Universidad de las Ciencias Informáticas

Facultad 8

Análisis, Diseño e Implementación del Submódulo Servicios del Módulo de Investigación en Ciencias Forenses del SIIPOL.

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

AUTORES: Leonel Enríquez Guisado

Alejandro Medina Mejías

TUTORES: Ing. Yordankis Matos López

Ing. Rosalba Carralero Medina

Ciudad de la Habana, 22 de febrero del 2010.

"Año 52 de la Revolución"

Dedicatoria

De Leonel:

A mis padres, por su amor incondicional, por su apoyo y preocupación constante hacia mí y mi hermana, por ser guías en mi formación como persona.

A mi hermana, por escucharme siempre y estar siempre ahí para conversar.

A mi familia, por su apoyo y preocupación constante.

A la revolución, por darnos la oportunidad de formarnos como profesionales.

De Alejandro:

A mi abuelo, por el recuerdo.

A mis queridas abuelitas, por el amor inigualable.

A mis padres, por ser la fuente que da sustento a mi vida, por su amor incondicional, por la eterna confianza.

A mis hermanas Katy y Yudelkis y mis hermanos Nordelis, Yoelvis y Yorbelis, para que sigan el ejemplo.

Al comandante Fidel, por la oportunidad, por la fabulosa idea de crear la Universidad que me formó como profesional.

Agradecimientos

De Leonel:

… A mis padres por todo el amor y la confianza que me han tenido todos estos años, que son los mejores padres del mundo y siempre les estaré agradecido por darme una vida llena de felicidad juntos a ustedes. A mi hermana por su apoyo y amor incondicional sabes que siempre estaré junto a ti. A mi familia que siempre me apoyó. A Yisell por su cariño y amistad sin límites. A mis amigos y amigas por su confianza.

... A mis compañeros y compañeras de aula los viejos y los nuevos, por haberme permitido pasar estos maravillosos cinco años a su lado.

... A los tutores Rosalba, Yordankis y Lázaro que siempre estuvieron ahí para apoyarnos y criticarnos, que más que tutores fueron amigos. Al pana por ser el mejor compañero de tesis.

… A todos los que dieron o quitaron un granito de arena para obtener este triunfo.

De Alejandro:

… Agradezco especialmente a mis padres, papi, eres mi ejemplo, mami, eres mi vida. A mis abuelos, cuyo cariño es inigualable y me ha permitido situar metas cada vez mayores. A mis hermanos, Katy, Mejer, Norde, Niño y Negrito, por la confianza, por el aliento, por ver a diario en mí, lo que consideran su ejemplo, a mi sobrinita Naty, por ser la luz inspiradora, a mis primos a los cuales considero hermanos, en especial a Eduardo, Martica y Marylín por el apoyo, la hospitalidad y la acogida, a toda la familia en general. A los amigos del barrio, Davisito, Daniel, Pipi, Marlon, Pancholo, Chichi y los demás (imposible mencionarlos a todos).

… Hay personas que transitan desapercibidas por la vida, pero un día llegan a ti y lo transforman todo, por eso le agradezco a Arle por su amistad infinita, por servirme de refugio cuando no encontraba salidas.

… A mis compañeros de estudio, en especial a Miguel, Andy, Raciel, Terry, Ale Rey, La Suse, Luisinho, Daley, Tabel, Neima, y Yordanis, por su apoyo. A Yaneisis, El Crack Adrián, a Leo y al Tropa Michel, por haberse comportado como mis hermanos, por comprenderme y apoyarme siempre.

… A mis tutores, Yordan, Rosy, Vazquez y Lázaro, por la guía acertada y sus sabios consejos, por ser, más que tutores, incondicionales amigos.

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____ _____

Leonel Enríquez Guisado Alejandro Medina Mejías

Resumen

El Cuerpo de Investigaciones Científicas, Penales y Criminalísticas (CICPC) es el organismo encargado del enfrentamiento al delito en la República Bolivariana de Venezuela. La Coordinación Nacional de Ciencias Forenses (CNCF) es una institución adscrita al CICPC cuyo objetivo principal es administrar, organizar y supervisar todas las actividades relacionadas con las investigaciones forenses en todo el territorio nacional. El CICPC cuenta con un Sistema Integrado de Información Policial el cual se encarga de gestionar toda la información de interés para la organización. Este sistema no brinda funcionalidades para el manejo o consulta de la información que se genera en las áreas de la CNCF y está desarrollado sobre una tecnología actualmente obsoleta. En el marco de las relaciones entre Cuba y Venezuela por la colaboración de los países del ALBA, se ha concebido el proyecto de Modernización del CICPC, el cual se centra en la construcción de un nuevo Sistema de Investigación e Información Policial, (en lo adelante SIIPOL), que sustituya las prestaciones del sistema actual, sin obviar la CNCF, para la cual se construyó el Módulo de Investigación en Ciencias Forenses, que abarca grandes procesos, los cuales necesitan de la prestación de varios servicios. Este trabajo contiene la investigación y desarrollo de una aplicación web, que se integrará al sistema SIIPOL y que informatizará los procesos de servicios de la CNCF, para lograr una mejor organización con respecto a los medios y recursos que son utilizados en los estudios forenses, además del control administrativo de las actividades realizadas por el equipo técnico. Como solución al problema planteado se desarrolló una aplicación web con tecnología Java, haciéndose uso de algunos frameworks como son: Java Server Faces, Spring, Hibernate, entre otros, usando como guía la metodología de desarrollo de software Rational Unified Process (RUP), se realiza el análisis, diseño e implementación, obteniéndose una aplicación que cumple con los requisitos identificados durante el proceso de Ingeniería de Requerimientos.

Índice

Índice de Figuras

Introducción

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

Uno de los organismos encargados del enfrentamiento al delito en la hermana república es el Cuerpo de Investigaciones Científicas, Penales y Criminalísticas, (en lo adelante CICPC), el cual agrupa investigaciones de tipo criminalísticas y científicas, sobre el basamento de las leyes vigentes y los resultados son puestos a disposición de los órganos judiciales del Ministerio Público.

La Coordinación Nacional de Ciencias Forenses, (en lo adelante CNCF), es una institución adscripta al CICPC; tiene como función primordial servir de órgano auxiliar de justicia y se encarga de todo lo relativo a la política, organización, administración y supervisión de las actividades forenses en el territorio nacional.

Su **misión** es garantizar los servicios públicos forenses en la nación de acuerdo con las necesidades actuales de la colectividad, efectuando todos los procedimientos técnicos científicos de la Medicina Forense y otras ciencias, con la finalidad de auxiliar a la administración de justicia a través de un trabajo pericial óptimo.

Su **visión** consiste en ser la institución nacional rectora acreditada científica y académicamente en las Ciencias Forense, que garantice el aporte óptimo de peritaje legal, objetivo y fidedigno como medios de prueba en la administración de justicia.

La actual situación por la que atraviesa la Coordinación Nacional de Ciencias Forenses limita el aporte de la misma al desarrollo del proceso de investigación. Las instituciones o cuerpos policiales prestan poca atención a las cuestiones forenses y no utilizan en su totalidad los servicios de esta coordinación, muchas veces por el desconocimiento de estas mismas instituciones. Los expertos no tienen acceso a documentos generados en el comienzo del proceso de investigación que les pudieran ser útiles para ofrecer un resultado más confiable y completo a la entidad solicitante. Los datos recogidos suelen perderse durante la investigación, los cuales pudieran aportar información de interés a los expertos en el momento de emitir una valoración.

La información generada en las distintas unidades de las medicaturas[1] distribuidas por todo el país no llega a las que necesitan datos de personas desaparecidas o cadáveres no identificados. La

[1] Áreas de trabajo de la CNCF.

desactualización de la información referente a un caso hace que muchas veces se tomen decisiones o se den respuestas equivocadas. Dentro de las mismas unidades es imposible acceder a la información de las demás áreas de manera ágil pues no existen sistemas que permitan esta comunicación ni que se comuniquen con sistemas de otras instituciones externas como la Oficina Nacional de Identificación y Extranjería (en lo adelante ONIDEX).

El CICPC cuenta en la actualidad con el Sistema Integrado de Información Policial que centraliza la información de interés para los cuerpos policiales, mediante el cual los funcionarios, cuerpos policiales estadales y municipales que tienen acceso a lo largo de todo el país, pueden consultar personas buscadas por la justicia, antecedentes penales, delictivos, así como vehículos robados; el mismo no brinda funcionalidades para el manejo o consulta de la información que se genera en las áreas de la CNCF y que es de suma importancia para el esclarecimiento de los delitos. Este sistema está desarrollado sobre una tecnología actualmente obsoleta que posibilita el intercambio de información entre las diferentes áreas y entidades del CICPC; pero no resuelve todas las necesidades de la institución.

Estas problemáticas traen como consecuencias que las investigaciones se tornen lentas. Las respuestas desactualizadas y en ocasiones hasta erróneas, conllevan a fallos en los resultados de los procesos investigativos, que provocan no encontrar a los culpables de un hecho delictivo o en el peor de los casos incriminar a inocentes. Las jornadas de trabajo se tornan agotadoras, por lo que el desempeño de los funcionarios y técnicos no se aprovecha en su totalidad. Se producen desvíos de recursos técnicos al no poder llevar un control de los mismos.

En el marco de las relaciones entre Cuba y Venezuela por la colaboración de los países del ALBA, se ha concebido el proyecto de Modernización del CICPC, cuyo proceso de desarrollo es asumido por la Universidad de las Ciencias Informáticas (UCI). Este proyecto se centra en la construcción de un nuevo Sistema de Investigación e Información Policial, (en lo adelante SIIPOL), que sustituya las prestaciones del sistema actual, mejore e incluya nuevas funcionalidades, contribuya a la disminución de los tiempos de respuesta de las investigaciones de cada área, sin obviar la CNCF. Para la cual se construyó el Módulo de Investigación en Ciencias Forenses, que abarca grandes procesos. Estos procesos durante necesitan de la prestación de algunos servicios para llevar a cabo sus funciones, pero estos servicios no están informatizados.

La informatización de los servicios que se brindan en la CNCF posibilitará una mejor organización con respecto a los medios y recursos que son utilizados en los estudios forenses, además del control administrativo de las actividades realizadas por el equipo técnico, así como una reducción del margen de errores a cometer durante el manejo de las informaciones, contribuyendo así al éxito en los procesos investigativos.

Por tanto, el **problema a resolver** es: ¿Cómo mejorar la gestión de información de los servicios que brinda la Coordinación Nacional de Ciencias Forenses?

El **objeto de estudio** en el cual se enmarca la investigación es el proceso de desarrollo de software orientado a los sistemas de gestión de información.

El **campo de acción** está centrado en el proceso de desarrollo de software orientado a la gestión de información policial asociados al Submódulo Servicios del Módulo de Investigación en Ciencias Forenses del SIIPOL.

El **objetivo general** es desarrollar el Submódulo Servicios del Módulo de Investigación en Ciencias Forenses del SIIPOL.

Como **objetivos específicos** se presenta:

- Desarrollar el marco teórico orientado a los sistemas de gestión de información y centrado en los sistemas de gestión de información policial.
- Realizar el análisis y diseño del Submódulo Servicios del Módulo de Investigación en Ciencias Forenses del SIIPOL.
- Implementar el Submódulo Servicios del Módulo de Investigación en Ciencias Forenses del SIIPOL que de respuesta a las funcionalidades asociadas al mismo.
- Evaluar los resultados obtenidos y análisis de las pruebas realizadas al Submódulo Servicios del Módulo de Investigación en Ciencias Forenses del SIIPOL.

Como **idea a defender** se plantea que si se desarrolla un submódulo que gestione la información de los servicios que brinda la CNCF, mejorarán los Procesos de Investigación Forense del CICPC.

Se espera que este trabajo mejore considerablemente los procesos de investigación forense del CICPC y agilice el trabajo de los expertos forenses. Por otra parte, el Módulo de Investigación en Ciencias Forenses del SIIPOL representa una porción importante de todo el sistema, por lo que la confección del mismo aportará grandes ingresos a la economía cubana.

Para darle cumplimiento a los objetivos se han trazado las siguientes **tareas de investigación**:

1. Análisis de los sistemas de gestión de información y sistemas de gestión de información policial.
2. Análisis de aplicaciones o soluciones similares, profundizando en el Sistema Integrado de Información Policial.

3. Descripción de la metodología, lenguajes de implementación y modelado y las herramientas definidas para el desarrollo de la aplicación.
4. Análisis de las pautas de arquitectura definidas para el desarrollo del sistema.
5. Análisis detallado de las especificaciones de los casos de uso del sistema para el Submódulo Servicios.
6. Confección de los diagramas de clases del análisis y diagramas de colaboración.
7. Confección de los diagramas de contrato entre paquetes y diagrama de clases del diseño con estereotipos web.
8. Confección de los diagramas de clases persistentes.
9. Confección de los diagramas de subsistemas de implementación y diagramas de componentes.
10. Codificación del Submódulo Servicios.
11. Integración continúa de la solución propuesta al sistema SIIPOL.
12. Análisis de los diferentes tipos de pruebas de calidad.
13. Análisis de los resultados de las pruebas de calidad realizadas a la aplicación.
14. Análisis comparativo de los procesos de servicios que brinda la CNCF y las funcionalidades implementadas.

Para el desarrollo de la investigación se utilizan métodos científicos teóricos como son el Analítico – Sintético y la Modelación. El primero está dado por el análisis de los documentos generados en el levantamiento y captura de requisitos, extrayendo y analizando los principales elementos relacionados con el objeto de estudio. La Modelación se manifiesta en la creación de los diferentes modelos y diagramas que representan la propuesta de solución, los cuales permiten visualizar el sistema que se pretende desarrollar desde diferentes puntos de vista.

El trabajo está dividido en tres capítulos. El primero, titulado Fundamentación Teórica, tiene como objetivo hacer un estudio del alcance de los sistemas de gestión de información y la situación en el mundo de los sistemas de gestión de información policial, realizando un análisis del entorno actual de la institución que se desea automatizar y los sistemas a los cuales se vincula. Se describen la metodología, los lenguajes y las herramientas a utilizar y se presenta la propuesta de solución y la arquitectura para el desarrollo de la misma.

En el capítulo dos, denominado Análisis, Diseño e Implementación de la propuesta de solución, se muestra la descripción de los casos de uso del Submódulo Servicios, así como los diagramas generados durante el proceso de desarrollo.

El tercer capítulo, nombrado Validación de la propuesta de solución, resume el conjunto de pruebas realizadas al Submódulo Servicios. Se analizan los resultados obtenidos en las pruebas, teniendo en cuenta el grado de cumplimiento de los requisitos funcionales y no funcionales asociados al Submódulo. Se realiza un análisis comparativo de los procesos de la CNCF y las funcionalidades implementadas que denoten el grado de mejora alcanzado.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo se realiza un estudio de los sistemas de gestión de información y de la situación actual de los sistemas de gestión de información policial, lo que funciona como antesala para el análisis del entorno de la institución que se desea automatizar y los sistemas con los que se relacionan. Se describen las tecnologías, la metodología, los lenguajes y las herramientas seleccionadas, las cuales tributan buenas prácticas para el proceso de desarrollo de software.

1.1 Sistemas de Gestión de Información

"Se entiende por gestión de la información como un proceso que incluye operaciones de extracción, manipulación, tratamiento, depuración, conservación, acceso y/o colaboración de la información adquirida por una organización a través de diferentes fuentes y que gestiona el acceso y los derechos de los usuarios sobre la misma." (Information Management, 2006)

Se establece, por lo tanto, como una disciplina transversal que aparece entrelazada en todas las diferentes capas o tejidos de una organización, en todos los conceptos de management (recursos humanos, marketing, finanzas, estrategia, operaciones,…) y les proporciona soporte.

"La gestión de la información requiere el trabajo de personas que comprendan tanto las tecnologías y la teoría tras los sistemas de gestión de la información como del modelo de negocio de la organización para que los sistemas se conviertan en medios al servicio de la estrategia de la organización y no un fin. Soportan desde la toma de decisiones hasta la realización de una simple carta a un cliente. Se establece, entonces, como un recurso básico para cualquier organización." (Information Management, 2006)

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 la 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 y formularios. 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.

La puesta en marcha de un Sistema de Gestión de Información puede proporcionar numerosos beneficios, tales como la optimización del tiempo que transcurre desde la búsqueda de documentos y la reducción drástica de los riesgos de pérdida del documento físico original. Se puede tener acceso concurrente a un documento. Mejora la atención a los clientes, tanto internos como externos, y al disponer de la información de forma centralizada y rápidamente accesible, es posible solucionar *online* cuestiones relacionadas con una consulta de un cliente; y por otra parte incrementa la satisfacción de los usuarios internos, por agilizar sus procesos de trabajo, y reduce los costes legales. Protege la documentación sensible o de alto valor de deterioro, que puede derivar elevados costes por incurrir en demandas y otros procesos legales. Aunque es necesario tener en cuenta que el éxito de los sistemas de Gestión de Información depende en una buena parte de las habilidades y la experiencia de la fuerza de trabajo y pueden llegar a ser difíciles de usar.

Ofrecen numerosas ventajas, que se pueden apreciar en la disponibilidad inmediata de la información, teniéndose un intercambio instantáneo de los resultados obtenidos, permitiendo una rapidez en la toma de decisiones, además de que la información se encuentra centralizada, evitando la réplica, lo que permite que varias personas puedan acceder a la misma información desde diferentes localidades, reduce el tiempo de respuesta de las peticiones de los clientes, aumentando así la satisfacción de los usuarios finales, debido a que los procesos se vuelven más eficientes.

No obstante, existen desventajas debido a que en ocasiones el personal para desarrollar u operar los sistemas de gestión de información no se encuentra altamente calificado ni capacitado, requisito necesario pues estos sistemas pueden ser complejos de usar ya que, en ocasiones, no cumplen con

los requerimientos solicitados por los clientes. Además, provocan reducciones de plantillas en las instituciones, repercutiendo en la sociedad al aumentar el desempleo.

1.1.1 Sistemas de Gestión de Información Policial

"Un Sistema de Gestión de Información Policial, (en lo adelante SGIP), de acuerdo con la exposición anterior, se define como un Sistema de Gestión de Información que no permite la divulgación de la información, pues su característica representativa es la confidencialidad. Los SGIP constituyen la piedra angular en el entorno cotidiano de las instituciones policiales que cuentan con este tipo de herramienta para gestionar los procesos que cubren las necesidades reales de la organización." (Rodríguez Ramos, 2009)

El elevado índice de criminalidad a nivel mundial y la propensión inminente de sociedades cada vez más violentas, ha obligado a los organismos policiales a optimizar las acciones de investigación criminal tendente a lograr el esclarecimiento de los hechos delictivos y dotar al capital humano de la organización de herramientas, mecanismos logísticos e infraestructura que garanticen el óptimo desempeño de sus funciones.

Estos sistemas informatizan los procesos de las instituciones policiales. El registro y control de los datos recogidos en los procesos investigativos centralizan la información de manera que se acceda a ellos de manera rápida y por las personas debidamente autorizadas. Permiten medir el comportamiento social a través de las estadísticas que genera, identificando localidades, áreas o sectores con mayor índice de delitos, posibilitando así encaminar la labor preventiva hacia los objetivos de mayor necesidad.

1.1.2 Antecedentes de Sistemas Policiales

Son muchos los países que en la actualidad llevan a cabo proyectos de modernización para los sistemas policiales, esto se debe al aumento desmedido de la violencia y a la necesidad de dar una respuesta inmediata ante el crimen. A continuación se describen algunos de estos sistemas, distribuidos por diferentes regiones del mundo.

ePatrol: Existe en España un sistema de información corporativa denominado ePatrol que conforma la unión de un sistema de movilidad y un sistema de gestión administrativa donde participan la administración local y los cuerpos de seguridad, y que utiliza como soporte de comunicación y gestión las Nuevas Tecnologías de la Comunicación y la Información (TIC). (eLABORO, ingeniería del software S.L., 2008)

Sistema de Control de Evidencia Documental: En el año 2001 la División de Proyectos Especiales de la empresa Management Systems Designers (MSD) emprendió el desarrollo y la puesta en práctica

de un sistema de colección de documentos de referencia forense para cada una de las cuatro agencias gubernamentales en Bogotá, Colombia. Mediante la firma de un contrato adjudicado por el Programa Internacional de Asistencia para el Entrenamiento en Investigaciones Criminales (ICITAP) dependiente del Ministerio de Justicia de los EE. UU. (policiales.net)

Sistema de Información Policial (SIP): La Secretaría de Seguridad Pública del Distrito Federal de México emprendió a partir del 2002 la Reforma de la Policía de la Ciudad con el objetivo principal de modernizar la operación policial a partir del uso inteligente de la información. Con esta premisa, la Dirección General de Estadística e Información Policial comenzó a desarrollar el Sistema de Información Policial el cual es una nueva forma de trabajar con base en los siguientes principios (Secretaria de Seguridad Pública del Distrito Federal):

- Uso sistemático de la información para diseñar estrategias y medir resultados.
- Seguimiento permanente del trabajo de cada unidad policial.

Sistema Territorial de Emergencias y Gestión Policial (STEGPOL): STEGEPOL es un Sistema de Información Geográfica con tecnología de punta sobre una Plataforma Nacional Común de Información aplicada al Sistema de Emergencias Nacionales y al Sistema Territorial de Gestión Policial que actualmente operan en Chile, el cual identifica de forma veraz y efectiva geográficamente dónde se están cometiendo o se han cometido actos delictivos a nivel territorial, apoyado con sistemas de información en línea desde el lugar de los hechos, que permiten la acción rápida y coordinada de los encargados de la seguridad ciudadana a nivel nacional. (Mapas Digitales S.A.)

Sistema de Gestión Penitenciaria (SIGEP): SIGEP es un proyecto que da respuesta a las necesidades de gestión, información y apoyo a la toma de decisiones de la Dirección General de Custodia y Rehabilitación del Recluso (DGCRR) en Venezuela. (Dirección Nacional de Servicios Penitenciarios de Venezuela, 2007)

Este sistema tiene como objetivos fundamentales aumentar la eficacia, profesionalismo y equidad en el sistema penitenciario venezolano, para lograr un incremento de la confianza que se le pueda tener, además de generar y diseminar información vital para el funcionamiento de los establecimientos penitenciarios. Esto permitirá obtener estadísticas confiables y actualizadas sobre la situación jurídica de los privados de libertad, condiciones de vida y salud, actividades de rehabilitación y reinserción, la situación operativa y la actividad administrativa. Posibilitará la comunicación en línea con tribunales, sistemas de identificación y antecedentes penales que complementan la información necesaria para la gestión de los procesos vinculados con los presos.

Resultado del análisis de los Sistemas de Gestión de Información Policial

Los diferentes sistemas de información policial estudiados son construidos a la medida del cliente y del problema a resolver, se encuentran sumamente ajustados y regidos por las leyes y regulaciones vigentes en cada país y debido al alto nivel de seguridad que requieren nunca se puede obtener su código fuente total ni parcialmente por lo cual no pueden ser adaptados.

Al término del análisis se concluye que estos sistemas constituyen una plataforma definitiva para agilizar, simplificar y automatizar el trabajo diario de la policía, por lo que debe disponerse para ello de los medios tecnológicos más avanzados del mercado para ejercer su trabajo con mayor eficacia. Necesitan de un sistema de notificaciones que sin llegar a divulgar masivamente la información la comunique a las partes interesadas y autorizadas. Debe implementarse un sistema de búsqueda por criterios que permita un preciso y rápido acceso a la información que se necesita.

1.1.3 Sistema Integrado de Información Policial

El Sistema Integrado de Información Policial es una aplicación informática que el CICPC utiliza para llevar a cabo la gestión de la información generada en los procesos investigativos que realiza. Está desarrollado sobre el lenguaje de programación Natural y como gestor de base de datos Adabas, actualmente obsoleto. Su acceso se obtiene a través de un emulador[2], o un Telnet[3]. El servidor de aplicaciones cuenta con tecnología de Sun ya que posee Sistema Operativo Solaris. Utilizan un programa llamado Natural Security que valida la gestión de los usuarios para la seguridad del mismo.

