado el: 8 de febrero de 2010.] http://www.eclipse.org/.

Ferré Grau, Xavier y Sánchez Segura, María Isabel. 2004. Clikear.com. *Desarrollo Orientado a Objetos con UML.* [En línea] 2004. [Citado el: 8 de abril de 2010.] http://www.clikear.com/manuales/uml/index.aspx.

Figueroa, Roberth G. y Solís, Camilo J. 2007. *Metodologías Tradicionales Vs. Metodologías Ágiles.* 2007.

Fowler, Martin y Scott, Kendall. 1999. *UML gota a gota.* México : Prentice Hall, 1999. ISBN: 968-44-364-1.

Gamma, Erich, y otros. 1994. *Design Patterns Elements of Reuseable Object-Oriented Software.* s.l. : Addison Wesley, 1994.

Grand, Mark. 2002. *Patterns in Java, Volume 1.* s.l. : Wiley Publishing Company, 2002. 0-471-22729-3.

Gutiérrez, Javier. 2009. *Una comparación entre C, C++, Java y Ada.* Cantabria : s.n., 2009.

Ian Sommerville, Pete Sawyer. John Wiley & Sons. 1997. *Requirement Engineering: a good practice guide.* 1997.

IEEE. 1990. *Computer Dictionary. Compilation de IEEE Standard Computer Glossaries.* 1990. STD-610.

—. 1990. *Diccionario estándar de la computadora de IEEE: Una compilación de los glosarios estándares de la computadora de IEEE.* Nueva York : s.n., 1990. 1559370793.

Infomed. 1998. Biblioteca Virtual de Ciencia de la Información. *http://www.bvscuba.sld.cu/php/index.php.* [En línea] 1998. [Citado el: 14 de febrero de 2010.] http://cis.sld.cu/E/monografias/gestioncap1.html.

Jacobson, Ivar, Rumbaugh, James y Booch, Grady. 2007. *El lenguaje Unificado de Modelado. Manual de Referencia.* Madrid : Addison Wesley, 2007. Segunda Edición.

—. 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid : Pearson Educacion, 2000. 120.

Johnson, Rod. 2005. Theserverside.com. *Theserverside.com.* [En línea] 1 de mayo de 2005. http://www.theserverside.com/news/1364527/Introduction-to-the-Spring-Framework.

Joomla Venezuela. 2010. Joomla Venezuela. *Glosario Joomla Venezuela.* [En línea] 2010. [Citado el: 3 de marzo de 2010.] http://www.joomlavenezuela.org/web/glosario.html.

José Barrios, Emilio y Rojas Rojas, Johanna. 2007. INVESTIGACIÓN SOBRE ESTADO DEL ARTE EN DISEÑO Y APLICACIÓN DE PRUEBAS DE SOFTWARE. [En línea] 2007. [Citado el: 26 de abril de 2010.] http://www.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/Pruebasdesoftware.html.

Lago Bagüés, Ramiro. 2005. http://www.proactiva-calidad.com. *http://www.proactiva-calidad.com.* [En línea] 2005. [Citado el: 8 de enero de 2010.] http://www.proactiva-calidad.com/java/ejb/introduccion.html.

Larman, Craig. 2003. *UML y Patrones, Segunda Edición.* s.l. : PEARSON, 2003.

—. 1999. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : Prentice Hall, 1999. 970-17-0261-1.

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

Lopez, Angel. 2007. ajlopez. *http://www.ajlopez.net.* [En línea] 2007. [Citado el: 5 de 3 de 2010.] http://www.ajlopez.net.

Lou Torrijos, Ricard. 2003. Programación en Castellano. *Introducción a la Tecnología JavaServer Faces.* [En línea] 2003. [Citado el: 18 de enero de 2010.] http://www.programacion.com/articulo/introduccion_a_la_tecnologia_javaserver_faces_233.

Masadelante.com. 1999. Masadelante.com. *Masadelante.com.* [En línea] 1999. [Citado el: 2010 de abril de 25.] http://www.masadelante.com/faqs/plug-in.

Medín Piñeiro, Juan y García Figueras, Antonio. 2006. *Hacia una arquitectura con JavaServer Faces, Spring, Hibernate y otros frameworks.* Sevilla : s.n., 2006.

MKM Publicaciones Informáticas. 2007. MKM. *http://www.mkm-pi.com.* [En línea] MKM Publicaciones Informáticas, 7 de 1 de 2007. [Citado el: 24 de febrero de 2010.] http://www.mkm-pi.com/mkmpi.php?article2154.

Oracle Corporation. 2002. Core J2EE Patterns - Data Access Object. *Core J2EE Patterns - Data Access Object.* [En línea] 2002. [Citado el: 1 de marzo de 2010.] http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html.

Palos, Juan Antoni. 2006. *Servlets y JSP.* 2006.