Toda la información que maneja el sistema es texto, siendo los tiempos de respuesta muy bajos, incluso desde los puntos de acceso más lejanos que trabajan haciendo peticiones directamente al centro de datos.

El nivel de seguridad del sistema es bastante estricto, se lleva una traza de todas las operaciones realizadas y se restringen algunas operaciones como la actualización y la eliminación de cualquier información que haya sido introducida.

Dicho sistema abarca sólo los procesos básicos que se realizan actualmente en la institución. Utiliza datos de otras instituciones como la base de datos de SAIME (Servicio Autónomo de Identificación, Migración y Extranjería) y la del Instituto Nacional de Transporte Terrestre (INTT) enviada en soportes digitales para actualizar la base de datos del sistema una vez por semana o una vez al mes.

---

[2] Emulador: Es un software que permite ejecutar programas de computadora en una plataforma *(arquitectura hardware o sistema operativo)* diferente de aquella para la cual fueron desarrollados originalmente.

[3] Telnet: Es una de las formas de conectarse a otra computadora por medio de llamada directa del ordenador terminal al ordenador host y ejecutar secuencias de comandos, al estilo DOS o UNIX.

1.2 Cuerpo de Investigaciones Científicas Penales y Criminalísticas (CICPC)

El Cuerpo de Investigaciones Científicas Penales y Criminalísticas (CICPC) es el organismo rector del enfrentamiento al delito en La República Bolivariana de Venezuela, integra las investigaciones de tipo científicas y criminalísticas sobre la base de las leyes vigentes y pone los resultados a disposición de los órganos judiciales del Ministerio Público, responsable legal de la investigación. El CICPC tiene como visión convertirse en la institución de obligada consulta a nivel nacional en el enfrentamiento al delito por sus capacidades científicas y su equipamiento técnico. (Roselló Nuñez, 2007)

1.2.1 Proceso de Solicitud de Servicio en la Coordinación Nacional de Ciencias Forenses (CNCF).

El proceso de Solicitud de Servicio brinda apoyo a otros procesos de mayor complejidad dentro de la investigación forense, como los procesos de realización de experticias a personas vivas y cadáveres.

Este proceso está compuesto por subprocesos como la toma de fotografías a cadáveres donde el funcionario responsable introduce los datos de la sesión de fotografía para que luego sea confirmada o rechazada por el funcionario solicitante. El registro de la preparación de las muestras tomadas a cadáveres y la realización de radiografías a personas vivas y cadáveres siguen el mismo patrón pero con la diferencia de los datos introducidos que pertenecen a cada una de las sesiones correspondientes.

Estos procesos se llevan a cabo cuando las dependencias relacionadas con la investigación tales como; División de Antropología, División de Odontología y División de Patología, solicitan servicios a los departamentos de fotografía, radiografía e histología. Una vez realizados los servicios, estos departamentos registran las solicitudes para tener constancia de que los mismos fueron cumplidos. Otro de los subprocesos es la planificación de citas. Se realiza en la Dirección de Evaluación y Diagnostico Mental Forense que consiste en planificar una cita de un experto con la persona asociada a una experticia de peritaje psiquiátrico forense, para que este último la atienda.

1.3 Metodología, Lenguajes y Herramientas de Desarrollo

La metodología de desarrollo de software es el marco usado para estructurar, planear y controlar el proceso de desarrollo de un proyecto, que en dependencia de la plataforma de desarrollo y el lenguaje de programación, utiliza herramientas que permiten obtener los diferentes artefactos y el producto final. A continuación se exponen la metodología, lenguajes y herramientas que fueron escogidas por la dirección del proyecto para desarrollar el sistema SIIPOL.

1.3.1 **Metodología de desarrollo. Rational Unified Process (RUP).**

El Proceso Unificado de Desarrollo Software (RUP) es un marco de desarrollo de software dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental. El RUP abarca áreas de trabajo prolongables que pueden ser adaptados a organizaciones o proyectos específicos y pretende implementar las mejores prácticas de la ingeniería de software para asegurar una producción de alta calidad, dentro de plazos específicos y presupuestos predecibles para satisfacer las necesidades de los usuarios finales. Para conseguir lo que se propone se basa fundamentalmente en el orden y la documentación. Divide el proceso de desarrollo en ciclos, donde se obtiene un producto al final de cada ciclo y se apoya en el Lenguaje de Modelado Unificado (UML). Cada período se divide en cuatro Fases: Inicio, Elaboración, Construcción, y Transición; cada fase concluye con un hito bien definido donde deben tomarse ciertas decisiones. (Jacabson, y otros, 2000)

Fases del proceso:

- Inicio: 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.

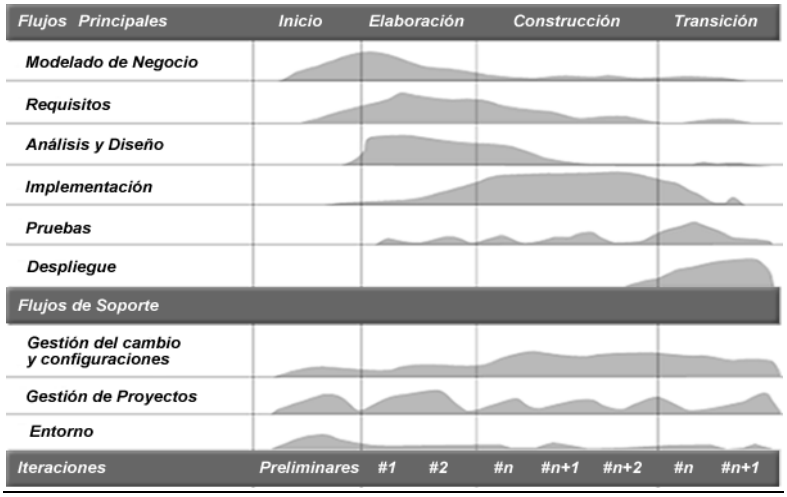

Figura 1. Fases y flujos de RUP.

Las principales bibliografías que abordan esta metodología coinciden en definir tres características fundamentales al hablar de RUP:

a. **Dirigido por casos de uso**: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo, ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso (cómo se llevan a cabo).
b. **Centrado en la arquitectura:** La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que 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. RUP se desarrolla mediante iteraciones, comenzando por los CU relevantes desde el punto de vista de la arquitectura. El modelo de arquitectura se representa a través de vistas (4+1), o sea, perspectivas del sistema, que logran una abstracción particular en cada uno de los casos, en otras palabras, se trata de que en cada una de las llamadas vistas se represente el sistema en su totalidad teniendo en cuenta solo determinados aspectos.
c. **Iterativo e incremental:** RUP propone que cada fase se desarrolle en iteraciones. 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.

RUP es más apropiado para proyectos grandes, dado que requiere un equipo de trabajo capaz de administrar un proceso complejo en varias etapas; en proyectos pequeños puede que no sea posible

cubrir los costos de dedicación del equipo de profesionales necesarios. Los requerimientos de los diversos inversores pueden ser diferentes, contradictorios o disputarse recursos limitados, así que debe encontrarse un balance que satisfaga los deseos de todos y el control de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción.

Análisis de la selección de la Metodología de Desarrollo.

A la hora de tomar la decisión de cuál metodología de desarrollo usar en la solución, se tuvieron en cuenta varios puntos definitorios como: la magnitud del proyecto, la cantidad de miembros del equipo, el tiempo de que se dispone para su culminación, lo distante que se encuentran desarrolladores y clientes geográficamente, las expectativas del cliente y la experiencia de los desarrolladores, entre otros.

La magnitud del software a producir hace que las partes a entregar deban quedar bien documentadas internamente; los formalismos se hacen necesarios para prevenir malentendidos que no son deseables en el marco en el que se desenvuelve la producción del software. La cantidad de miembros del equipo, que supera los 100, deriva en la decantación por un proceso que promueva el orden y el control, tanto de los recursos humanos como de los artefactos producidos. El tiempo estimado de producción, que supera los 16 meses, unido a los factores anteriormente mencionados y a la distancia que separa a Cuba de Venezuela, hacen que la opción más viable económicamente sea producir el software en Cuba, y no es conveniente tampoco mantener a un grupo de clientes en el equipo de desarrollo por las mismas razones; la consecuencia es que la comunicación personal se dificulta y la mejor opción es la formalidad.

Por otra parte, el cliente espera un producto que contenga manuales y documentación; este requisito es cumplido de manera más efectiva por RUP. La transferencia tecnológica, otro de los puntos que se deben asegurar en el proyecto, resulta posible gracias a la documentación que esta metodología genera. La experiencia de los desarrolladores es muy poca para enfrentar un proyecto de esta magnitud, sin pautas bien definidas y una referencia disponible en todo momento. Por todas estas razones se considera que la elección de RUP para desarrollar el SIIPOL es correcta.

1.3.2 Lenguaje de Modelado UML

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un plano del sistema, incluyendo aspectos conceptuales, tales como procesos de negocios, funciones del sistema y aspectos concretos como; expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. En fin UML es el

enlace entre quien tiene la idea y el desarrollador, definiendo la comunicación como objetivo principal. (Rumbaugh, et al.)

Es importante resaltar que UML es un lenguaje para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como RUP), pero no especifica en sí mismo qué metodología o proceso usar. El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que es un lenguaje, cuenta con reglas para combinar tales elementos. En este caso, se centra en la representación gráfica de un sistema e indica cómo crear y leer los modelos, pero no dice cómo crear los sistemas. Esto último es el objetivo de las metodologías de desarrollo. (Larman, 1999)

Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones:

- **Visualizar:** UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- **Especificar:** UML permite especificar cuáles son las características de un sistema antes de su construcción.
- **Construir:** A partir de los modelos especificados se pueden construir los sistemas diseñados.
- **Documentar:** Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

Durante este trabajo el uso UML ha permitido especificar los procesos en el desarrollo del software, modelar el sistema y determinar los diferentes artefactos, utilizando conceptos orientados a objetos, permitiendo una alta reutilización y minimizando los costos y el esfuerzo del personal. Su uso ha servido como vía de comunicación y documentación.

1.3.3 Plataforma de desarrollo

En términos informáticos una plataforma de desarrollo es el entorno de software común en el cual se desenvuelve la programación de un grupo definido de aplicaciones. Comúnmente se encuentra relacionada directamente a un sistema operativo; sin embargo, también es posible encontrarla ligada a una familia de lenguajes de programación o a una Interfaz de Programación de Aplicaciones (API). (ADMINISTRACIÓN PÚBLICA DEL ESTADO DE QUINTANA ROO, 2005)

El desarrollo de aplicaciones y servicios web ha sufrido un auge muy importante durante los últimos años. Frente a esta nueva demanda han surgido varias plataformas para el desarrollo de este tipo de aplicaciones como es el caso de J2EE de Sun Microsystems.

Java 2 Enterprise Edition (J2EE)

Debido a la necesidad del mercado de desarrollo de software de contar con medios y herramientas que permitan construir aplicaciones corporativas se diseñó la plataforma abierta y estándar de Java para este fin, mejor conocida como J2EE (Java 2 Enterprise Edition, Java 2 edición empresarial). Se le denomina plataforma porque proporciona técnicas específicas que describen el lenguaje, pero, además, provee las herramientas para implementar productos de software basados en dichas especificaciones.

Esta plataforma ha sido diseñada para aplicaciones distribuidas con base en componentes o unidades funcionales de software que interactúan entre sí para formar parte de una aplicación empresarial. Un componente de esta plataforma debe formar parte de una aplicación y ser desplegado en un contenedor, o sea, en la parte del servidor J2EE que le ofrece al componente ciertos servicios de bajo nivel y de sistema, tales como seguridad, manejo de concurrencia, persistencia y transacciones. J2EE no es solo una plataforma o una tecnología, sino un estándar de desarrollo, construcción y despliegue de aplicaciones. Ofrece muy buenas perspectivas para la implementación de software empresarial para aquellos sistemas informáticos que requieran basar su arquitectura en productos basados en software libre.

Posee muchas ventajas como, las de proveer soporte para múltiples sistemas operativos: al ser una plataforma Java, es posible desarrollar arquitecturas basadas en J2EE usando cualquier sistema operativo donde pueda estarse ejecutando una máquina virtual de Java, tiene una independencia total de la arquitectura de hardware. Está controlada por un organismo formado por más de 400 empresas. Entre esas empresas se encuentran muchas de las más importantes del mundo informático, tales como Sun Microsystems, IBM, Oracle, BEA, HP, AOL, etc. Muchas empresas crean soluciones basadas en J2EE que ofrecen características tales como rendimiento y precio muy diferentes. De esta forma, se ha desarrollado a un nivel exponencial la plataforma y los clientes tienen la posibilidad de escoger entre una gran cantidad de opciones dándole competitividad a los productos creados en dicha plataforma. Sobre la misma es posible crear arquitecturas basadas por completo en productos de software libre. Además, pone a disposición de los arquitectos de software varias soluciones libres para cada una de las partes de su arquitectura. La experiencia y madurez de la plataforma están a favor de J2EE, ya que fue creada en el año 1997, y en dicho tiempo se han ido desarrollando multitud de productos y servicios, a la vez que se han ido corrigiendo errores y cubriendo las carencias y necesidades detectadas, por lo que hoy cuenta con una gama de productos altamente consolidados. (ORACLE: Sun Developer Network (SDN))

Para la implementación del SIIPOL se seleccionó J2EE, teniendo en cuenta, como elemento indispensable, el entorno de ejecución, ya que se requiere de un sistema multiplataforma, de alto

rendimiento, escalabilidad y seguridad, por lo que la opción más acertada para estos requerimientos es la plataforma J2EE.

1.3.4 Lenguaje de programación

Los lenguajes de programación son herramientas que nos permiten crear programas, entre ellos se encuentran: Delphi, Visual Basic, Pascal, Java, etc. Una computadora funciona bajo el control de un programa el cual debe estar almacenado en la unidad de memoria; tales como el disco duro. Los lenguajes de programación de una computadora en particular se conocen como código de máquina o lenguaje de máquina. (Lenguajes de Programación, 2009)

Estos lenguajes facilitan la tarea de programación, ya que disponen de formas adecuadas que representan los códigos en forma simbólica y en manera de un texto lo que permite ser leído y escrito por personas, a su vez resultan independientes del modelo de computador a utilizar. (Lenguajes de Programación, 2009)

Java

El lenguaje de programación Java surge a principios de los años 90 en los laboratorios de Sun Microsystems. A diferencia de los lenguajes convencionales, que generalmente están diseñados para ser compilados a código nativo, Java es compilado a un código intermedio o bytecode, el cual es interpretado por una máquina virtual de Java. La máquina virtual hace posible que una aplicación que haya sido implementada en Java se ejecute en cualquier sistema operativo con soporte para ella. Proporciona un entorno de ejecución que convierte el código neutro de Java al código nativo del ambiente en que está siendo ejecutada. Como lenguaje de programación es multipropósito, reúne todas las características de un ambiente orientado a objetos: es sencillo, cuenta con capacidad de generación de aplicaciones distribuidas, robusta, segura, de arquitectura neutral, portable, multihilo, dinámico y de alto rendimiento.

La API (Application Program Interface) de Java es muy versátil, ya que está formada por un conjunto de paquetes de clases que le proporcionan una extensa funcionalidad. El núcleo de la API cuenta con cada una de las implementaciones de la máquina virtual: tipos de datos, clases y objetos, manejo de red, seguridad, componentes, etc. Estos componentes son llamados Java Beans, los cuales son código reusable que se puede utilizar fácilmente para crear aplicaciones sofisticadas. Se puede decir que con este lenguaje, Sun Microsystems introdujo en el mercado la primera plataforma de software universal diseñada desde y para el crecimiento de Internet y de las intranets corporativas. Esta tecnología permite escribir aplicaciones una sola vez y ejecutarlas en cualquier computadora, lo que, desde entonces ha revolucionado el mundo del desarrollo de software por representar un cambio de paradigma.

En la actualidad Java es utilizado por una gran cantidad de usuarios e instituciones, ya que brinda un amplio abanico de posibilidades y casi cualquier cosa que se puede hacer en otro lenguaje se hace también en este y muchas veces con grandes ventajas. Permite programar páginas web dinámicas con accesos a bases de datos utilizando XML con cualquier tipo de conexión de red entre cualquier sistema. En general, cualquier aplicación que se desee hacer con acceso a través de la web se puede desarrollar utilizando Java. (Álvarez, 2001)

Este lenguaje permitirá la codificación de la propuesta de solución de una manera rápida, flexible, y dinámica, al integrarse con los diferentes frameworks de desarrollo. La amplia gama de recursos que posee y el soporte a la Programación Orientada a Objetos (POO), permitirán reutilizar códigos, lo que beneficiará al desarrollador reduciendo el trabajo mecánico y posibilitando una codificación rápida y fácil de entender.

1.3.5 Entorno Integrado de Desarrollo (IDE)

Un IDE (Integrated Development Environment) es un programa que incorpora un conjunto de herramientas que sirven de soporte a un programador para construir software, es considerado un programa de aplicación que incluye un editor de código, un compilador, un depurador e integración con sistemas controladores de versiones o repositorios. (LuAuF, 2008)
A medida que ha avanzado el proyecto de modernización del SIIPOL, se han utilizado varios entornos de desarrollo para obtener los beneficios propios del impulso de la tecnología. Para el desarrollo de este trabajo se utilizó el IDE Red Hat Developer Studio: con un gran reconocimiento mundial.

Red Hat Developer Studio

Novedoso entorno de desarrollo sobre eclipse, caracterizado por contar con herramientas que ofrecen un modelo de programación unificado, además de muchas otras que permiten desarrollar y depurar aplicaciones J2EE. Proporciona un entorno de desarrollo Ajax basado en RichFaces y un excelente soporte para poderosos frameworks como JSF, Spring, Hibernate entre otros.

Developer Studio se caracteriza por (La Flecha: tu diario de Ciencia y Tecnología):

- Un modelo de programación unificado: Developer Studio aumenta y proporciona nuevas herramientas alrededor de JBoss Seam para construir aplicaciones de manera sencilla y consistente. Hoy, el tipo de aplicación típicamente dicta el modelo de programación que el desarrollador utiliza, lo que significa que deben aprender a utilizar diferentes modelos. JBoss, que actualmente está siendo estandarizado como Web Beans en el Java Community Process, ofrece un modelo unificado y sencillo para desarrollar cualquier clase de aplicación, eliminando la necesidad de múltiples modelos de programación.

- Potentes capacidades Ajax: proporciona un entorno de desarrollo Ajax integrado y potente con JBoss Seam y JBoss Ajax4jsf, componentes Web JBoss RichFaces, y herramientas What You See Is What You Get (WYSIWYG) para crear interfaces y páginas web que soportan Ajax.
- Utilidades Java Platform Enterprise Edition (EE): Developer Studio hace más sencilla la construcción de aplicaciones Java EE, con capacidades como WYSIWYG y edición de JavaServer Faces FSF y páginas Facelets, asistencia de código dinámico y una paleta de componentes. Además, al incluir e integrar JBoss Application Server, Developer Studio simplifica el despliegue, la ejecución y la depuración de las aplicaciones Java EE.
- Un tiempo de ejecución integrado con las herramientas de desarrollo: Developer Studio es el primer entorno de desarrollo open source basado en Eclipse que une el runtime con las herramientas. Esto elimina la necesidad de que el desarrollador improvise estructuras y componentes open source antes de que empiece a desarrollar. Ahora pueden tener un IDE listo para todo su desarrollo.

Este IDE ha permitido una codificación cómoda a los desarrolladores, a través de las diferentes perspectivas en las que organiza los elementos en el marco de trabajo. Posee una poderosa ayuda en línea para el trabajo con las diferentes librerías de clases que le aportan a los desarrolladores valiosas informaciones ante una situación dada. La integración con el IDE de algunas herramientas como el subeclipse 1.4, permitirá sincronizar el proyecto con un repositorio; subir, actualizar o revertir cambios; ignorar contenido y manejar diversas ramas del proyecto.

1.3.6 Herramientas CASE

El concepto de CASE es muy amplio; y una buena definición genérica, que pueda abarcar esa amplitud de conceptos, sería la de considerar a la Ingeniería de Software Asistida por Computación (CASE), como la aplicación de métodos y técnicas a través de las cuales se hacen útiles a las personas comprender las capacidades de las computadoras, por medio de programas, de procedimientos y su respectiva documentación. (cyta.com)

Concentrando la atención en el uso de estas herramientas, para el desarrollo de proyectos informáticos que tengan como objetivo la automatización de procedimientos administrativos, se puede decir que:

Las herramientas CASE representan una forma que permite Modelar los Procesos de Negocios de las empresas y desarrollar los Sistemas de Información.

Algunos de los componentes de las herramientas CASE permiten:

- Confeccionar la definición de requerimientos de los usuarios.
- Mejorar el diseño de los sistemas.

- Mejorar la eficiencia en la programación (por su generación automática de códigos).
- Otorgar a la administración un mejor soporte en la documentación.

Para ello, y sin importar la arquitectura de la herramienta CASE, en general tales herramientas deben abarcar las siguientes propiedades:

- Tener una interfaz gráfica y textual, que le permita al usuario manejar los objetos de diseño.
- Contar con un Diccionario de Datos, a fin de rastrear y controlar los objetos diseñados.
- Disponer de un conjunto de herramientas que permitan: chequear las reglas del diseño y analizar la lógica del diseño.

A partir de esta descripción conceptual, sobre las herramientas; se hace notar que las herramientas CASE serán un elemento muy importante, que le permitirá al administrador de un proyecto informático, llevar adelante el mismo de forma eficaz y eficiente.

También es un hecho que estas mismas herramientas, como toda Tecnología de la Información se encuentra en continua evolución y existe además una gran variedad de proveedores y productos y cada uno de ellos con sus diferentes aplicaciones y especificaciones.

Otro elemento importante conveniente de destacar, es que las herramientas CASE, son eso: "HERRAMIENTAS", y que como tales permiten aumentar la productividad en el desarrollo de un proyecto y como herramientas que son, deben ser aplicadas a una metodología determinada.

Visual Paradigm

El Visual Paradigm es un producto de calidad, que soporta aplicaciones Web, es muy fácil de instalar y actualizar. Permite la generación de código para varios lenguajes, (en especial Java). Es una herramienta de código abierto y presenta un entorno de creación de diagramas para UML 2.0. Su diseño está centrado en casos de uso y enfocado al negocio generando un software de mayor calidad, presenta capacidades de ingeniería directa e inversa y disponibilidad en múltiples plataformas. (Entorno Virtual de Aprendizaje, 2009)

Es una herramienta diseñada para desarrollar software con programación orientada a objetos. Busca reducir la duración del ciclo de desarrollo, brindando ayuda a arquitectos, analistas, diseñadores y desarrolladores; permite el uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.

Una de las características más importantes de su uso es que brinda la posibilidad de sincronización del modelo de diseño y el código en todo el ciclo de desarrollo una vez que se integra con Eclipse,

permitiendo la facilidad de programar directamente sobre el código fuente generado y a su vez actualizar el diseño con cambios que se realicen en la programación.

Por todas las características anteriormente planteadas se escogió para la modelación del proyecto la herramienta Visual Paradigm. Esta herramienta permitirá la creación de los distintos diagramas útiles para que el desarrollador pueda implementar de una forma más dinámica la propuesta de solución. Estos diagramas darán una vista general de lo que se quiere implementar y de esta forma obtener un resultado positivo.

1.3.7 Frameworks utilizados

Un framework (término en inglés para marco de trabajo) es una estructura de soporte mediante la cual puede ser desarrollado un proyecto de software. Generalmente incluye programas, bibliotecas y un lenguaje interpretado que optimizan y aceleran el proceso de desarrollo del software. Estos diagramas dan una vista general de lo que se desea desarrollar para una mejor comprensión.

Capa de presentación

JSF (Java Server Faces), es un poderoso framework que se ha convertido en un estándar para aplicaciones web basadas en Java, facilita la construcción de aplicaciones siguiendo el patrón MVC (Modelo-Vista-Controlador); permite manipular varios eventos en cada página; posee un modelo de componentes orientado a objetos, conversión de tipos y validación, y tiene un poderoso sistema de navegación declarativa; usa simples clases Java como controladores. Permite la fácil incorporación de potencialidades AJAX, posee un conjunto prefabricado de componentes de interfaz de usuario, y permite utilizar el modelo de programación orientado a eventos.

A este framework se integran con gran facilidad algunas librerías de componentes como Ajax4JSF y RichFaces que provee a la aplicación de un efecto “aplicación de escritorio” al incorporar capacidades AJAX (del término en inglés Asincronous Java Script and XML), además de hacerla más profesional y agradable al usuario.

Capa de lógica del negocio

El Spring Framework (también conocido como Spring) es un framework de código abierto de desarrollo de aplicaciones para la plataforma Java. A pesar de que Spring Framework no obliga a usar un modelo de programación en particular, se ha popularizado en la comunidad de programadores en Java al considerársele una alternativa y sustituta del modelo de Enterprise JavaBean. Por su diseño el framework ofrece mucha libertad a los desarrolladores en Java y soluciones muy bien documentadas y fáciles de usar, prácticas comunes en la industria.

Spring Framework está diseñado como una serie de módulos que pueden trabajar independientemente uno de otro. Además, intenta mantener un mínimo acoplamiento entre la aplicación y el propio framework de forma que podría ser desvinculada de él sin demasiada dificultad.

Los principales módulos de este framework son (Walls, 2008):

- Core: Como su nombre indica, es el núcleo de Spring. Permite técnicas de Inversión del Control (IoC) como la inyección de dependencias.
- Context: Proporciona herramientas para acceder a los beans y da soporte a propagación de eventos, resource bundles, carga de recursos y creación transparente de contextos por parte de los contenedores.
- DAO: Proporciona una capa de abstracción JDBC (Java Database Connectivity) y una forma de administrar transacciones.
- ORM: Provee capas de integración para APIs de mapeo objeto-relacional.
- AOP: Proporciona una implementación de programación orientada a aspectos, permitiendo definir puntos de corte e interceptores.
- Web: Provee de características de integración orientadas a la web, como funcionalidad multiparte, inicialización de contextos mediante servlet listeners y un contexto de aplicación orientada a la web. También permite integrar de forma sencilla otros frameworks como Struts, JSF o WebWork.
- Spring MVC: Provee una implementación Modelo-Vista-Controlador que permite el uso del resto de las funcionalidades del Spring Framework.

Entre las principales características de este framework se encuentran (Walls, 2008):

- Se centra en la capa intermedia y proporciona enganches para manejar la solución elegida para la capa de presentación y de integración.
- Permite Internacionalización (i18n) y localización (L10n).
- Spring viene integrado con un framework de seguridad, Acegi Security, que gestiona todo el mecanismo de autentificación y autorización.
- No proporciona un mecanismo de plantillas propio, ya que permite elegir la tecnología a utilizar para la vista (JSP, JSF, velocity, PDF…) integrándola con el resto de la aplicación.

- Provee un sistema de validación que no está ligado a una capa concreta, si no que puede ser usado en cualquier capa de la aplicación. Provee de una interfaz para escribir tu lógica de validación y se encarga del manejo de errores. El sistema de seguridad Acegi también ofrece un paquete de validación.
- Para los sistemas de navegación ofrece el módulo Spring Web Flow que proporciona un motor capaz de capturar los flujos de páginas de una aplicación e integrarlo con otros frameworks de presentación como JSF.
- No proporciona un sistema de caché nativo, pero su diseño permite la integración de módulos que se encarguen de ello, como por ejemplo Spring AOP Cache.
- En cuanto al mapeo objeto-relacional (ORM) proporciona integración con varias implementaciones ORM. Existen dos formas de integración, a través de plantillas predefinidas del módulo SpringDAO o codificando DAOs directamente contra la API del ORM elegido. Cualquiera de las dos aproximaciones ofrece los beneficios de Spring, como ser configurados a través de IoC (Inversión del Control), transaccionalidad, wrapping común para excepciones de acceso a datos y manejo de la configuración independiente de la implementación.
- La Inyección de dependencias (DI) es una de las bases de Spring sobre la que se cimienta el resto de la arquitectura.
- Al permitir la elección de la tecnología de la capa de presentación, el soporte para Ajax dependerá principalmente de la elección.
- La configuración de Spring está basada en XML. Al no tener anotaciones no le hace dependiente de Java EE 5. Aunque en la versión 2.5 se introduce el uso de las anotaciones.
- El diseño de Spring está pensado para ofrecer un modelo de cómo debe trabajar la aplicación y cómo se comunican sus partes. Está expresamente concebido para que deba ser extensible y acoplable con otros frameworks, ya que no ofrece una solución completa que abarque desde la presentación al modelo.
- A pesar de su juventud (Spring 1.0 fue lanzado en el año 2003), Spring ha demostrado tener una arquitectura sólida y, sobre todo, muy flexible, capaz de adaptarse a los requerimientos de proyectos grandes y pequeños.
- No puede decirse que la curva de aprendizaje sea suave debido a que Spring introduce conceptos relativamente novedosos como son la AOP o la IoC y a la cantidad de configuración que hay que manejar. Sin embargo para alguien con experiencia en desarrollo J2EE y familiarizado con dichos conceptos no debería resultar muy difícil. Para facilitar el desarrollo existen IDEs open source como SpringIDE, que ayudan entre otras cosas a lidiar con los archivos de configuración y diseñar el flujo de la aplicación.

- Cuenta con una buena documentación oficial que cubre todos sus módulos y funcionamiento así como su integración con otros frameworks. También tiene una nutrida comunidad de usuarios que aportan artículos y trabajos.

Capa de acceso a datos

Hibernate es una poderosa herramienta de Mapeo objeto-relaciona Open Source, que al igual que los de su tipo están diseñados para casar el modelo relacional con el objetual, permitiendo desarrollar clases persistentes siguiendo las pautas de la programación orientada a objeto tal como asociaciones, herencia, polimorfismo, composición y manejo de colecciones.
Entre sus características más significativas están (Baver, et al., 2005):

1. Permite generar una base de datos a partir del modelo de clases, para ello tiene compatibilidad con los motores de base de datos más actuales: Oracle, DB2, MySQL, Postgres y más, esto se logra especificándole en un archivo XML el driver a usar en dependencia del motor escogido.
2. Mapeo entre las entidades persistentes y las tablas de la base de datos, soportando los diferentes tipos de relaciones y amplia gama de tipos de datos: string, boolean, double, integer, date, etc.
3. Posee un lenguaje propio de consultas HQL (Hibernate Query Language) e incorpora una API mediante el cual se pueden efectuar consultas programáticamente llamadas "criteria"; todo esto sin quitar el uso del SQL nativo.

Todas estas características liberan al desarrollador de las preocupaciones del tipo de motor de base de datos, así como el lenguaje nativo a usar en cada caso, aparte de que facilita trabajar limpiamente en un modelo objetual y brinda la flexibilidad de usar SQL en caso que la situación lo permita, en la mayoría de los casos para buscar rendimiento.

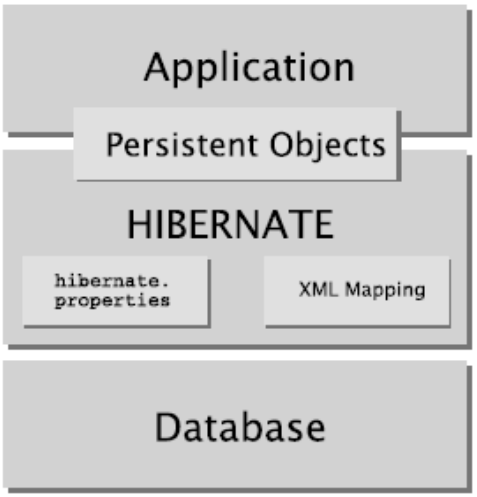

Figura 2. Estructura de Hibernate.

1.3.8 Propuesta de solución

Una aplicación web es un sistema informático que los usuarios utilizan accediendo a un servidor a través de Internet o de una intranet. Estas aplicaciones son populares debido a la practicidad del navegador como cliente ligero y la facilidad para actualizar y mantener aplicaciones de este tipo sin distribuir e instalar software.

Como propuesta de sistema para el SIIPOL, se propuso una aplicación web con la siguiente estructura:

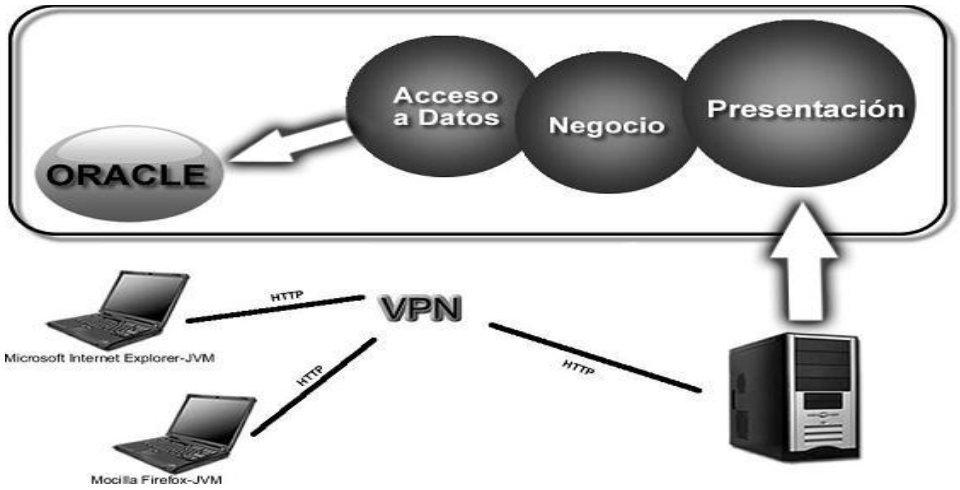

Figura 3. Propuesta de Sistema para el SIIPOL.

Aplicación web basada en tecnología Java, con un gestor de base de datos Oracle, un servidor de aplicaciones Apache Tomcat 5.5, y una arquitectura en tres capas. A este sistema se integra el Submódulo Servicios el cual cuenta con la siguiente arquitectura:

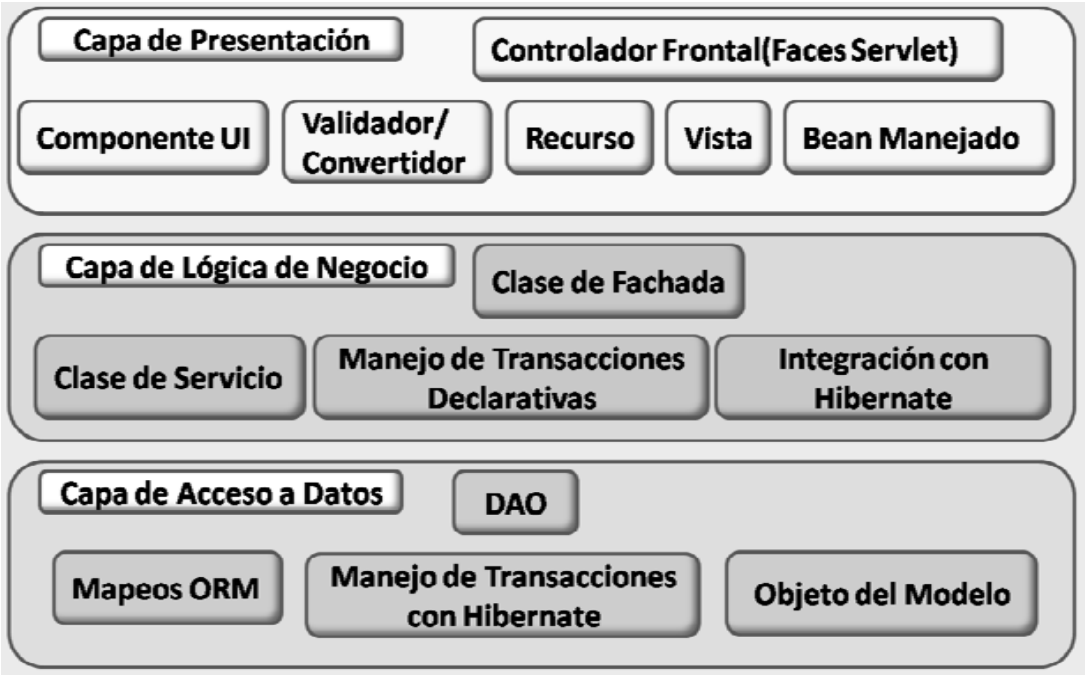

Figura 4. Arquitectura para el Submódulo Servicios.

Como se observa en la figura anterior, es una arquitectura en tres capas donde se identifican las capas de presentación, lógica de negocio y acceso a datos que dará soporte a la propuesta de solución.

El sistema SIIPOL está organizado en diferentes módulos entre los que se encuentran Administración, Análisis de Información, Aprehensión, Estadística, Gestión Administrativa, Investigación Criminalística, Investigación Penal, Registro y Control e Investigación Forense, dentro este último se encuentra el Submódulo Servicios. La siguiente imagen muestra como estaría organizado el Submódulo dentro del ambiente de desarrollo:

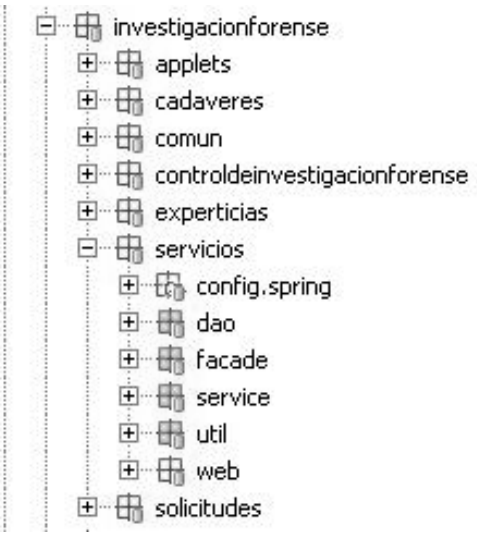

Figura 5. Estructura del Submódulo Servicios.

El diagrama de despliegue es utilizado normalmente para representar el modelo de despliegue. A continuación se muestra el diagrama que representa cómo será desplegado el sistema del cual forma parte el Submódulo Servicios:

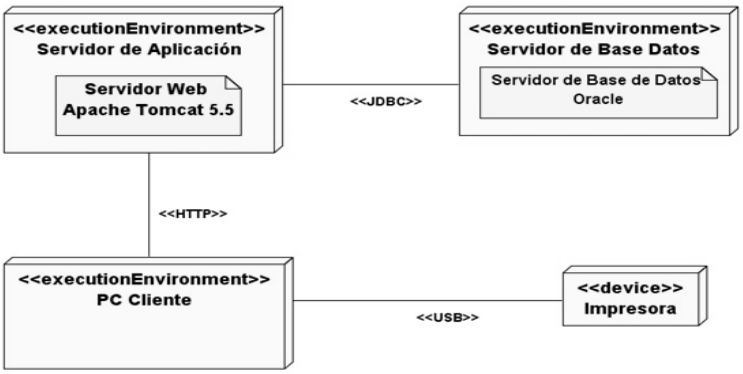

Figura 6. Diagrama de Despliegue.

1.4 Conclusiones parciales

Con la realización de este capítulo se sentaron las bases para el desarrollo del proceso, pues al culminar su elaboración se obtuvieron los elementos conceptuales necesarios para dar lugar a la construcción de una aplicación que cumpla con las metas propuestas.

Aunque los sistemas de gestión de información policial estudiados no revelan elementos de codificación u otros aspectos específicos, si presentan características que permiten a partir de su estudio la construcción de una aplicación que cumpla con los paradigmas trazados en el mundo.

La metodología seleccionada permitirá el seguimiento de un proceso de desarrollo que facilite el trabajo de todos los implicados en el proyecto y la obtención de un producto con las características esperadas.

El ambiente de desarrollo integrado por la plataforma J2EE, el IDE Red Hat Developer Studio, el gestor de base de datos Oracle, los frameworks JSF, Spring e Hibernate y la herramienta de modelado Visual Paradigm, así como el uso de un estilo arquitectónico en 3 capas, permitirá el desarrollo claro y fluido de un sistema construido sobre bases sólidas y un entorno de desarrollo bien definido.

A partir de este análisis se dará paso a los temas específicos del Módulo de Investigación en Ciencias Forenses y específicamente el análisis, diseño e implementación del Submódulo Servicios.

CAPÍTULO 2. ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE LA PROPUESTA DE SOLUCIÓN.

2.1 Introducción

En el presente capítulo se tratarán los temas relacionados con el análisis y diseño e implementación de la propuesta de solución. Se mostrarán las descripciones de los casos de uso del Submódulo Servicios, los cuales fueron el resultado de la Ingeniería de Requerimientos que se tuvo como entrada para el desarrollo de la propuesta de solución, así como los diagramas generados durante el proceso.

2.2 Submódulo Servicios. Requisitos.

El Submódulo Servicios brinda apoyo a los especialistas forenses de otras áreas dentro de la CNCF, tiene como objetivos principales el control de los medios y recursos que son utilizados en la realización de los diferentes estudios, así como el control administrativo de las actividades realizadas por el equipo técnico. Entre los principales servicios que brinda se encuentran Fotográfico, Radiográfico, Preparación de Muestra y Planificación de Citas.

Con los requisitos levantados por los analistas se obtuvieron 9 casos de uso para el Submódulo Servicios, los cuales son reflejados en el siguiente diagrama de casos de uso del sistema:

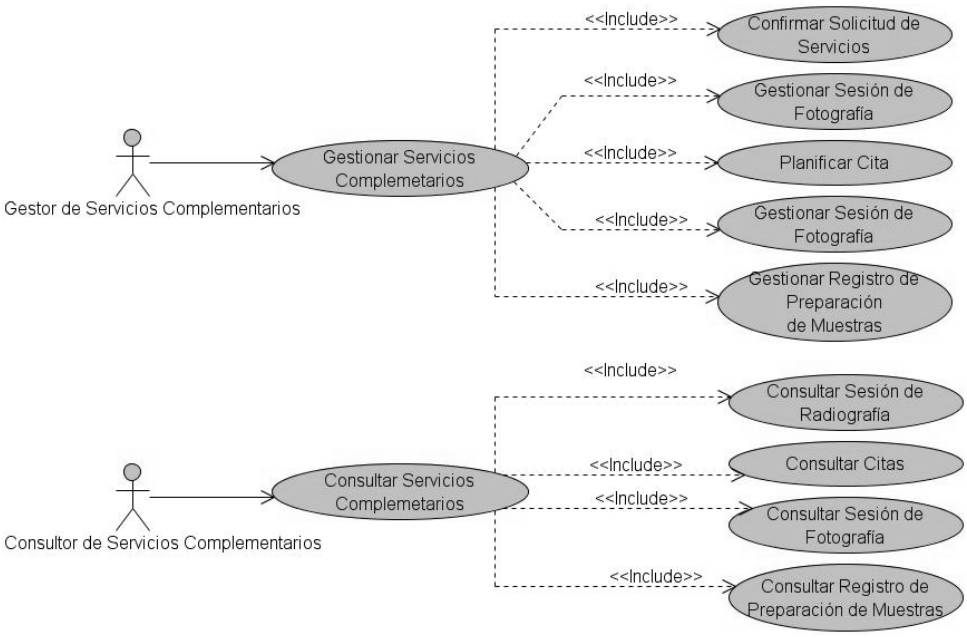

Figura 7. Diagrama de casos de uso del sistema.

Teniendo en cuenta que los procesos a los que responden estos casos de uso tienen una lógica muy similar, solo tres de ellos fueron seleccionados para incluirse en el documento y representar los artefactos generados en los flujos de trabajo Análisis y Diseño e Implementación. Los procesos de gestión de sesión de fotografía, radiografía y registro de preparación de muestras tienen una lógica bastante similar, es por ello que se escogió el CU Gestionar Registro de Preparación de Muestras en representación de los procesos antes mencionados. Siguiendo la lógica antes expuesta se selecciona el CU Consultar Registro de Preparación de Muestras, por último, se representa el CU planificar Cita, pues presenta una naturaleza y lógica diferente de los demás.

A continuación se muestra una breve descripción de los casos de uso seleccionados:

Nombre del CU	Gestionar Registro de Preparación de Muestras.
Actor	Gestor de Registros de Preparación de Muestras.
Descripción	El caso de uso inicia cuando el actor accede a la opción de realizar una acción sobre un Registro de Preparación de Muestras seleccionado. En caso que el actor haya seleccionado la opción de incluir un Registro de Preparación de Muestras, el sistema brinda la posibilidad de introducir la información de un Registro de Preparación de Muestras. En caso que el actor haya seleccionado la opción de ver los detalles de un Registro de Preparación de Muestras, el sistema muestra la información del mismo, si el registro ha sido rechazado muestra los comentarios de la revisión. En caso que el actor haya seleccionado la opción de modificar un Registro de Preparación de Muestras, el sistema muestra toda la información del mismo, permitiendo su edición.
Referencia	Gestionar Registro de Preparación de Muestras. • Incluir un Registro de Preparación de Muestras. • Mostrar la información de un Registro de Preparación de Muestras. • Modificar la información de un Registro de Preparación de Muestras. Asociar elemento relacionado.

	Disociar elemento relacionado. Validar la integridad de los datos introducidos por el usuario. Mantener al usuario informado del resultado de las operaciones. Imprimir o exportar a formato PDF un Registro de Preparación de Muestras.
Nombre del CU	Consultar Registro de Preparación de Muestras.
Actor	Consultor de Registros de Preparación de Muestras.
Descripción	El caso de uso inicia cuando el actor accede a la opción de consultar los Registros de Preparación de Muestras registrados en el sistema. El sistema muestra un listado de Registros de Preparación de Muestras que están registrados en el sistema y que aún no han sido confirmados por el solicitante, permitiendo el filtrado de los mismos por diversos criterios (como por ejemplo: experto solicitante, rango de fechas, etc.), así como la búsqueda de Registros de Preparación de Muestras incluidos históricamente, también por diversos criterios (mencionados anteriormente). El actor puede imprimir o exportar a formato PDF el listado de Registros de Preparación de Muestras. El sistema permite seleccionar un elemento de la lista.
Referencia	Consultar Registros de Preparación de Muestras. • Mostrar los Registros de Preparación de Muestras que se encuentran sin confirmar. • Realizar una búsqueda o filtrado por diversos criterios entre todos los Registros de Preparación de Muestras. • Ordenar los resultados de manera ascendente o descendente por campos. Imprimir o exportar a formato PDF un listado de Registros de Preparación de Muestras. Seleccionar un Registro de Preparación de Muestras de la lista. Mantener informado al usuario del resultado de las operaciones.

.

Nombre del CU	Planificar Cita.
Actor	Planificador de Citas.
Descripción	El caso de uso se inicia cuando el actor selecciona la opción que le permite realizar una acción sobre la Tarjeta de Cita seleccionada. En caso que el actor haya seleccionado la opción de planificar una Cita, el sistema brinda la posibilidad de introducir la información de una Tarjeta de Cita. El actor especifica la fecha y hora así como el motivo de la nueva Cita. En caso de que el actor haya seleccionado la opción de replanificar una Tarjeta de Cita, el sistema muestra la información de la misma, permitiendo su edición. Para ambos casos el sistema comprueba que la nueva Cita no entra en conflicto con Citas previamente planificadas por el funcionario y guarda la Tarjeta de Cita. En caso que el actor haya seleccionado la opción de ver los detalles de una Tarjeta de Cita, el sistema muestra la información de la misma. Si el actor selecciona la opción de imprimir la tarjeta el sistema muestra una vista preliminar de la Tarjeta de Cita correspondiente a la cita planificada. El actor puede imprimir o exportar a formato PDF la Tarjeta de Cita.
Referencia	Planificar una cita. Asociar solicitud a la Tarjeta de Cita. Disociar solicitud de la Tarjeta de Cita. Mostrar los datos de la Tarjeta de Cita. Imprimir o exportar a formato PDF una Tarjeta de Cita. Replanificar cita.

	Validar que no exista coincidencia entre citas. Mantener al usuario informado del resultado de las operaciones.

Requisitos suplementarios más relevantes:

Funcionalidad

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

Usabilidad

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

Interfaz de usuario

- 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.
- El sistema proporcionará claridad y buena organización de la información, permitiendo la interpretación correcta e inequívoca de esta.
- Todos los textos y mensajes en pantalla aparecerán en idioma español.
- Los errores serán visibles al usuario e incluirán sugerencias de las posibles soluciones.
- Ante la ocurrencia de errores, el sistema señalará los campos incorrectos, ya sea porque contienen información incompleta, o porque se encuentran vacíos.