Pressman, Roger S. 1997. *Ingeniería de Software un enfoque práctico. Quinta Edición.* 1997.

Rational Software Corporation. 2003. *Rational Unified Process.* 2003. Version 2003.06.00.65.

Rivero Amador, Msc. Soleydi y de Lyz Contreras Díaz, Lic. Yimian. 2007. www.gestiopolis.com. *Diseños de flujos de información para el posterior perfeccionamiento del sistema de gestión de información y conocimiento del centro de estudios de medio ambiente y recursos naturales (CEMARNA) de la universidad de Pinar del Río.* [En línea] 11 de 12 de 2007. [Citado el: 8 de enero de 2010.] http://www.gestiopolis.com/administracion-estrategia/flujo-de-informacion-en-el-cuidado-del-medio-ambiente.htm.

Sánchez Ramón, Óscar. 2006. *Java Server Faces.* 2006.

Sommerville, Ian. 2005. *Ingienería de Software - 7ma Edición.* Madrid : Addison Wesley, 2005. ISBN-84-7829-074-5.

SpringSource. 2010. Community Spring Source. *www.springframework.org/documentation.* [En línea] 2010. [Citado el: 19 de febrero de 2010.] http://www.springsource.org/documentation.

Suárez González, Héctor. 2003. *Manual Hibernate.* s.l. : JavaHispano, 2003. 22.04.2003.

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: 18 de febrero de 2010.] http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%20de%20Desarrollo%20Integrado.pdf.

Velasco Elizondo, Perla Inés. 2001. *PRUEBA DE COMPONENTES DE SOFTWARE BASADAS EN EL MODELO DE JAVABEANS.* Tlaxcala, México : s.n., 2001.

Villán García, Berman. 1998. *Implementación de una Intranet para el tratamiento de la información interna en una organización.* 1998.

Visual Paradigm Organización. 2009. Sitio Web oficial Visual-Paradigm. *Sitio Web oficial Visual-Paradigm.* [En línea] 2009. [Citado el: 22 de febrero de 2010.] http://www.visual-paradigm.com/product/vpuml/.

Walls, Craig y Breidenbach, Ryan. 2005. *Spring in Action.* s.l. : Manning, 2005.

Wayne Bartle, Philip Francis. 2009. Potenciación comunitaria. *www.scn.org.* [En línea] 2009. [Citado el: 18 de febrero de 2010.] http://www.scn.org/mpfc/modules/mon-miss.htm.

Yang Shen, Derek. 2010. Integración de JSF, Spring e Hibernate para crear una Aplicación Web del Mundo Real. *Programación en castellano.* [En línea] 2010. [Citado el: 5 de febrero de 2010.] http://www.programacion.net/tutorial/jap_jsfwork/3/.

GLOSARIO DE TÉRMINOS

[1] **Bytecode** es el código intermedio entre el código fuente y el código máquina. Suele tratársele como un fichero binario que contiene un programa ejecutable similar a un módulo objeto. (ALEGSA, 2010)

[2] **IBM** o International Business Machines es una empresa multinacional que fabrica y comercializa herramientas, programas y servicios relacionados con la informática.

[3] Un **plugin** es un módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande. (Masadelante.com, 1999)

[4] Java Server Pages (**JSP**) es una tecnología Java, que permite combinar HTML estático con HTML dinámico, también se puede generar contenido dinámico en otros formatos como, XML y otros. (Palos, 2006)

[5] **UI** es parte de un programa que permite el flujo de información entre un usuario y la aplicación, o entre la aplicación y otros programas o periféricos. Esa parte de un programa está constituida por un conjunto de comandos y métodos que permiten estas intercomunicaciones (ALEGSA, 2010).

[6] **Ajax**, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas.

[7] La Programación Orientada a Aspectos (**AOP**, por las siglas de Aspect-Oriented Programming) es una nueva metodología de programación que aspira a soportar la separación de las propiedades para los aspectos. (Asteasuain, y otros, 2002)

[8] Un **EJB** (Enterprise JavaBeans) es un conjunto de clases y un archivo XML que describe el despliegue de dichas clases en un contenedor. Un EJB se ejecuta en un contenedor y ofrece al desarrollador del bean servicios que no necesita programar (persistencia, seguridad, etc.) (Lago Bagüés, 2005).

[9] Un **POJO** (acrónimo de Plain Old Java Object) es una sigla creada por Martin Fowler, Rebecca Parsons y Josh MacKenzie, utilizada por programadores Java para enfatizar el uso de clases simples y que no dependen de un framework en especial.

[10] Una **API** (del inglés Application Programming Interface - Interfaz de Programación de Aplicaciones) es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizados por otro software como una capa de abstracción. (Joomla Venezuela, 2010)

[11] **JUnit** es un conjunto de clases opensource que permite probar aplicaciones Java.