Seguridad

- El sistema brindará permisos para la ejecución, acceso o visualización de las funcionalidades al usuario autenticado en dependencia de los perfiles asignados a este último.

Restricciones de diseño

- El sistema estará basado en un estilo arquitectónico en capas.
- Capa de Presentación: incluye los componentes de interfaz de usuario (páginas web y componentes visuales) para interactuar y mostrar el resultado de las peticiones de servicio que ofrece la Capa de Aplicación, formateados para los distintos tipos de interfaces de usuario.
- El sistema usará el Framework de Presentación JSF para manejar la Capa de Presentación.

2.3 Modelo de Análisis

Descrito por el lenguaje del desarrollador, el análisis es una vista interna del sistema, estructurado por clases y paquetes estereotipados. Es utilizado fundamentalmente por los desarrolladores para comprender cómo debería darse forma al sistema, es decir, cómo debería ser diseñado e implementado. Sirve como una primera aproximación del diseño y su objetivo es comprender perfectamente los requisitos del software y no precisar cómo se implementará la solución. 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. (Entorno Virtual de Aprendizaje, 2009)

2.3.1 Diagrama de clases del análisis

Un Diagrama de clases del análisis es un artefacto en el que se representan los conceptos en un dominio del problema. Representa los objetos del mundo real, no de la implementación automatizada de estos objetos. Las clases del análisis se centran en los requisitos funcionales y son evidentes en el dominio del problema porque representan conceptos y relaciones del dominio. RUP propone clasificar a las mismas en interfaz, control y entidad. (Entorno Virtual de Aprendizaje, 2009)

En una aplicación de tres capas, en la capa de usuario aparecen fundamentalmente clases interfaz ya que allí se ejecutan las aplicaciones del cliente. En la capa intermedia están las clases controladoras que agrupan los servicios y funcionalidades del sistema. En la última capa estarían las clases entidad porque allí se tiene la base de datos.

A continuación se presentan los diagramas de clases del análisis de los casos de uso seleccionados:

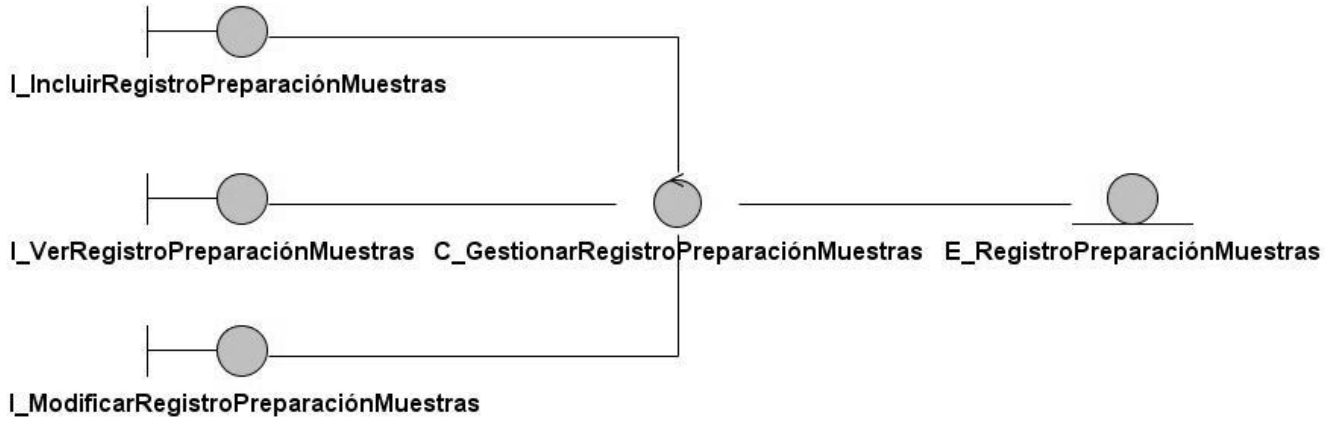

Figura 8. Diagrama de Clases de Análisis. CU Gestionar Registro de Preparación de Muestras.

Figura 9. Diagrama de Clases de Análisis. CU Consultar Registros de Preparación de Muestras.

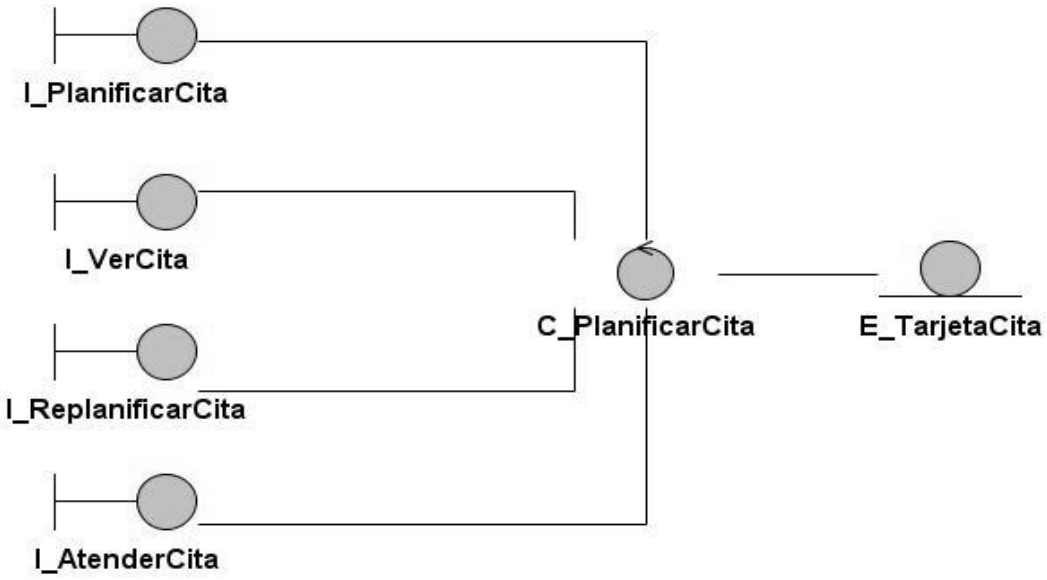

Figura 10. Diagrama de Clases de Análisis. CU Planificar Cita.

2.3.2 Diagramas de Colaboración

La realización de casos de uso del análisis describe cómo se lleva a cabo un caso de uso determinado y la interacción de las clases del análisis. Para representar dichas interacciones se construyeron diagramas de colaboración para los casos de uso seleccionados:

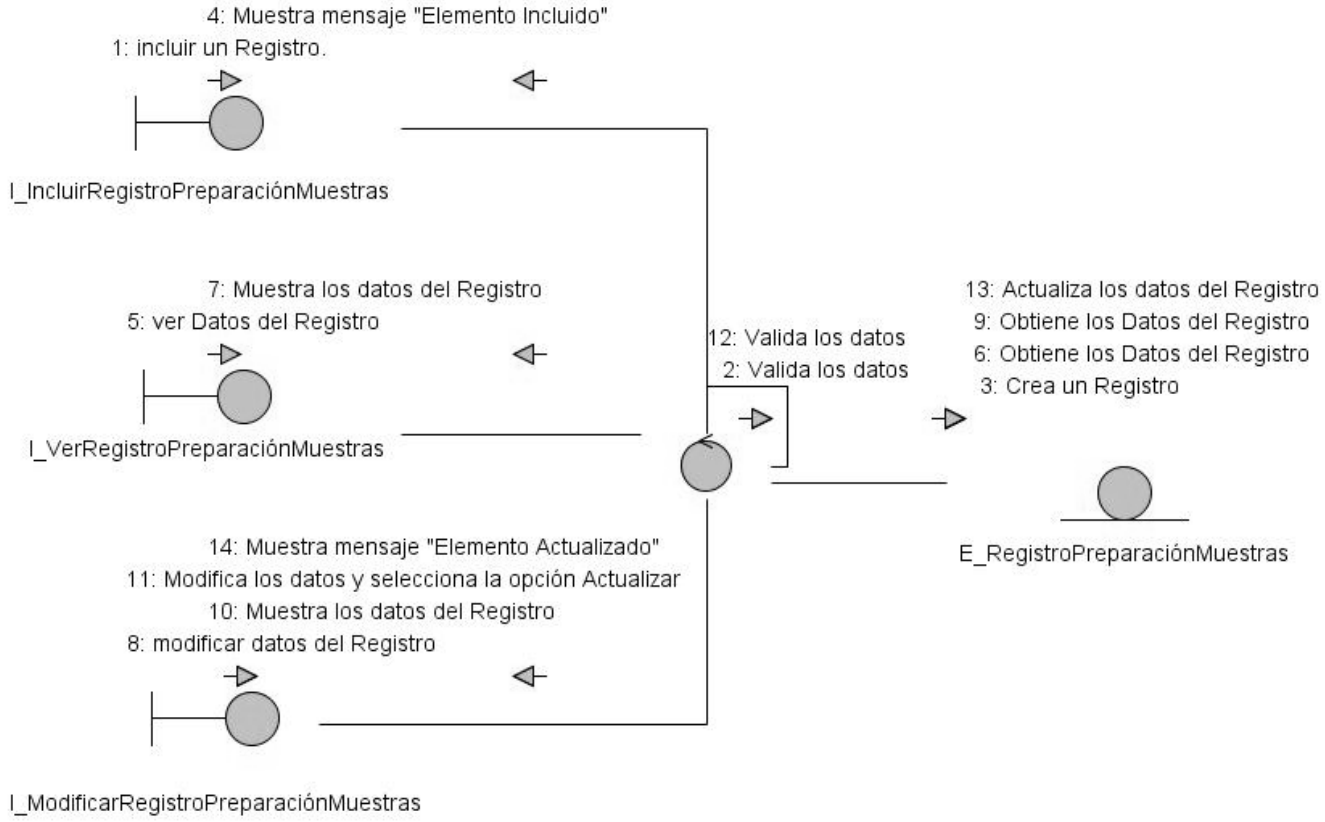

Figura 11. Diagrama de Colaboración. CU Gestionar Registro de Preparación de Muestras.

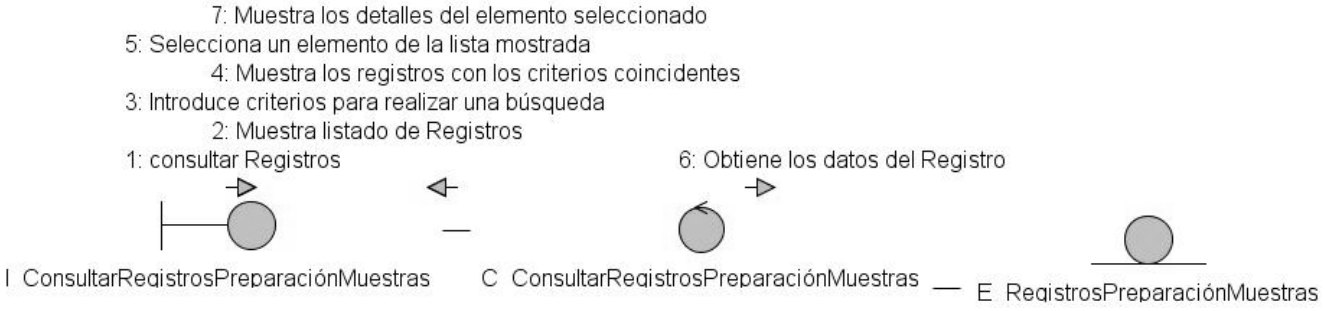

Figura 12. Diagrama de Colaboración. CU Consultar Registros de Preparación de Muestras.

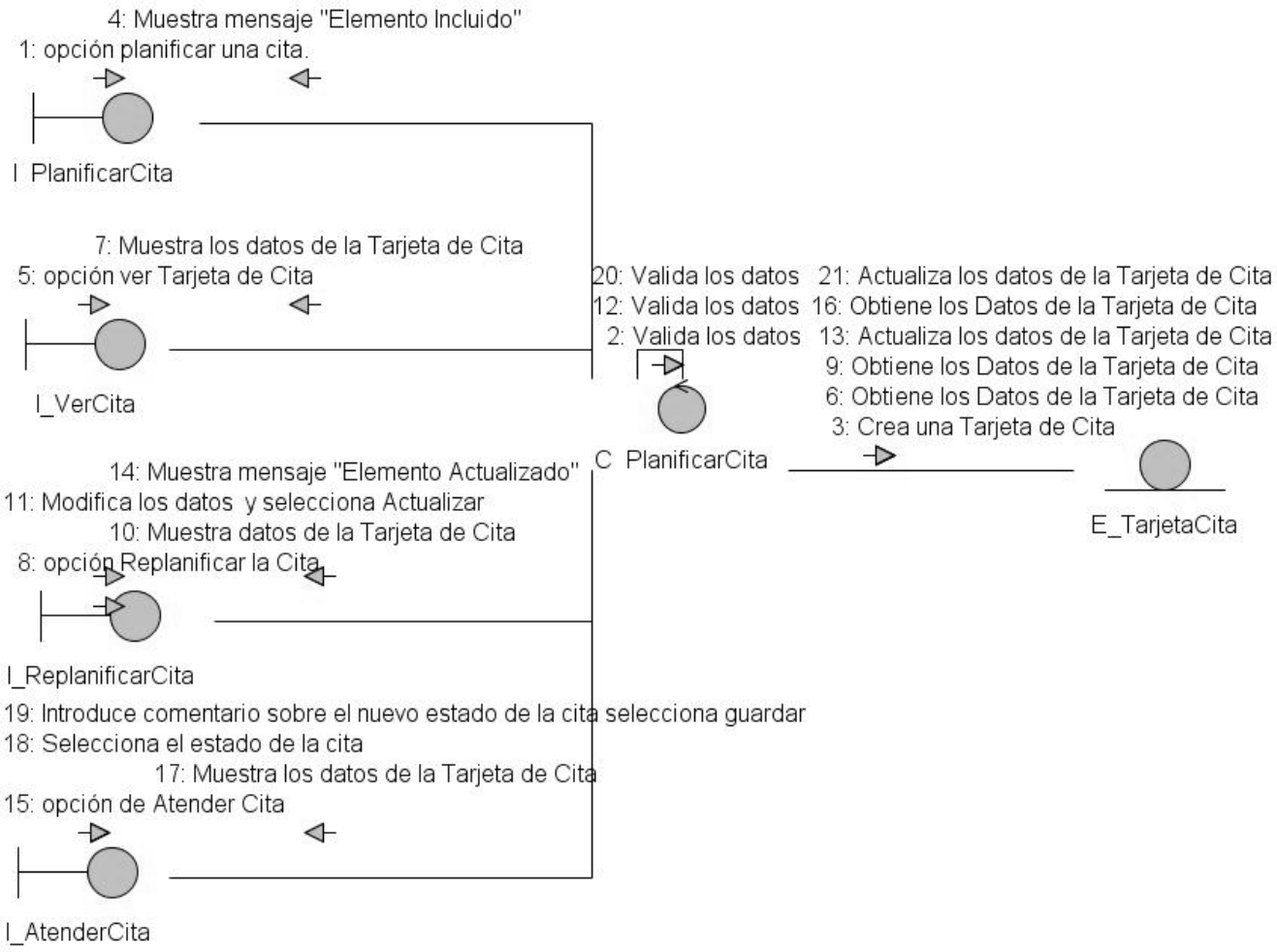

Figura 13. Diagrama de Colaboración. CU Planificar Cita.

2.4 Modelo de Diseño

En el flujo de trabajo de diseño se modela el sistema y se encuentra la forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. Una entrada esencial en el diseño es el resultado del análisis, o sea, el modelo de análisis. El diseño tiene el propósito de formular los modelos que se centran en los requisitos no funcionales y en el dominio de la solución y que preparan para la implementación y prueba del sistema. Es un modelo físico, no genérico, específico para una implementación. (Entorno Virtual de Aprendizaje, 2009)

Concretamente se puede definir como propósitos del diseño (Entorno Virtual de Aprendizaje, 2009):

- Adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables,

sistemas operativos, tecnologías de distribución y concurrencia y tecnologías de interfaz de usuario.

- Crear una entrada apropiada y un punto de partida para actividades de implementación, capturando los requisitos o subsistemas individuales, interfaces y clases.

- Descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo.

- Capturar las interfaces entre los subsistemas antes en el ciclo de vida del software, lo cual es muy útil cuando se utilizan interfaces como elementos de sincronización entre diferentes equipos de desarrollo.

Algunos de los tipos de clases del diseño son (ALBET, SA, 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, borrar y 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, e 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 bases del mismo. Los convertidores transforman una cadena del lado de la vista, a un tipo de dato Java en el lado del servidor y viceversa.

2.4.1 Diagramas de Paquetes

Los paquetes reflejan la arquitectura de alto nivel de un sistema: su descomposición en subsistemas y sus dependencias. Una dependencia entre paquetes resume las dependencias entre los contenidos del paquete. Los mismos están organizados de manera funcional, siguiendo un cierto principio racional, tal como: funcionalidad común, implementación estrechamente relacionada, y un punto de vista común. Los paquetes pueden contener otros paquetes. Hay un paquete raíz, que contiene indirectamente el modelo completo de un sistema. (Rumbaugh, et al.)

A continuación se muestra el diagrama de paquetes para el Submódulo Servicios y su relación con otros subsistemas:

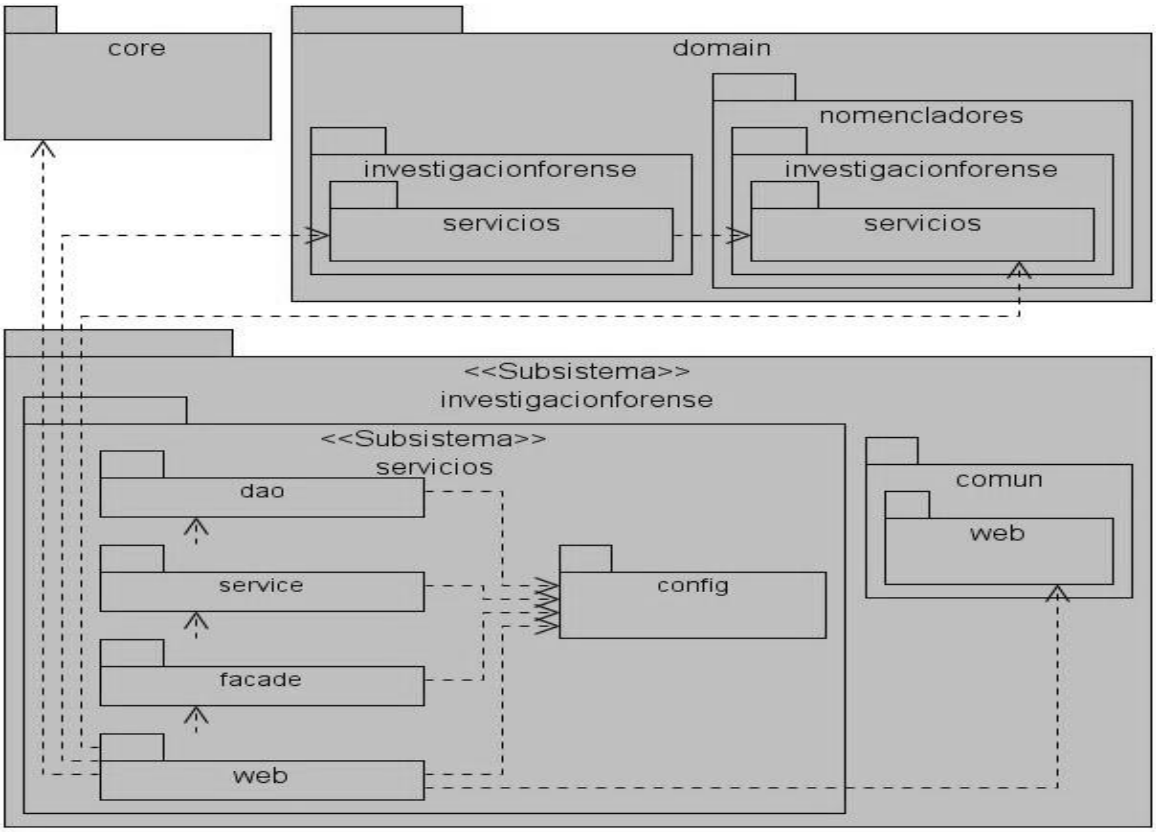

Figura 14. Diagrama de Paquetes para el Submódulo Servicios.

2.4.2 Diagramas de Clases del Diseño

Las clases de diseño reflejan un mayor nivel de detalle, se conciben para satisfacer los requisitos funcionales y no funcionales, teniendo en consideración la tecnología en la cual se implementará el diseño.

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 los casos de uso seleccionados quedarían de la siguiente manera:

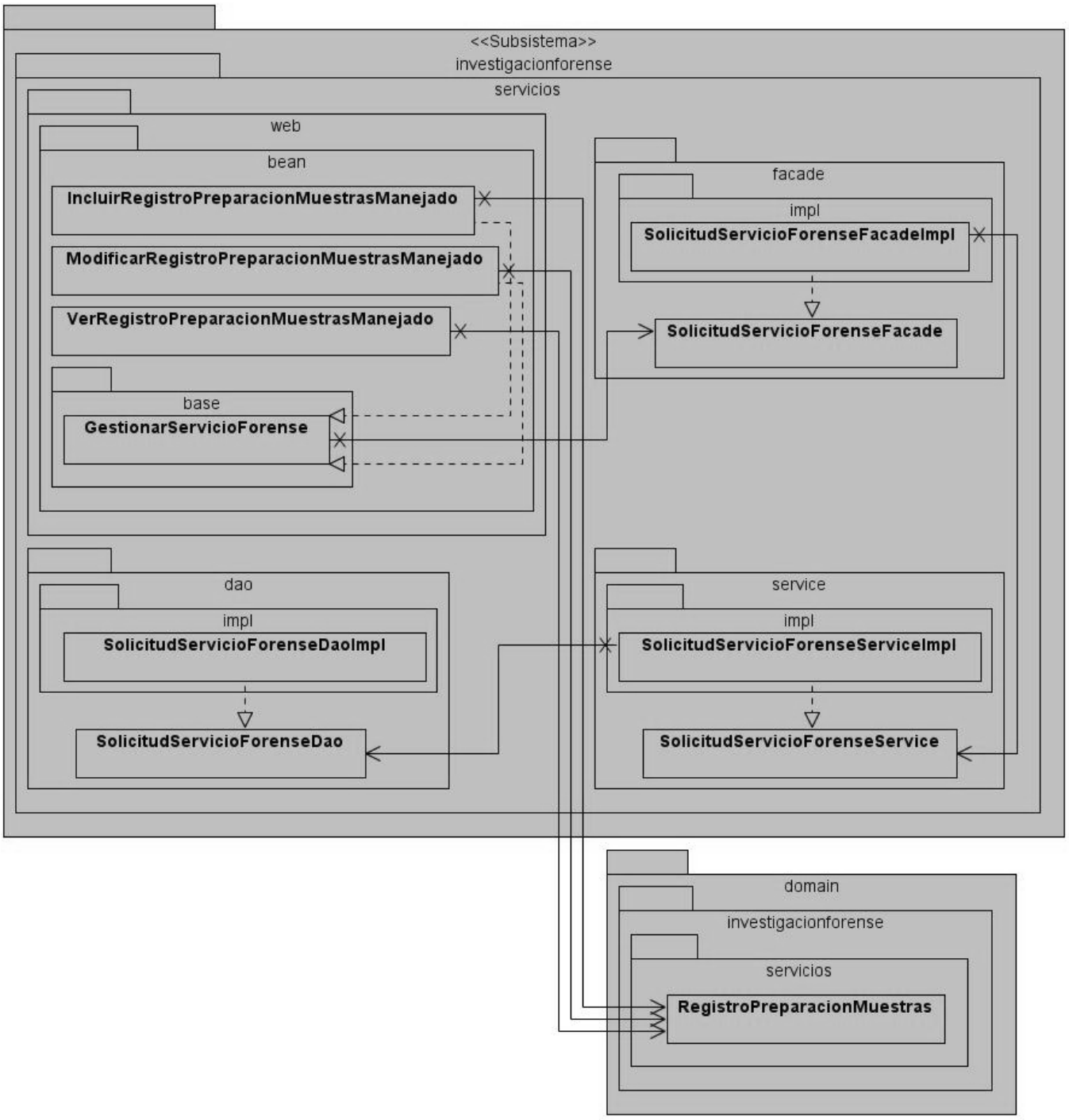

Figura 15. Diagrama de Clases de Diseño. CU Gestionar Registro de Preparación de Muestras.

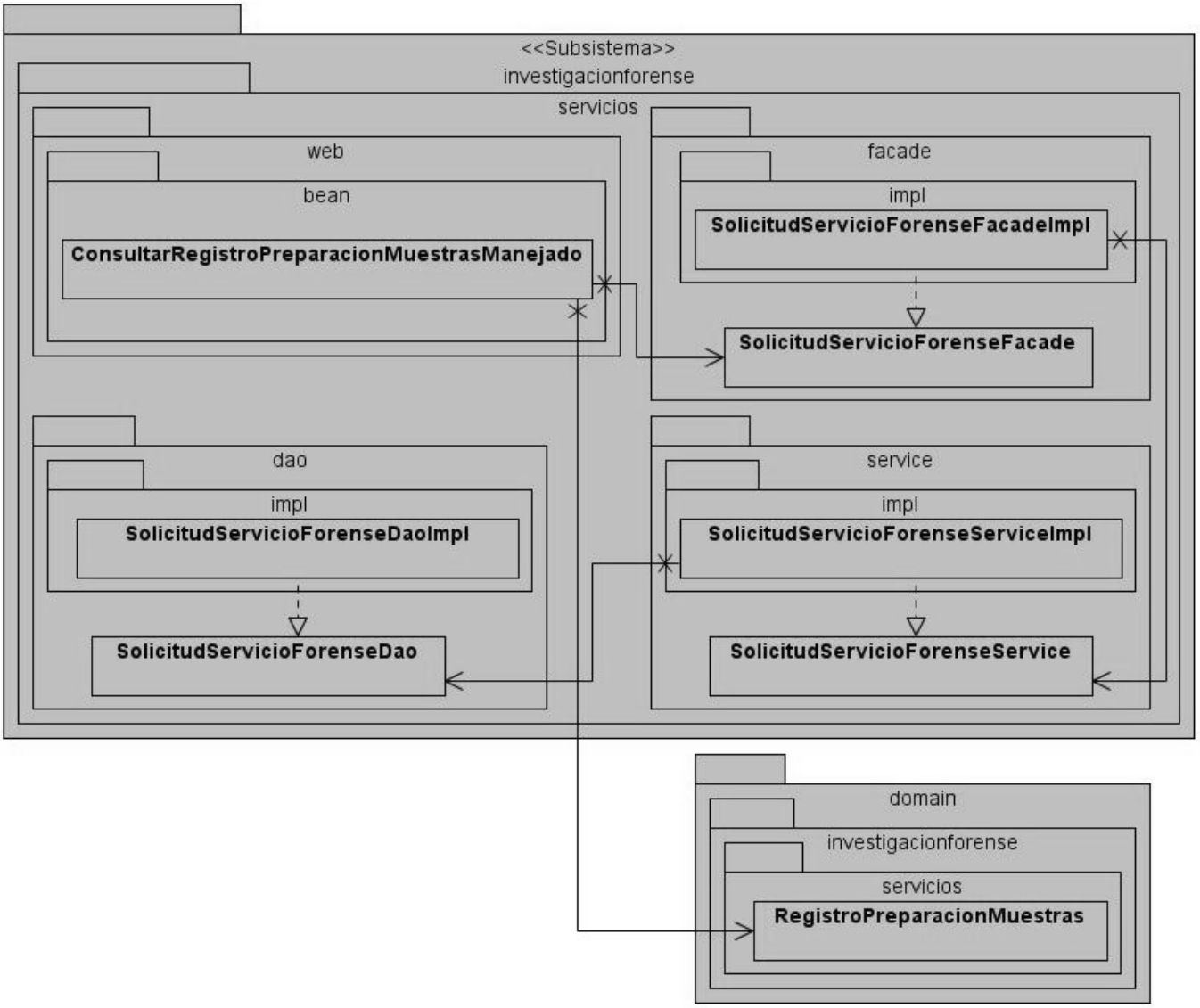

Figura 16. Diagrama de Clases de Diseño. CU Consultar Registros de Preparación de Muestras.

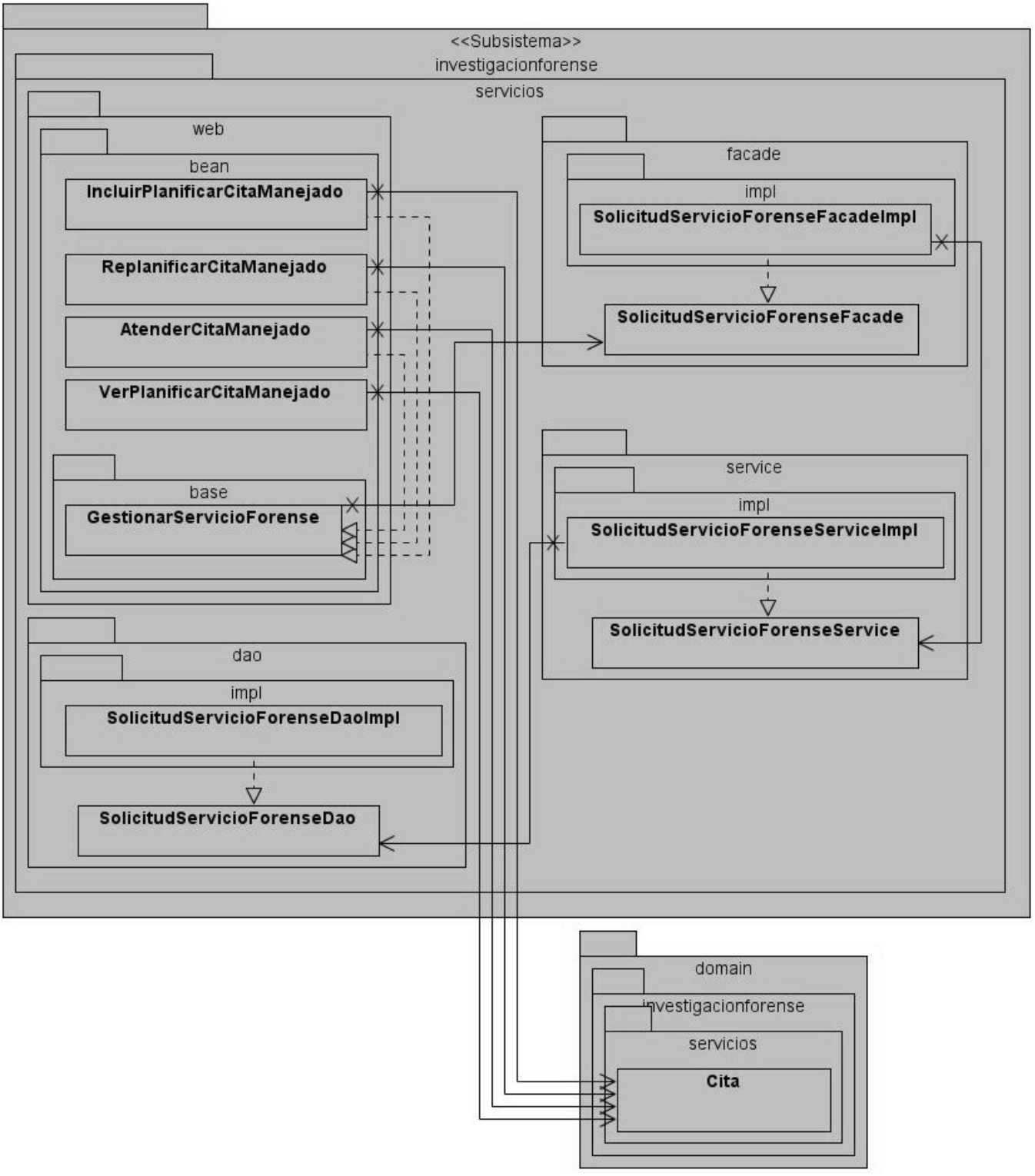

Figura 17. Diagrama de Clases de Diseño. CU Planificar Cita.

2.4.3 Clases Significativas para la solución

A continuación se presenta la descripción de las clases más importantes de los CUs analizados en el documento. Se describen en tablas nombradas tarjetas CRC, donde se muestra de una forma más concisa y representativa por cada método de una clase, sus respectivas dependencias y lo que emplea, lo cual da una visión clara de lo que se necesita para su implementación.

Las tarjetas CRC pueden ser utilizadas en cualquier espacio de trabajo, no necesariamente con una computadora, lo que posibilita ganar en tiempo. Además, permite a los participantes en las sesiones de diseño tener una experiencia cercana sobre cómo el sistema trabajará. No existe una herramienta informática que pueda sustituir la interacción física con las tarjetas, su manipulación, su recogida y despliegue. Constituyen una herramienta sumamente útil para el modelado de sistemas. Permiten además la construcción del mismo desde sus partes más simples hasta sus interacciones más complejas a partir de la unión de varias tarjetas.

CU Gestionar Registro de Preparación de Muestras. Los datos de este CU son procesados mediante las clases representadas en la siguiente figura:

Tabla 1. Descripción del Bean Manejado IncluirRegistroPreparacionMuestrasManejado.

Nombre: IncluirRegistroPreparacionMuestrasManejado	
Tipo de Clase: Controladora	
Atributo	Tipo
tejido	TipoTejido
listaTejidos	List<TipoTejido>
Para cada responsabilidad	
Nombre:	inicializarIncluir ()
Descripción:	Esta función inicializa los datos predeterminados que debe tener un Registro de Preparación de Muestras.
Nombre:	incluir (ActionEvent e)
Descripción:	Esta función incluye el Registro de Preparación de Muestras

	validando los datos.
Nombre:	adicionarTejido (ActionEvent e)
Descripción:	Esta función adiciona los datos de un tejido al listado de tejidos obtenidos.
Nombre:	eliminarTejido (ActionEvent e)
Descripción:	Esta función elimina los datos de un tejido del listado de tejidos obtenidos.

Tabla 2. Descripción del Bean Manejado VerRegistroPreparacionMuestrasManejado.

Nombre: VerRegistroPreparacionMuestrasManejado	
Tipo de Clase: Controladora	
Atributo	Tipo
registroPreparacionMuestras	RegistroPreparacionMuestras
tiposTejidos	List<TipoTejido>
comentariosRevision	List<ComentarioRevisionSolicitudes>
Para cada responsabilidad	
Nombre:	modificar (ActionEvent e)
Descripción:	Esta función muestra la página que permite modificar el registro de preparación de muestra.

Tabla 3. Descripción del Bean Manejado ModificarRegistroPreparacionMuestrasManejado.

Nombre: ModificarRegistroPreparacionMuestrasManejado
Tipo de Clase: Controladora

Atributo	Tipo
tejido	TipoTejido
listaTejidos	List<TipoTejido>
tablaTegidos	UIData
urlAnteriorVer	String
Para cada responsabilidad	
Nombre:	modificarRegistroPreparacionMuestras(RegistroPreparacionMuestras registroPreparacionMuestras, String urlAnterior, String urlAnteriorVer)
Descripción:	Esta función muestra los datos del registro de preparación de muestra entrado como parámetro para que pueda ser modificado.
Nombre:	actualizar (ActionEvent e)
Descripción:	Esta función actualiza los cambios realizados sobre el registro de preparación de muestra y valida que los datos sean correctos.
Nombre:	adicionarTejido (ActionEvent e)
Descripción:	Esta función adiciona los datos de un tejido al listado de tejidos obtenidos.
Nombre:	eliminarTejido (ActionEvent e)
Descripción:	Esta función elimina los datos de un tejido del listado de tejidos obtenidos.

Tabla 4. Descripción del Bean Manejado ConsultarRegistroPreparacionMuestrasManejado.

Nombre: ConsultarRegistroPreparacionMuestrasManejado	
Tipo de Clase: Controladora	
Atributo	Tipo

listado	List<RegistroPreparacionMuestras>
Para cada responsabilidad	
Nombre:	inicializarConsultar()
Descripción:	Esta función muestra el listado de registros de preparación de muestras que no han sido confirmadas.
Nombre:	consultar (ActionEvent e)
Descripción:	Esta función muestra el listado de registros de preparación de muestras coincidentes con los criterios de búsquedas introducidos y/o seleccionados.
Nombre:	seleccionarElemento(ActionEvent e)
Descripción:	Esta función muestra los datos del registro de preparación de muestras seleccionado.
Nombre:	nuevaBusqueda(ActionEvent e)
Descripción:	Esta función limpia los campos para realizar una nueva búsqueda.

Tabla 5. Descripción del Bean Manejado IncluirPlanificarCitaManejado.

Nombre: IncluirPlanificarCitaManejado	
Tipo de Clase: Controladora	
Atributo	Tipo
horarioCitaDesde	Date
horarioCitaHasta	Date
Para cada responsabilidad	
Nombre:	incluir(ActionEvent e)

Descripción:	Esta función incluye la tarjeta de cita validando que todos los datos sobre la misma sean correctos.
Nombre:	inicializarIncluir()
Descripción:	Esta función inicializa todos los datos predeterminados para incluir una tarjeta de cita.

Tabla 5. Descripción del Bean Manejado ReplanificarCitaManejado.

Nombre: ReplanificarCitaManejado	
Tipo de Clase: Controladora	
Atributo	Tipo
horarioCitaDesde	Date
horarioCitaHasta	Date
Para cada responsabilidad	
Nombre:	actualizar(ActionEvent e)
Descripción:	Esta función actualiza los cambios realizados sobre los horarios de la cita y valida que sean correctos.
Nombre:	replanificarCita (String urlAnterior, Cita cita)
Descripción:	Esta función muestra los datos de la cita entrada como parámetro para que pueda ser replanificada.

Tabla 6. Descripción del Bean Manejado AtenderCitaManejado.

Nombre: AtenderCitaManejado	
Tipo de Clase: Controladora	
Atributo	Tipo

cita	Cita
estado	Boolean
estadoCitaItems	List<SelectItem>
Para cada responsabilidad	
Nombre:	guardar(ActionEvent e)
Descripción:	Esta función guarda los cambios realizados al estado de la cita.
Nombre:	atenderCita(String urlAnterior, Cita cita)
Descripción:	Esta función muestra los datos de la cita entrada como parámetro para que pueda ser atendida.

Tabla 7. Descripción del Bean Manejado VerPlanificarCitaManejado.

Nombre: VerPlanificarCitaManejado	
Tipo de Clase: Controladora	
Atributo	Tipo
cita	Cita
horarioCitaDesde	Date
horarioCitaHasta	Date
Para cada responsabilidad	
Nombre:	replanificar (ActionEvent e)
Descripción:	Esta función muestra la página que permite replanificar la cita.
Nombre:	atender(ActionEvent e)
Descripción:	Esta función muestra la página que permite atender la cita.

Nombre:	verSolicitud(ActionEvent e)
Descripción:	Esta función muestra los datos de la tarjeta de cita.
Nombre:	verPersonaAsociada(ActionEvent e)
Descripción:	Esta función muestra los datos de la persona asociada a la tarjeta de cita.

2.4.4 Realizaciones de casos de uso

Las realizaciones de los casos de uso que se utilizaron en el Submódulo son los diagramas de contrato entre paquetes. Los mismos son un artefacto que se decidió generar como documentación del proyecto. Estos diagramas son una descripción de interacción entre subsistemas que se hacen cuando se tiene un esbozo de los subsistemas necesarios para realizar el caso de uso, describe cómo interactúan los objetos de las clases que contiene en el nivel de subsistema. Se hará mediante diagramas de secuencia que contenga las instancias de actores, subsistemas y transferencias de mensajes en que estos participan. (Jacabson, y otros, 2000)

Los diagramas de contrato entre paquetes para los casos de uso seleccionados quedarían de la siguiente manera:

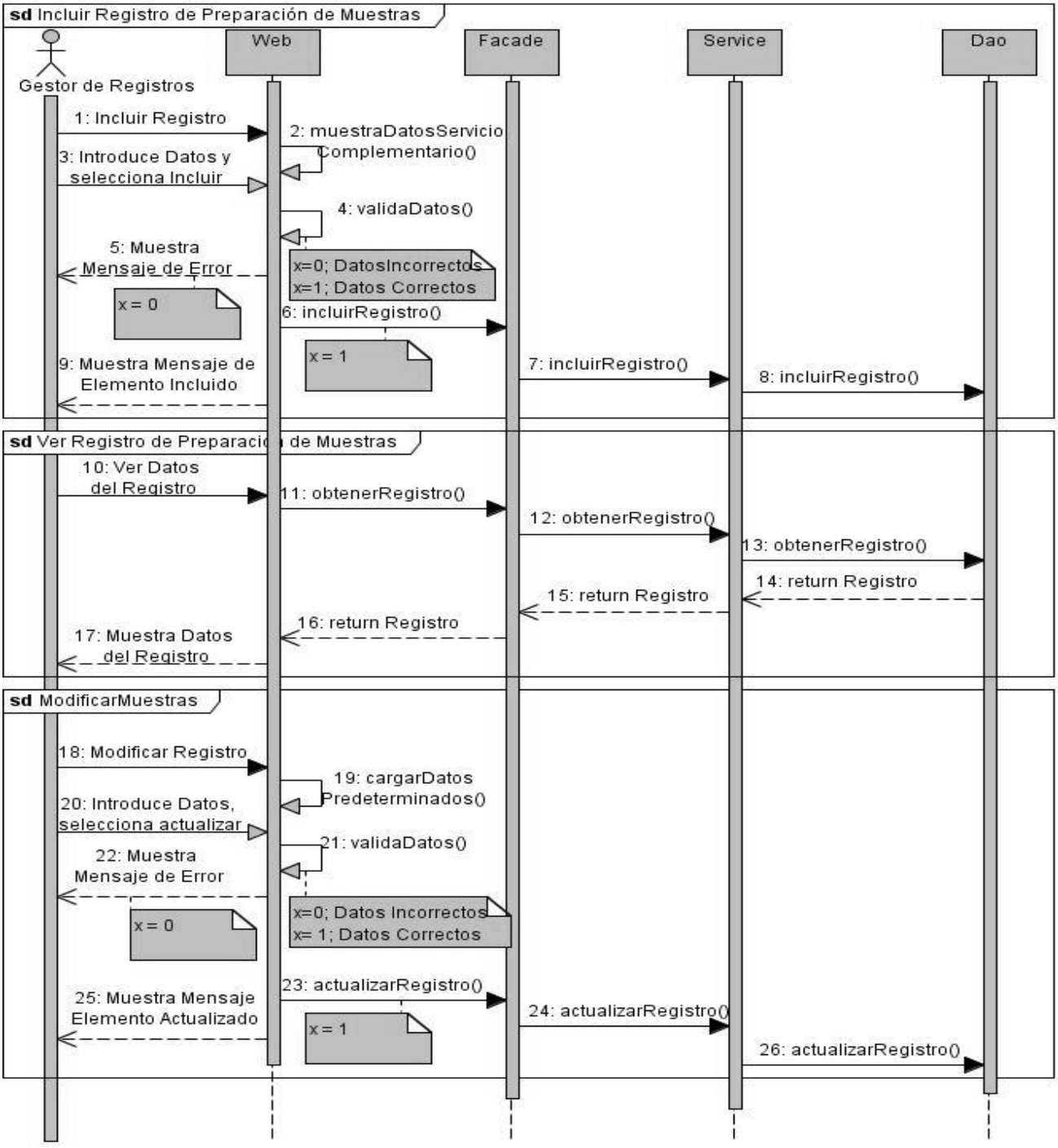

Figura 18. Diagrama de Contrato entre Paquetes. CU Gestionar Registro de Preparación de Muestras.

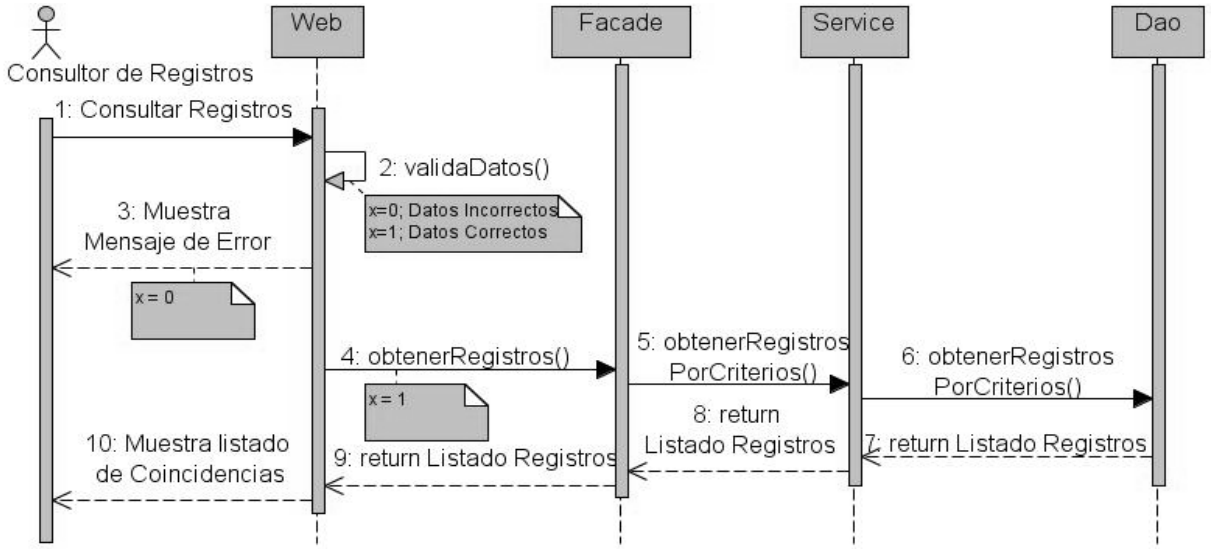

Figura 19. Diagrama de Contrato entre Paquetes. CU Consultar Registros de Preparación de Muestras.

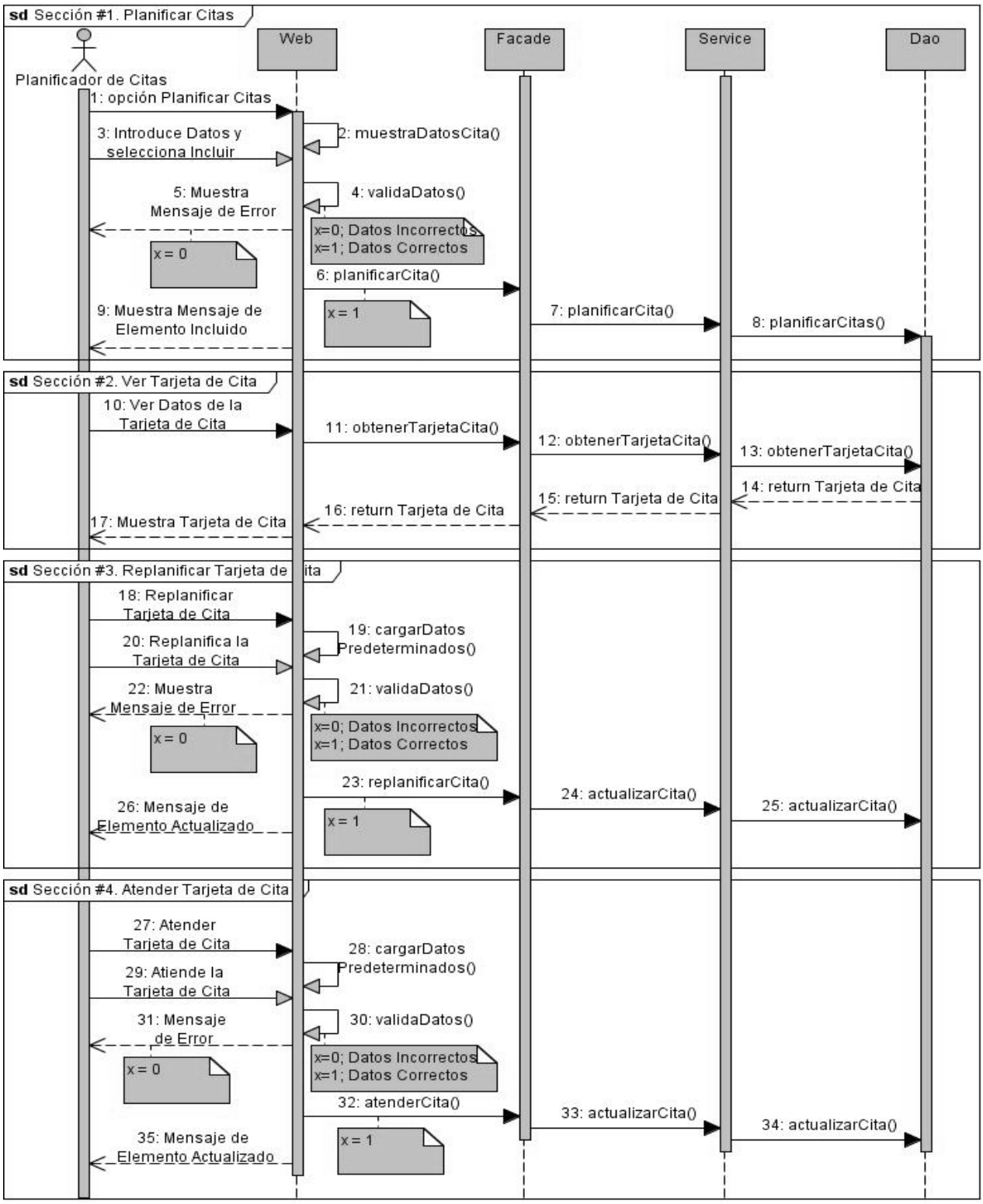

Figura 20. Diagrama de Contrato entre Paquetes. CU Planificar Cita. Sección #1 Incluir.

2.5 Modelo de Datos

Los modelos de datos aportan la base conceptual para diseñar aplicaciones que hacen un uso intensivo de datos. Un modelo de datos describe las representaciones lógicas y físicas de datos persistentes utilizados por la aplicación. Es usado para describir la representación lógica y física de la información persistente manejada por el sistema. Puede ser inicialmente creado a través de ingeniería inversa de un almacenamiento de datos persistentes (en la base de datos) o puede ser inicialmente creado a partir de un conjunto de clases del diseño persistentes en el modelo de diseño. (Carralero Medina, y otros, 2009)

2.5.1 Diagrama de Clases Persistentes

La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. A continuación se presenta el diagrama de clases que se determinaron como persistentes en el dominio del Submódulo Servicios:

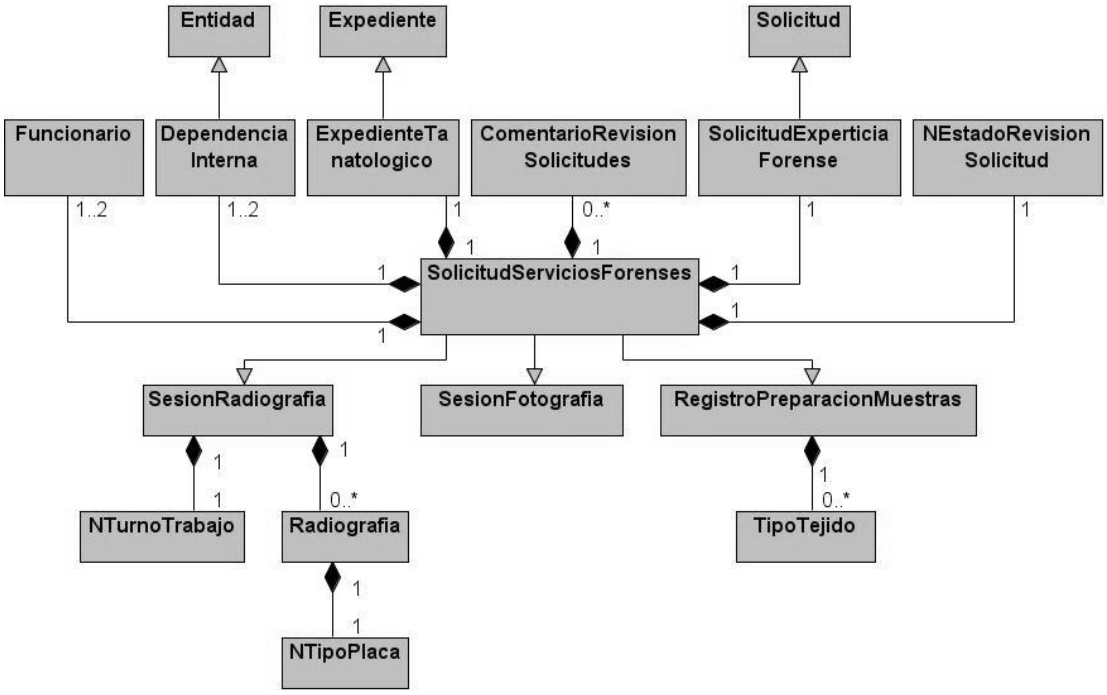

Figura 21. Diagrama de clases persistentes para el Submódulo Servicios.

2.6 Modelo de Implementación

En la implementación se parte del resultado del diseño y se implementa el sistema en término de componentes, el propósito principal de la implementación es desarrollar la arquitectura y el sistema como un todo. El modelo de implementación representa la composición física de la implementación en término de subsistemas de implementación, y elementos de implementación (directorios y archivos,

incluyendo código fuente, datos y archivos ejecutables). Además, define las principales unidades de integración alrededor de las cuales se organizan los equipos, así como las unidades que se pueden versionar, desplegar y reemplazar separadamente. (Carralero Medina, y otros, 2009)

2.6.1 Diagrama de Subsistemas de Implementación

Los subsistemas de implementación proporcionan una forma de organizar los artefactos del Modelo de Implementación y están estrechamente relacionados con los subsistemas obtenidos durante el flujo de Diseño.

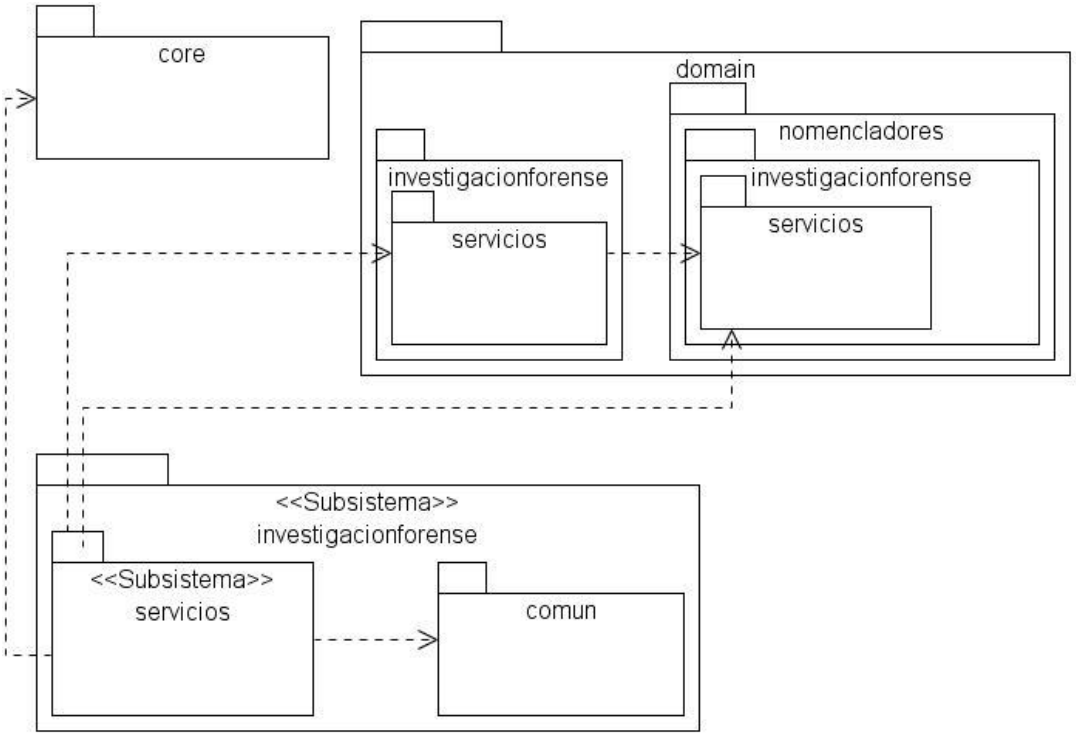

Figura 22. Diagrama de Subsistemas de Implementación.

2.6.2 Diagrama de Componentes

Los diagramas de componentes son usados para estructurar el modelo de implementación en término de Subsistemas de implementación y mostrar las relaciones entre los elementos de implementación.

El uso más importante de estos diagramas es mostrar la estructura de alto nivel del modelo de implementación, especificando:

- Muestra las organizaciones y las dependencias entre tipos de componentes.
- Organizar los subsistemas de implementación en capas.

También se utilizan para mostrar las dependencias de compilación de los ficheros de código, relaciones de derivación entre ficheros de código fuente y ficheros que son resultados de la

compilación, dependencias entre elementos de implementación y los correspondientes elementos de diseño que son implementados.

Se utilizan para modelar la vista estática de un sistema. Muestran la organización y las dependencias lógicas entre un conjunto de componentes de software, sean estos componentes de código fuente, librerías, binarios o ejecutables. (Rumbaugh, et al.)

A continuación se presentan los diagramas de componentes de los casos de uso seleccionados:

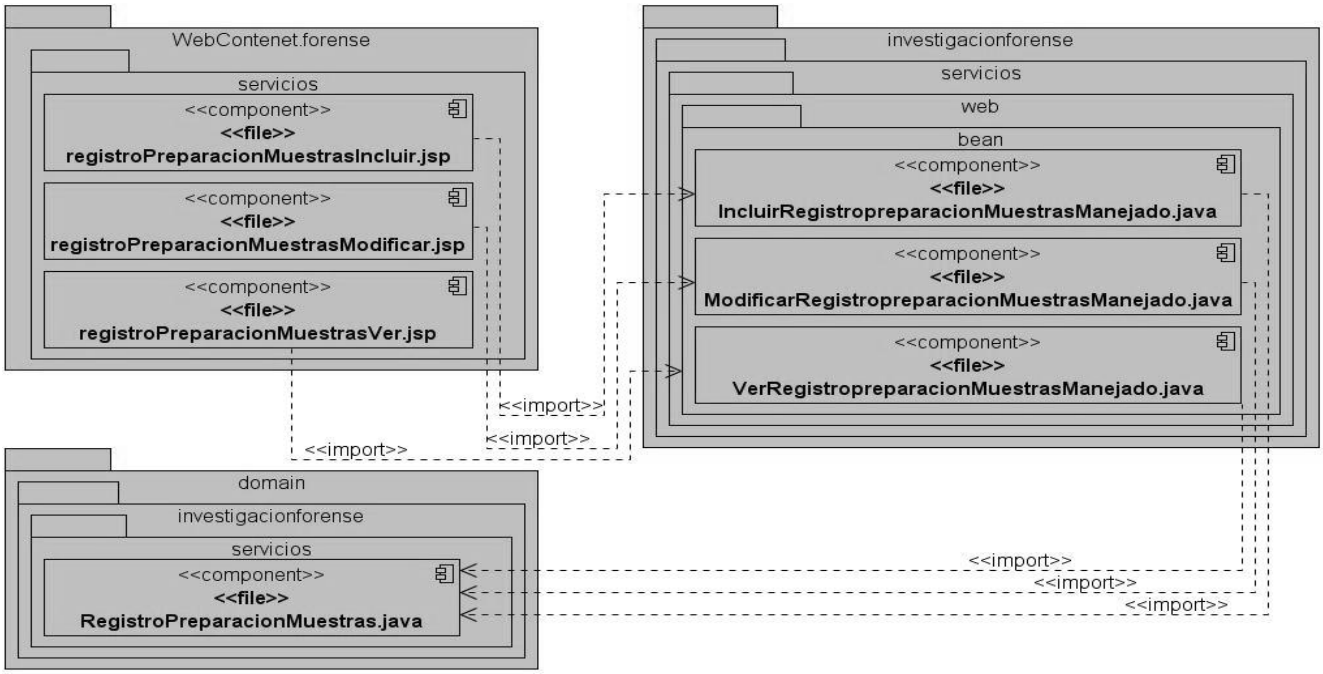

Figura 23. Diagrama de Componentes. CU Gestionar Registro de Preparación de Muestras.

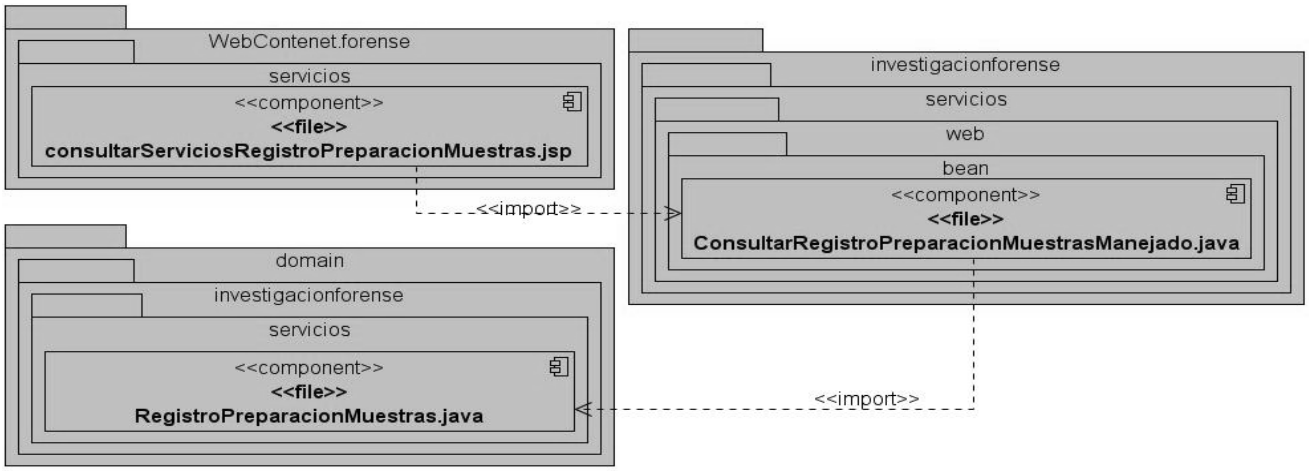

Figura 24. Diagrama de Componentes. CU Consultar Registros de Preparación de Muestras.

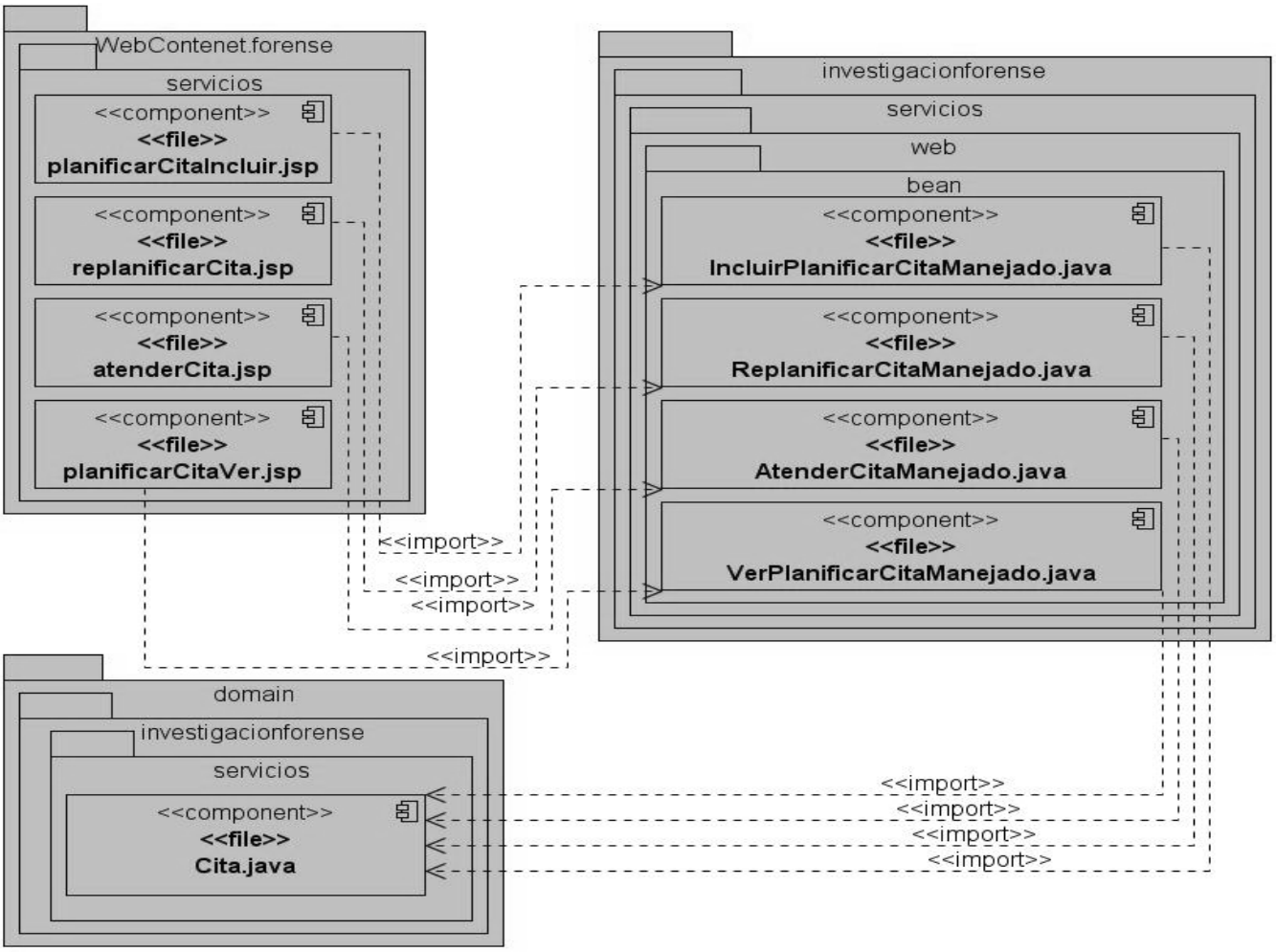

Figura 25. Diagrama de Componentes. CU Planificar Cita.

2.7 Conclusiones parciales

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, tarjetas CRC y diagramas de contrato entre paquetes.

En este capítulo se realizó el análisis y diseño del Submódulo Servicios, utilizando como principal fuente de información la descripción de los casos de uso de este, lo que posibilitó desarrollar de forma cómoda el modelo de implementación de la solución propuesta. Se generaron los artefactos que complementan el cumplimiento de los requisitos funcionales y no funcionales descritos para el sistema.

CAPÍTULO 3. VALIDACIÓN DE LA PROPUESTA DE SOLUCIÓN.

3.1 Introducción

Ya implementado el Submódulo Servicios fue sometido a diferentes pruebas realizadas por el equipo de calidad interna del proyecto CICPC para garantizar el cumplimiento de los requisitos funcionales y no funcionales asociados al mismo. En este capítulo se abordarán los tipos de pruebas realizadas, los resultados obtenidos en las mismas y la evaluación de esos resultados.

3.2 Tipos de prueba

Durante el flujo de trabajo de pruebas se confirmó el resultado de la implementación probando cada construcción, incluyendo tanto construcciones internas como intermedias, así como las versiones finales del sistema a ser entregadas a los clientes.

Dentro de los tipos de prueba definidos para la propuesta se encuentran las pruebas de Funcionalidad, Seguridad, Disponibilidad y Red, Rendimiento, Compatibilidad, Resistencia o Stress y Usabilidad. De los mismos, solo las pruebas de funcionalidad son las que se adaptan al rol. (Jacabson, y otros, 2000)

Estas pruebas tienen como objetivo comprobar la funcionalidad del sistema y validar las funciones, métodos, servicios y casos de uso.

Metas

Validar que la aplicación:

- Cumpla con los requisitos funcionales especificados en el diseño de la solución por medio de casos de uso.
- Cumpla con los requisitos no funcionales especificados en el diseño de la solución.
- Cumpla con las restricciones de entrada y salida de la información especificada en el diccionario de datos.
- Cumpla íntegramente con la estructura referencial especificada en el mapa de navegación.

3.3 Niveles de prueba

Pruebas Unitarias: Comienzan con la prueba de cada módulo. Una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de estos funcione correctamente por separado. El objetivo de las pruebas unitarias es aislar cada parte

del programa y mostrar que las partes individuales son correctas. Proporcionan un contrato escrito que el fragmento de código debe satisfacer. Estas pruebas aisladas proporcionan cinco ventajas básicas; fomentan el cambio, simplifican la integración, documentan el código, separan la interfaz del código y hacen que los errores estén más acotados y sean fáciles de localizar. (Jacabson, y otros, 2000)

Pruebas de Integración: A partir del esquema del diseño, los módulos probados se vuelven a probar combinados para probar sus interfaces. Pruebas integrales o pruebas de integración son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias. Únicamente se refieren a la prueba o pruebas de todos los elementos unitarios que componen un proceso, hecha en conjunto, de una sola vez. Consiste en realizar pruebas para verificar que un gran conjunto de partes de software funcionan juntos. (Jacabson, y otros, 2000)

Pruebas del Sistema: El software ensamblado totalmente con cualquier componente hardware que requiera, se prueba para comprobar que se cumplen los requisitos funcionales. Cualquier pieza de software completo, desarrollado o adquirido, puede verse como un sistema que debe probarse, ya sea para decidir acerca de su aceptación, para analizar defectos globales o para estudiar aspectos específicos de su comportamiento, tales como seguridad o rendimiento. Este tipo de pruebas estudia el producto completo. (Jacabson, y otros, 2000)

Pruebas de Aceptación: Estas pruebas las realiza el cliente. Son básicamente pruebas funcionales, sobre el sistema completo, y buscan una cobertura de la especificación de requisitos y del manual del usuario. Estas pruebas no se realizan durante el desarrollo, pues sería impresentable al cliente; sino que se realizan sobre el producto terminado e integrado. (Jacabson, y otros, 2000)

Una vez implementado el sistema fue sometido a los niveles de pruebas anteriormente mencionados. Entre estos se encuentran el nivel de pruebas unitarias, nivel de pruebas del sistema y nivel de pruebas de aceptación, los cuales ayudaron a la detección de los errores existentes.

3.4 Métodos de prueba

Pruebas de Caja Negra

Las pruebas de caja negra, también denominadas pruebas de comportamiento, se centran en los requisitos funcionales del software. O sea, permiten al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Estas pruebas no son una alternativa a las técnicas de pruebas de caja blanca, más bien se trata de un

enfoque complementario que intenta descubrir diferentes tipos de errores que se llevan a cabo sobre la interfaz del software. El objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene, saber qué es lo que hace el software pero sin entrar en detalles de código, es decir, que es lo que hace, y no cómo lo hace (no se ve el código). Estas pruebas se centran principalmente en los requisitos funcionales del software y permiten encontrar: (Pressman, 2002)

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

Pruebas de Caja Blanca

Las pruebas de caja blanca, denominadas pruebas 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. (Pressman, 2002)

Se comprueban los caminos lógicos del software. Examina el programa en varios puntos para determinar si el estado real coincide con el esperado (sobre el código). Así como las pruebas de caja negra ejercitan los requisitos funcionales desde el exterior del módulo, las de caja blanca están dirigidas a las funciones internas. Las pruebas de caja blanca se llevan a cabo en primer lugar, sobre un módulo concreto, para luego realizar las de caja negra sobre varios subsistemas.

Este método de prueba requiere el conocimiento de la estructura interna del programa y se debe garantizar como mínimo que: (Pressman, 2002)

- Se ejerciten por lo menos una vez todos los caminos independientes para cada módulo.
- Se ejerciten todas las decisiones lógicas en sus variantes verdaderas y falsas.
- Se ejerciten las estructuras internas de datos para asegurar su validez.

Con la realización de las pruebas se pretende demostrar que hay errores, en ningún momento es posible asegurar lo contrario. El principal objetivo al diseñar las pruebas es tratar de ejecutar todos los flujos posibles, para descubrir la mayor cantidad de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo.

3.5 Resultados Obtenidos.

Durante el proceso de desarrollo del Submódulo se llevaron a cabo pruebas unitarias para ir comprobando el funcionamiento del avance de la solución. Estas pruebas se enmarcan dentro del nivel de prueba de unidad, utilizando el método de caja blanca. Fueron realizadas por los desarrolladores aprovechando las ventajas de compilación paso a paso que brinda el IDE de desarrollo. Estas no se planificaron ni se registraron sus resultados, fueron haciéndose a medida que la solución se desarrollaba.

Para evaluar la solución desarrollada se concibieron varias iteraciones donde se estuvo probando el software íntegramente. Se implicaron los métodos y niveles de prueba expuestos anteriormente en cada una de las iteraciones, en las cuales están implicadas prácticamente todas las personas del proyecto, desde el equipo de calidad hasta el cliente final, proyectando resultados visibles.

A continuación se especifican las principales pruebas realizadas a la aplicación, las tres primeras pertenecientes al nivel de prueba de sistema y las dos últimas al nivel de aceptación, en todos los casos haciendo uso del método de caja negra.

Pruebas Internas: Son realizadas por el equipo de calidad interna del proyecto con el fin de entregar un producto con la menor cantidad de errores posibles. Se centraron en el cumplimiento de las funcionalidades descritas en el listado de requerimientos y de casos de uso.

Pruebas Cruzadas: Pruebas realizadas al sistema por el resto de los equipos de desarrollo del proyecto. Se realizaron con el fin de encontrar la mayor cantidad de errores posibles en término de validaciones, pautas definidas por arquitectura de información, formato de los campos, entre otras.

Pruebas de Liberación: Pruebas realizadas por un tercero, en este caso Calisoft, institución encargada de validar que el software cuente con la calidad requerida para ser entregado a los clientes finales.

Pruebas de Aceptación: Pruebas realizadas por los clientes para comprobar que el sistema hace lo que ellos esperan y necesitan.

Pruebas Piloto: Pruebas realizadas por los clientes para comprobar el rendimiento de la aplicación y su respuesta un entorno real de trabajo, así como el cumplimiento de las funcionalidades requeridas.

Dada su importancia, se muestran los resultados obtenidos durante las pruebas de liberación, aceptación y pruebas piloto.

Pruebas de Liberación:

	CU	PC	NC			
			Alta	Media	Baja	Total
SIIPOL	428	0	364	219	255	838
Servicios	9	0	4	2	1	7

CU: Casos de Uso **PC**: Pedidos de Cambios **NC**: No Conformidades

Pruebas de Aceptación 1ra Iteración:

	CU	PC	NC			
			Alta	Media	Baja	Total
SIIPOL	428	35	16	19	18	50
Servicios	9	0	0	0	0	0

CU: Casos de Uso **PC**: Pedidos de Cambios **NC**: No Conformidades

Pruebas de Aceptación 2da Iteración:

	CU	PC	NC			
			Alta	Media	Baja	Total
SIIPOL	428	0	2	4	2	8
Servicios	9	0	0	0	0	0

CU: Casos de Uso **PC**: Pedidos de Cambios **NC**: No Conformidades

Pruebas Piloto:

	CU	PC	NC			
			Alta	Media	Baja	Total
SIIPOL	428	69	28	25	63	116
Servicios	9	1	1	0	0	1

CU: Casos de Uso **PC**: Pedidos de Cambios **NC**: No Conformidades

En los siguientes gráficos se muestran los resultados obtenidos en las principales pruebas realizadas al software.

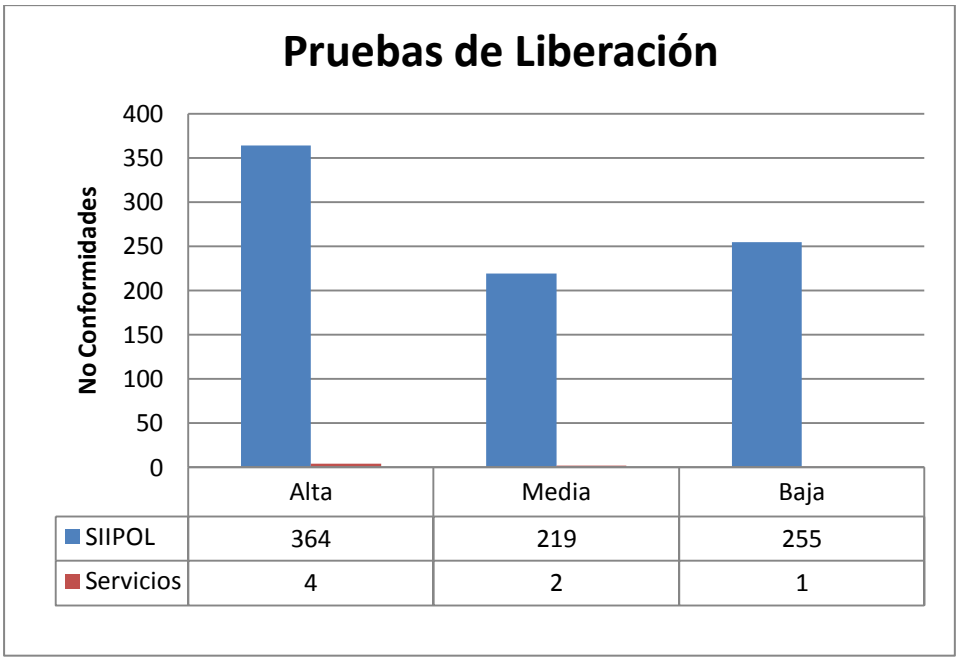

Figura 26. Pruebas de Liberación.

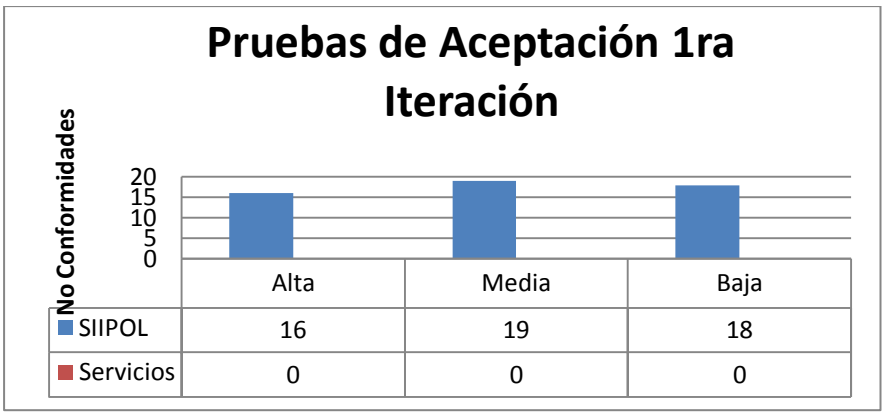

Figura 27. Pruebas de Aceptación 1ra Iteración.

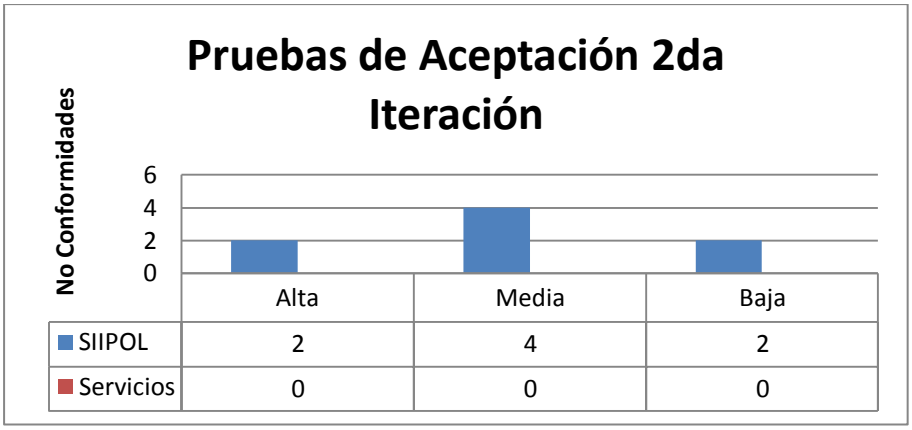

Figura 28. Pruebas de Aceptación 2da Iteración.

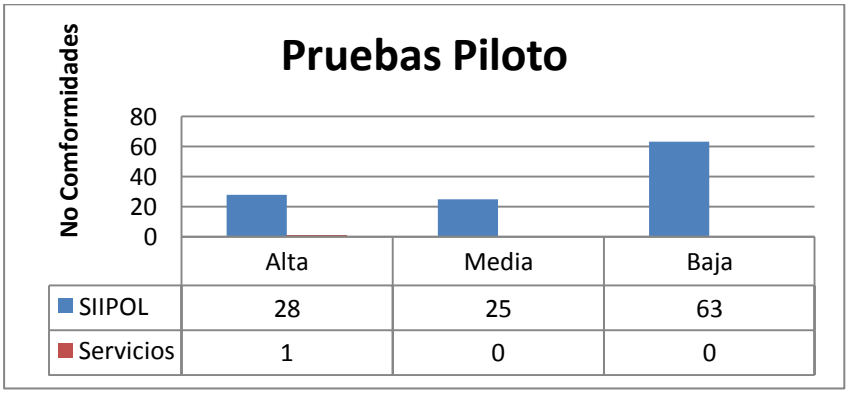

Figura 29. Pruebas Piloto.

Los datos mostrados anteriormente evidencian la estabilidad del sistema durante las pruebas realizadas, teniendo en cuenta la cantidad de casos de uso presentados en cada una de las mismas y su relación con las cantidades de no conformidades obtenidas.

Específicamente, el Submódulo Servicios, presentó un número muy pequeño de problemas en relación con la cantidad de casos de uso que lo conforman y con la cantidad de no conformidades por casos de uso detectadas al sistema de manera general. El Submódulo obtuvo muy buena aceptación por parte del cliente, todas las no conformidades fueron resueltas así como el pedido de cambio solicitado, presentando en cada iteración de prueba un sistema más sólido y funcional.

3.6 Análisis comparativo de los procesos de servicios y las funcionalidades implementadas.

Durante la fase de inicio se identificaron los procesos de servicios que intervienen en la CNCF, se detectaron fallas en cuanto a la fluidez de información, control de recursos y materiales utilizados en los diferentes estudios y se obtuvo como resultado final los casos de uso que guiaron la implementación del Submódulo Servicios. Estas funcionalidades fueron claramente definidas y necesarias a incluir en el sistema sin omitir ninguno de los procesos vigentes. Dichas funcionalidades fueron aprobadas y firmadas por los clientes.

En el Sistema Integrado de Información Policial ninguno de estos procesos se encontraba automatizado por lo que el trabajo debía efectuarse de forma manual o verbal. Una vez automatizados, incorporados al SIIPOL y validados por las pruebas realizadas por el cliente, se determinó que la informatización de estos procesos dio un gran paso de avance, pues mejoró la forma en que se realizaban.

Existen resultados que demuestran la mejora en dichos procesos de servicios como son:

- Las solicitudes de servicios forenses pueden ser confirmadas por los especialistas forenses.
- Los recursos destinados a los estudios forenses una vez utilizados quedan registrados, manteniendo el control de los mismos.
- Los especialistas pueden planificar sus citas sin que haya contradicciones con otras citas planificadas anteriormente.

Con la informatización del Submódulo Servicios se logró que los clientes quedaran satisfechos por las ventajas mencionadas anteriormente, proporcionando un apoyo a los grandes procesos de la CNCF.

3.7 Conclusiones parciales.

Con la realización de este capítulo se comprobó la validez de la propuesta de solución mediante las pruebas realizadas. El Submódulo Servicios cumple las expectativas de los clientes al aportarles numerosas ventajas como el ahorro de trámites innecesarios y el mantenimiento del control de los medios y recursos destinados a los estudios forenses, así como la constancia del cumplimiento de las actividades realizadas por el personal técnico.

CONCLUSIONES GENERALES.

Durante el desarrollo del presente trabajo se analizaron diferentes cuestiones que aportaron conocimientos teóricos sobre los Sistemas de Gestión de Información e Información Policial. El estudio de sistemas similares demostró lo importante que resultan estos sistemas para el buen desempeño de las actividades policiales.

Se describieron las herramientas, lenguajes y metodología empleados en la solución, los cuales poseen características que agilizan y facilitan el proceso de desarrollo de software. Las herramientas ayudaron a realizar los modelos y diagramas útiles para la implementación de la propuesta de solución, el lenguaje UML facilitó la realización de los modelos y el lenguaje de programación proporciono la codificación, la metodología registro y documento todo lo relacionado con la solución.

El diseño del Submódulo Servicios dotó al desarrollador de una vista lógica del sistema y los elementos que componen el mismo, lo que posibilitó la implementación del Submódulo Servicios respetando las pautas definidas, lo cual contribuyó a la obtención de un código limpio y flexible a futuros cambios.

Una vez integrado el Submódulo Servicio al sistema SIIPOL, luego de las pruebas de calidad aplicadas para la detección y corrección de errores existentes, (tanto por expertos en calidad como por los clientes), se demostró su validez, el cumplimiento de las características requeridas y de los objetivos propuestos.

RECOMENDACIONES.

Adicionar una funcionalidad que permita utilizar el sistema de notificaciones del SIIPOL, aplicados al proceso de Planificación de Citas del Submódulo Servicios, el cual permita notificarle al funcionario encargado de la cita conocer el estado en que se encuentra la misma.

Agregar una funcionalidad que permita al funcionario responsable de atender una cita, notificar al planificador de la misma en caso de tener algún inconveniente para realizar la actividad. Esto posibilitaría una temprana replanificación de la cita.

Realizar una refactorización al Submódulo, para lograr una aplicación mucho más funcional y optimizar su rendimiento.

BIBLIOGRAFÍA

ADMINISTRACIÓN PÚBLICA DEL ESTADO DE QUINTANA ROO. 2005. www.cemer.qroo.gob.mx. *MANUAL DE POLÍTICAS PARA EL USO DE LAS TECNOLOGÍAS DE LA INFORMACIÓN Y TELECOMUNICACIONES DEL INSTITUTO DEL PATRIMONIO INMOBILIARIO DE LA ADMINISTRACIÓN PÚBLICA DEL ESTADO DE QUINTANA ROO.* [Online] 2005. [Cited: marzo 22, 2010.] http://www.cemer.qroo.gob.mx/.../Manual%20de%20Politicas%20UTIC_IPAE.doc.

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.

—. 2003. *Introducción a Hibernate.* 2003.

ALEGSA. 2010. ALEGSA.COM. *Diccionario de informática.* [Online] 2010. [Cited: enero 8, 2010.] http://www.alegsa.com.ar/Dic/interfaz.php.

Alvarez, Miguel Angel. 2001. DesarrolloWeb.com. [Online] DesarrolloWeb.com, julio 18, 2001. [Cited: 2 12, 2010.] http://www.desarrolloweb.com/articulos/497.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.

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

Baver, Christian and King, Gavin. 2005. *Java Persistence with Hibernate: REVISED EDITION OF HIBERNATE IN ACTION.* New York : Manning Publications Co, 2005. ISBN 1-932394-88-5.

Bogss, Wendy y Bogss, Michael. 2002. *Mastering UML with Rational Rose.* 2002. 0−7821−4017−3.

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.

Carralero Medina, Rosalba y Pupo Ortiz, Carlos. 2009. *Trabajo de Diploma: Análisis, Diseño e Implementación de la Capa de Presentación del submódulo Cadáveres del módulo Investigación Forense del Sistema de Investigación e Información Policial (SIIPOL).* Ciudad de la Habana : s.n., 2009. s.n..

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.

Cuerpo de Investigaciones Científicas Penales y Criminalísticas. [Online] [Cited: abril 15, 2009.] http://www.cicpc.gov.ve/.

cyta.com. Herramientas CASE. *Ciencia y Técnica Administrativa.* [Online] [Cited: diciembre 15, 2009.] http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm.

D.D.I.Software. 2006. *Conferencia 5.Fase de Inicio. Flujo de Análisis y Diseño. Modelo de.* 2006.

desarrolloweb.com. desarrolloweb.com. *www.desarrolloweb.com/.* [Online] [Cited: abril 2010, 6.] http://www.desarrolloweb.com/articulos/497.php.

Diagramas UML. *Laboratorio de Televisión Digital Interactiva.* [Online] [Cited: Marzo 22, 2009.] http://tvdi.det.uvigo.es/~avilas/UML/node22.html.

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

Dirección Nacional de Servicios Penitenciarios de Venezuela. 2007. Ministerio del Poder Popular para Relaciones Interiores y Justicia. Dirección Nacional de Servicios Penitenciarios. [Online] Dirección de Informática. Dirección Nacional de Servicios Penitenciarios. *www.dnsp.gob.ve.* [Online] 2007. [Cited: febrero 24, 2010.] http://www.dnsp.gob.ve/?q=node/159.

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

eLABORO, ingeniería del software S.L. 2008. EPATROL. *www.epatrol.es.* [En línea] 2008. [Citado el: 3 de febrero de 2010.] http://www.epatrol.es/ventajas/index.html.

Entorno Virtual de Aprendizaje. 2009. Conferencia 1.Disciplina de Análisis y Diseño. *Entorno Virtual de Aprendizaje.* [Online] UCI, 2009. [Cited: Febrero 25, 2010.] http://eva.uci.cu/mod/resource/view.php?id=14069/Conferencia #1.

—. 2009. Conferencia 7 Fase de Elaboración. Flujo de trabajo de Análisis y Diseño. *Entorno Virtual de Aprendizaje.* [En línea] 2009. [Citado el: 16 de 1 de 2010.] http://eva.uci.cu/mod/resource/view.php?id=22668/Conferencia 7.

Evaluación de Software. *http://teleformacion.uci.cu.* [Online] [Cited: Marzo 24, 2009.] http://teleformacion.uci.cu/mod/resource/view.php?id=14103.

Exadel. 2009. Exadel Studio Pro. *Exadel.* [Online] 2009. [Cited: febrero 20, 2010.] http://exadel.com/web/portal/products/ExadelStudioPro.

Exadel Studio Pro. *Exadel.* [Online] [Cited: Enero 20, 2009.] http://exadel.com/web/portal/products/ExadelStudioPro.

Fase de Elaboración. Flujo de trabajo de Análisis y Diseño. *http://teleformacion.uci.cu.* [Online] [Cited: Febrero 4, 2010.] mms://ucimedia.uci.cu/teleclases/1er_Semestre/3er/Ingenieria_de_Software_I/conf5. s.n..

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.

Flujo de Implementación. *http://teleformacion.uci.cu.* [Online] [Cited: marzo 1, 2010.] mms://ucimedia.uci.cu/teleclases/2005-2006/2do-sem/3er/ingenieria_de_software_2/conf2. s.n..

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.

Grupo Soluciones Innova S.A. Grupo de Soluciones Innova. [En línea] [Citado el: 10 de marzo de 2010.] http://www.rational.com.ar/herramientas/roseenterprise.html.

Guevara, Ing. Humberto Rivero. 2008. *Trabajo de Diploma:Analisis, Diseño e Implementación del Modulo Aprehensión del SIIPOL.* Cuidad de la Habana : UCI, 2008.

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

Herramientas CASE. *Ciencia y Técnica Administrativa.* [Online] [Cited: Enero 25, 2009.] http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm.

http://java.sun.com. *http://java.sun.com.* [Online] [Cited: diciembre 10, 2009.] http://java.sun.com/applets/.

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

Ibarra, Armando F. Rational Unified Process. *U-Cursos.* [Online] [Cited: Enero 15, 2009.] https://www.u-cursos.cl/ingenieria/2004/2/CC62C/1/material_docente/objeto/44990.

Iborra, Ignacio. 2004. Especialista Universitario Java Enterprise. *Especialista Universitario Java Enterprise.* [En línea] 2004. [Citado el: 12 de abril de 2010.] http://www.jtech.ua.es/tutoriales/apuntes/sesion-junit-apuntes.htm.

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.

Ignside. 2007. www.ignside.net. *Telnet, introducción.* [Online] 01 14, 2007. [Cited: marzo 24, 2010.] http://www.ignside.net/man/telnet/.

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.

Information Management. 2006. Information Management, Reflexiones sobre las tecnologías de la información. *http://informationmanagement.wordpress.com.* [Online] Noviembre 28, 2006. [Cited: enero 23, 2010.] http://informationmanagement.wordpress.com/category/gestion/gestion-de-la-informacion/.

Interfaz de usuario. *Monografías.com.* [Online] [Cited: Abril 5, 2009.] http://www.monografias.com/trabajos6/inus/inus.shtml.

Introducción a la Ingeniería de Software. *teleformacion.uci.cu.* [Online] [Cited: enero 20, 2010.] mms://ucimedia.uci.cu/teleclases/1er_Semestre/3er/Ingenieria_de_Software_I/conf1.

Introducción a la Tecnología Java Server Faces. *Java en Castellano.* [En línea] [Citado el: 3 de febrero de 2009.] http://www.programacion.com/java/tutorial/jsf_intro/1/.

Jacabson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El proceso Unificado de Desarrollo de Software.* Madrid : Addison Wesley, 2000. 84-7829-036-2.

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.

Java Comunity Process. *jcp.org.* [En línea] [Citado el: 30 de enero de 2010.] http://jcp.org/en/jsr/overview.

Java en castellano. 1999. www.programacion.com. *Introducción a la Tecnología JavaServer Faces.* [Online] 1999. http://www.programacion.com/java/tutorial/jsf_intro/.

2002. JavaHispano. *www.javahispano.org.* [En línea] 2002. [Citado el: 15 de diciembre de 2009.] http://www.javahispano.org/contenidos/es/aplicaciones_de_escritorio_eficientes.

JBoss Seam. *JBoss.* [En línea] [Citado el: 10 de febrero de 2010.] http://www.jboss.com/products/seam.

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.

JUnit.org Resources for Test Driven Development. [Online] [Cited: Septiembre 24, 2008.] http://www.junit.org.

Kubuntu-es. www.kubuntu-es.org. *Glosario del Software Libre.* [Online] [Cited: marzo 10, 2010.] http://www.kubuntu-es.org/wiki/glosario-software-libre.

La Flecha, tu diario de Ciencia y Tecnología. www.laflecha.net. [Online] [Cited: marzo 10, 2010.] http://www.laflecha.net/canales/moviles/noticias/200409061.

La Flecha: tu diario de Ciencia y Tecnología. www.laflecha.net. *Red Hat presenta "Developer Studio" basado en Eclipse.* [Online] [Cited: febrero 22, 2010.] http://www.laflecha.net/canales/softlibre/red-hat-presenta-developer-studio-basado-en-eclipse/.

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. 1999. *UML y Patrones.* Mexico : Prentice Hall, 1999.

—. 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.

Lenguajes de Programación,2009. Lenguajes de Programación. *www.lenguajes-de-programacion.com.* [Online] [Cited: abril 2010, 5.] http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.

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.

LuAuF. 2008. LuAuF. *http://luauf.com.* [Online] 2008. [Cited: 3 22, 2010.] http://luauf.com/2008/05/13/entornos-de-desarrollo-integrado-para-java/.

Mapas Digitales S.A. Dmapas. *STEGPOL. SEGURIDAD CIUDADANA Y GESTIÓN POLICIAL.* [Online] [Cited: febrero 24, 2010.] http://www.dmapas.com/productos_stegpol.htm.

Masadelante.com. 1999. Masadelante.com. *Masadelante.com.* [Online] 1999. [Cited: abril 2010, 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.

Microsoft Solutions Framework. *MSDN Estudiantes.* [En línea] Microsoft. [Citado el: 13 de diciembre de 2009.] http://www.microsoft.com/spanish/MSDN/estudiantes/ingsoft/planificacion/msf.mspx.

2009. Ministerio del Poder Popular para la Comunicacción y la Información. [Online] 2009. [Cited: Enero 12, 2009.] http://www.minci.gob.ve/entrevistas/3/181453/se_redujo_violencia.html.

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.* [Online] 2002. [Cited: marzo 1, 2010.] http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html.

ORACLE: Sun Developer Network (SDN). Java 2 Platform, Enterprise Edition (J2EE) Overview. *Sun Developer Network (SDN).* [En línea] Oracle. [Citado el: 20 de febrero de 2010.] http://java.sun.com/j2ee/overview.html.

2009. Overview. SUN MicroSystems. *Java 2 Platform, Enterprise Edition(J2EE).* [Online] Enero 2009. http://java.sun.com/j2ee/overview.html.

policiales.net. Criminalística. *policiales.net.* [En línea] [Citado el: 23 de enero de 2010.] http://policiales.net/CRIMINALISTICA/CRIMINALISTICA-01.htm.

Pressman, Roger S. 2002. *Ingenirría del Software. Un enfoque práctico.* España : McGraw-Hill, 2002.

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

Productos & Servicios. [Online] Mapas Digitales S.A. [Cited: Septiembre 28, 2008.] http://www.dmapas.com/productos_stegpol.htm.

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

2009. Red Hat Developer Studio. *sitio Web de Red Hat Developer Studio.* [Online] 2009. http://www.redhat.com.

Red Hat. www.Redhat.com. [Online] [Cited: enero 15, 2010.] https://www.redhat.com/apps/store/developers/jboss_developer_studio.html.

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.

Rodríguez Ramos, Yadiel. 2009. *Trabajo de Diploma para optar por el título de Máster en Informática Aplicada: Sistema de Apoyo a la Investigación Criminalística para el Cuerpo de Investigaciones Científicas, Penales y Criminalísticas de Venezuela.* La Habana : s.n., 2009.

Roselló Nuñez, Reynaldo. 2007. Ingeniería de Requerimientos del Proceso de Investigación Forense del CICPC. *Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.* Ciudad de la Habana, Ciudad de la Habana, Cuba : s.n., julio de 2007. s.n.

—. 2007. *Trabajo de Diploma: Ingeniería de Requerimientos.* Ciudad de la Habana : s.n., 2007. s.n..

Rumbaugh, James, Jacobson, Ivar and Booch, Grandy. *El lenguaje unificado de modelado. Manual de referencia.* s.l. : Addison Wesley.

Secretaria de Seguridad Pública del Distrito Federal. SSP-DF. Sistema de Información Policial. *portal.ssp.df.gob.mx.* [Online] [Cited: febrero 10, 2010.] http://portal.ssp.df.gob.mx/portal.

2009. Sitio Oficial JAVA. [Online] 2009. http://java.com/es/.

Software Libre. *GNU Operating System.* [Online] [Cited: Marzo 15, 2009.] http://www.gnu.org/philosophy/free-sw.es.html.

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

Sparx. [Online] [Cited: Febrero 27, 2009.] http://www.sparxsystems.com.ar/.

SpringSource. 2010. Community Spring Source. *www.springframework.org/documentation.* [En línea] 2010. [Citado el: 19 de febrero de 2010.] http://www.theserverside.com/tt/articles/article.tss?l=SpringFramework.

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

Subsistema de Implementación . *Me Rinde.* [Online] [Cited: Marzo 23, 2009.] http://merinde.rinde.gob.ve/.

Terra. 2010. www.terra.es. *El emulador de máquinas recreativas más famoso.* [Online] 01 21, 2010. [Cited: marzo 24, 2010.] http://www.terra.es/tecnologia/articulo/html/tec10369.htm.

Universidad de Oviedo. 2003. Sitio Web de la E.U de Ingeniería Técnica Informática de Oviedo. *http://petra.euitio.uniovi.es/.* [En línea] 2003. [Citado el: 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 for UML. User's Guide (Part 1).

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/.

Visual-Paradigm for UML. *Visual-Paradigm.* [Online] [Cited: Febrero 10, 2009.] http://www.visual-paradigm.com/product/vpuml/.

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

Walls, Craig. 2008. *Spring in Action.* Greenwick : Manning, 2008. ISBN 1-933988-13-4.

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

Wayne Bartle, Philip Francis. 2009. Potenciación comunitaria. *www.scn.org.* [Online] 09 11, 2009. [Cited: 01 25, 2010.] http://www.scn.org/mpfc/modules/mon-miss.htm.

—. 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.

Web oficial Visual-Paradigm. Sitio Web oficial Visual-Paradigm. *Sitio Web oficial Visual-Paradigm.* [Online] [Cited: febrero 25, 2010.] http://www.visual-paradigm.com/product/vpuml/..

Yang Shen, Derek. Integración de JSF, Spring e Hibernate para crear una Aplicación Web del Mundo Real. *Programacion en Castellano.* [En línea] [Citado el: 10 de diciembre de 2009.] http://www.programacion.net/tutorial/jap_jsfwork/3/#34_logica.

—. 2010. Integración de JSF, Spring e Hibernate para crear una Aplicación Web del Mundo Real. *Programación en castellano.* [Online] 2010. [Cited: febrero 5, 2010.] http://www.programacion.net/tutorial/jap_jsfwork/3/.

Young, Mike y Young, Curtis W. 2003. *Deploying Solutions with .NET Enterprise Servers.* Canada : Wiley, 2003.

GLOSARIO DE TÉRMINOS.

A continuación, en orden alfabético, se muestra el significado de algunos términos usados en este documento que pueden dificultar la compresión del mismo:

Adabas-Natural: Es una base de datos jerárquica de alto rendimiento creada por la empresa alemana Software AG, en el año 1969. Actualmente se sigue comercializando bajo la versión Adabas 2006.

Ajax4JSF: Es una extensión Open Source del estándar JSF que se integra con este con gran facilidad. Brinda una amplia gama de componentes con capacidades AJAX (del término en inglés Asincronous Java Script and XML) incluso a aplicaciones JSF ya creadas efectuando sencillas sustituciones de componentes. La integración de JSF con AJAX, provee la aplicación de un efecto “aplicación de escritorio”, además de hacerla más profesional y agradable al usuario.

AOP: Aspect-Oriented Programming, traducido al español: Programación Orientada a Aspectos, paradigma de programación que separa la lógica del negocio de aspectos de servicios, tales como la seguridad, las transacciones, entre otros.

Apache TomCat: Servidor web con soporte de servlet y JSP. Incluye el compilador Jasper, que compila las páginas JSP convirtiéndolas en servlet.

API: Conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Bean de respaldo: Componente de JavaBeans que corresponde a una página JSP que incluye componentes de Java Server Faces. Define las propiedades de los componentes de la página y los métodos que realizan el procesamiento del componente.

Bean: En el lenguaje Java es un componente software reutilizable. Existe con la finalidad de ahorrar tiempo al programar.

Bug: Un defecto de software, es el resultado de un fallo o deficiencia durante el proceso de creación de algún algoritmo o línea de codificación.

Componentes: Clase abstracta o librería de clases que puede representar todo lo que tiene una posición, un tamaño, puede ser pintado en pantalla y recibir eventos.

CSV: Los ficheros CSV (del inglés comma-separated values) son un tipo de documento sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas y las filas por saltos de línea.

CVS: El CVS (del inglés Concurrent Versions System) es una aplicación informática que implementa un sistema de control de versiones: mantiene el registro de todo el trabajo y los cambios en los ficheros que conforman un proyecto.

DAO: El DAO (del inglés Data Access Object) es un Patrón de diseño de clases en ingeniería de software que permite a quien lo aplique suministrar una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos.

Inyección de Dependencia: Es una manera de lograr la inversión de control, utilizada por Spring para manipular las dependencias entre los objetos.

Emulador: Software que permite ejecutar programas de computadora o videojuegos en una plataforma diferente de aquella para la cual fueron escritos originalmente.

Enterprise Java Bean: Es una de las API que forman parte del estándar de construcción de aplicaciones empresariales J2EE.

Evidencia: Objeto involucrado en una falta disciplinaria o investigación penal, que aporta información sobre la forma o motivo en que ocurrió el hecho.

Experticia: Informe Técnico que contiene los estudios realizados a determinada solicitud.

HQL: Lenguaje de consultas del framework Hibernate similar al SQL, pero que se refiere a clases y objetos no a tablas de la base de datos.

IDE: Entorno Integrado de Desarrollo, es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse exclusivamente para un lenguaje de programación o bien para varios.

Informe Pericial: En este informe se especifica todo lo referente al resultado de la experticia realizada.

IoC: Inversión del control, técnica de programación utilizada para manejar las dependencias entre objetos.

Listener: Clase, registrada con un objeto de publicación, que informa de los procedimientos que deben seguirse cuando se produce un evento.

Mapeo: Técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional.

Open Source: Representa el software de dominio público, esto significa sin licencia, cuyo código fuente está disponible y se le permite usar y modificar.

ORM: Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos.

Plug-ins: Módulo que puede anexarse a otro para aumentar sus funcionalidades (generalmente sin afectar otras funciones ni afectar la aplicación principal).

RichFaces: Es una rica librería de componentes que añaden capacidades AJAX a las páginas. Inicialmente, como proyecto propietario, RichFaces se distribuía de manera independiente a pesar de que estaba implementado sobre una capa de Ajax4JSF. Al transformarse en Open Source, en la actualidad RichFaces y Ajax4JSF se distribuyen de manera conjunta.

Sesión: Secuencia de interacciones entre un usuario y una o varias aplicaciones, que comienza al iniciar una sesión y finaliza al cerrar la sesión o cuando se agota el tiempo de espera.

Telnet: Es una de las formas de conectarse a otra computadora por medio de llamada directa del ordenador terminal al ordenador host y ejecutar secuencias de comandos, al estilo DOS o UNIX.

Transacciones: Conjunto de órdenes que se ejecutan formando una unidad de trabajo, es decir, en forma indivisible o atómica, posibilitando la integridad de los datos.

XML: Metadata Interchange (XML de Intercambio de Metadatos) es una especificación que permite compartir diagramas entre diferentes herramientas de modelado UML.

ANEXOS.

Anexo I Modelo de Análisis

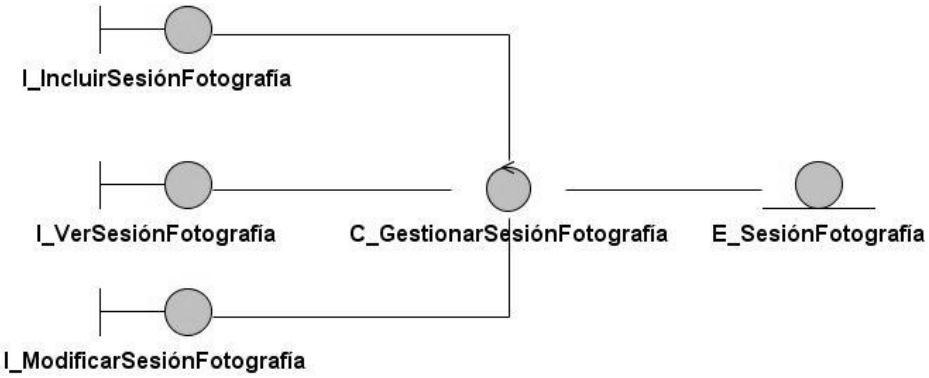

Figura 30. Diagrama de Clases de Análisis. CU Gestionar Sesión de Fotografía.

Figura 31. Diagrama de Clases de Análisis. CU Consultar Sesión de Fotografía.

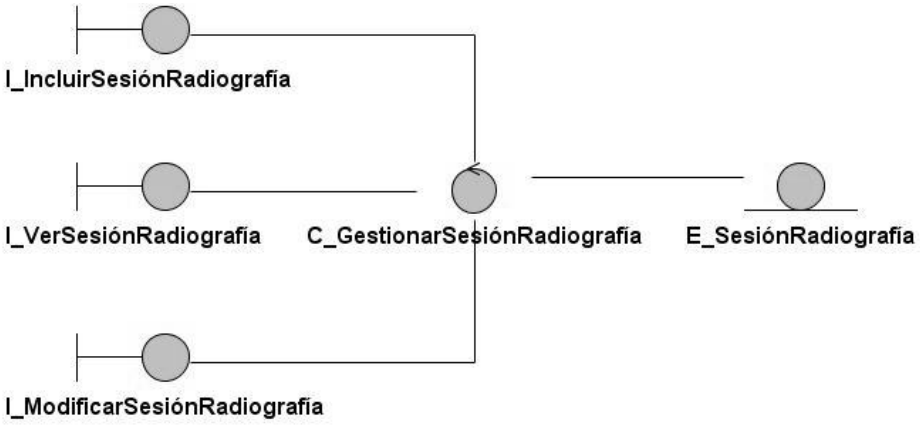

Figura 32. Diagrama de Clases de Análisis. CU Gestionar Sesión de Radiografía.

Figura 33. Diagrama de Clases de Análisis. CU Consultar Sesión de Radiografía.

Figura 34. Diagrama de Clases de Análisis. CU Consultar Citas.

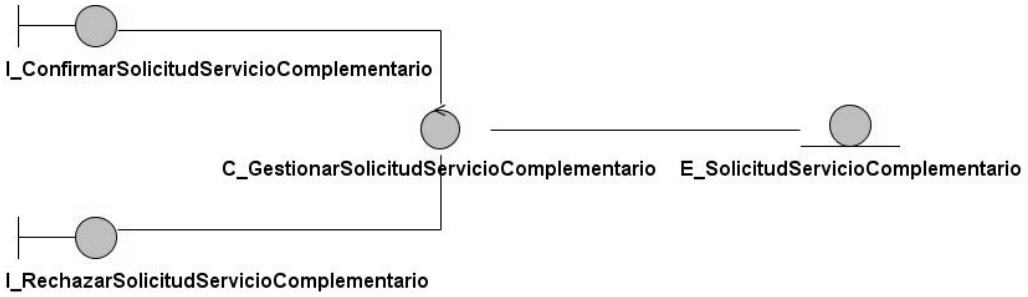

Figura 35. Diagrama de Clases de Análisis. CU Confirmar Solicitud de Servicio.

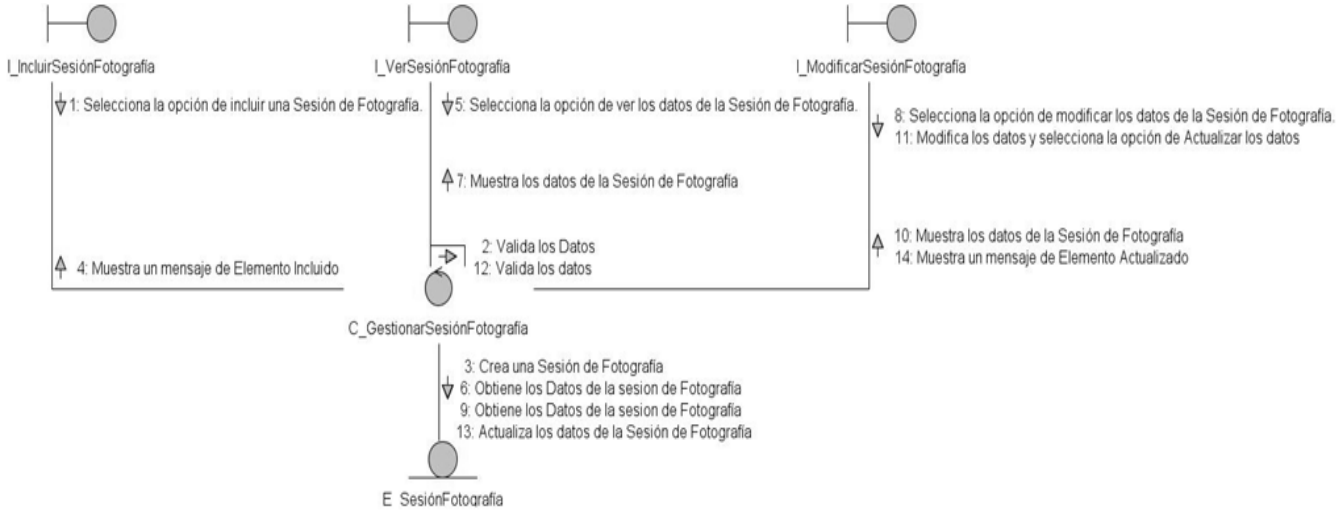

Figura 36. Diagrama de Colaboración. CU Gestionar Sesión de Fotografía.

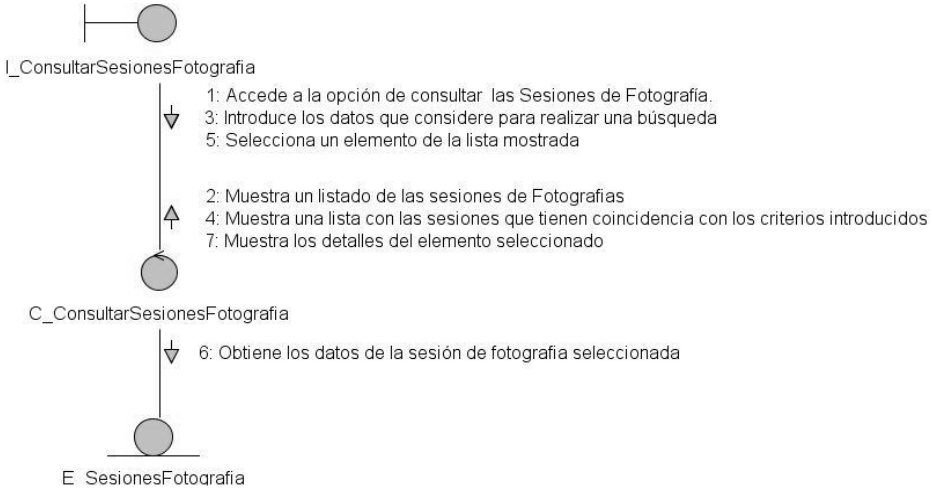

Figura 37. Diagrama de Colaboración. CU Gestionar Sesiones de Fotografías.

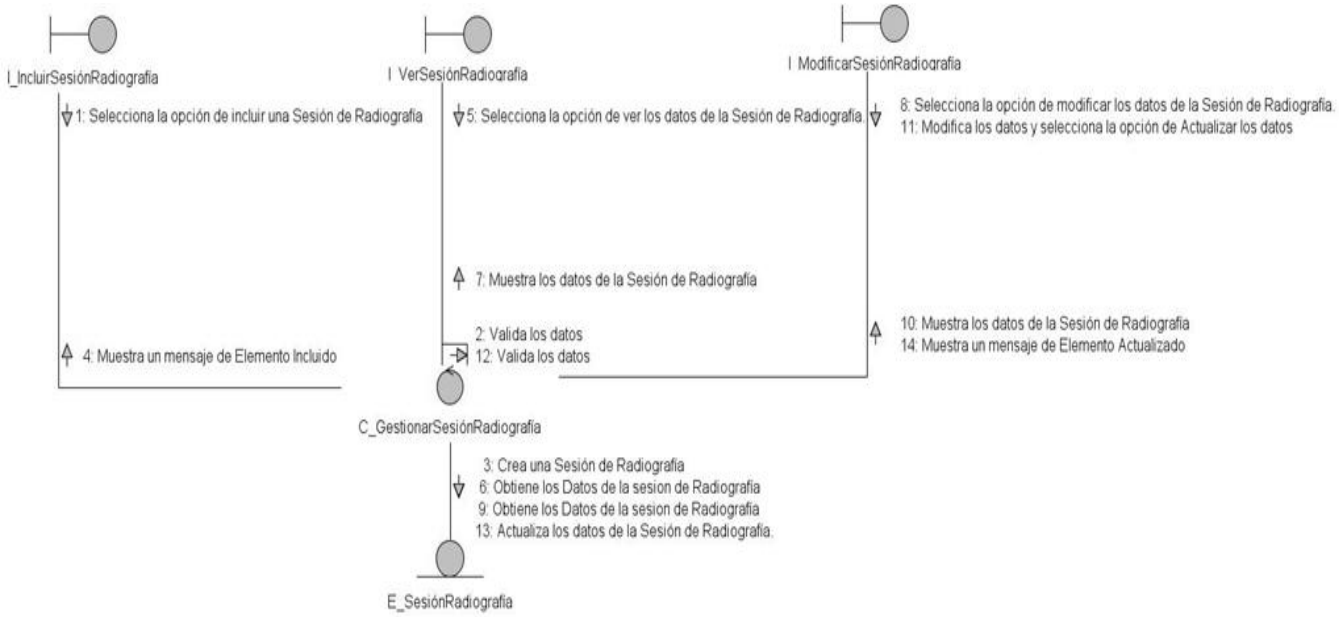

Figura 38. Diagrama de Colaboración. CU Gestionar Sesión de Radiografía.

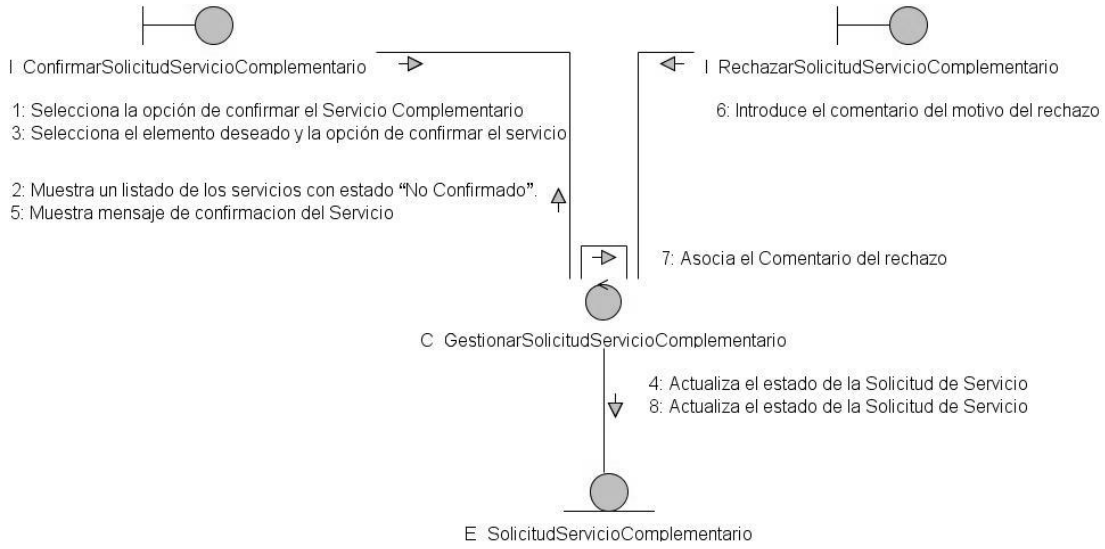

Figura 39. Diagrama de Colaboración. CU Confirmar Solicitud de Servicio.

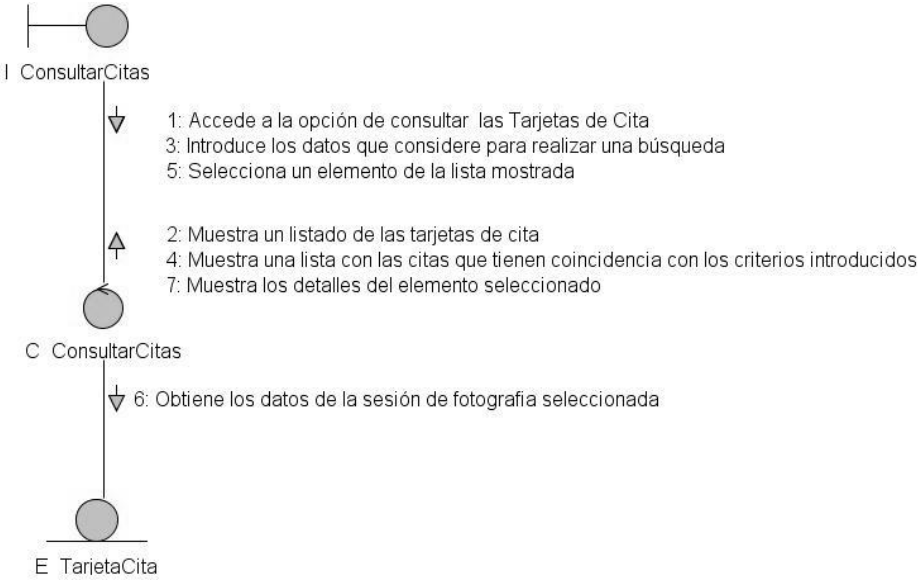

Figura 40. Diagrama de Colaboración. CU Consultar Citas.

Anexo II Modelo de diseño

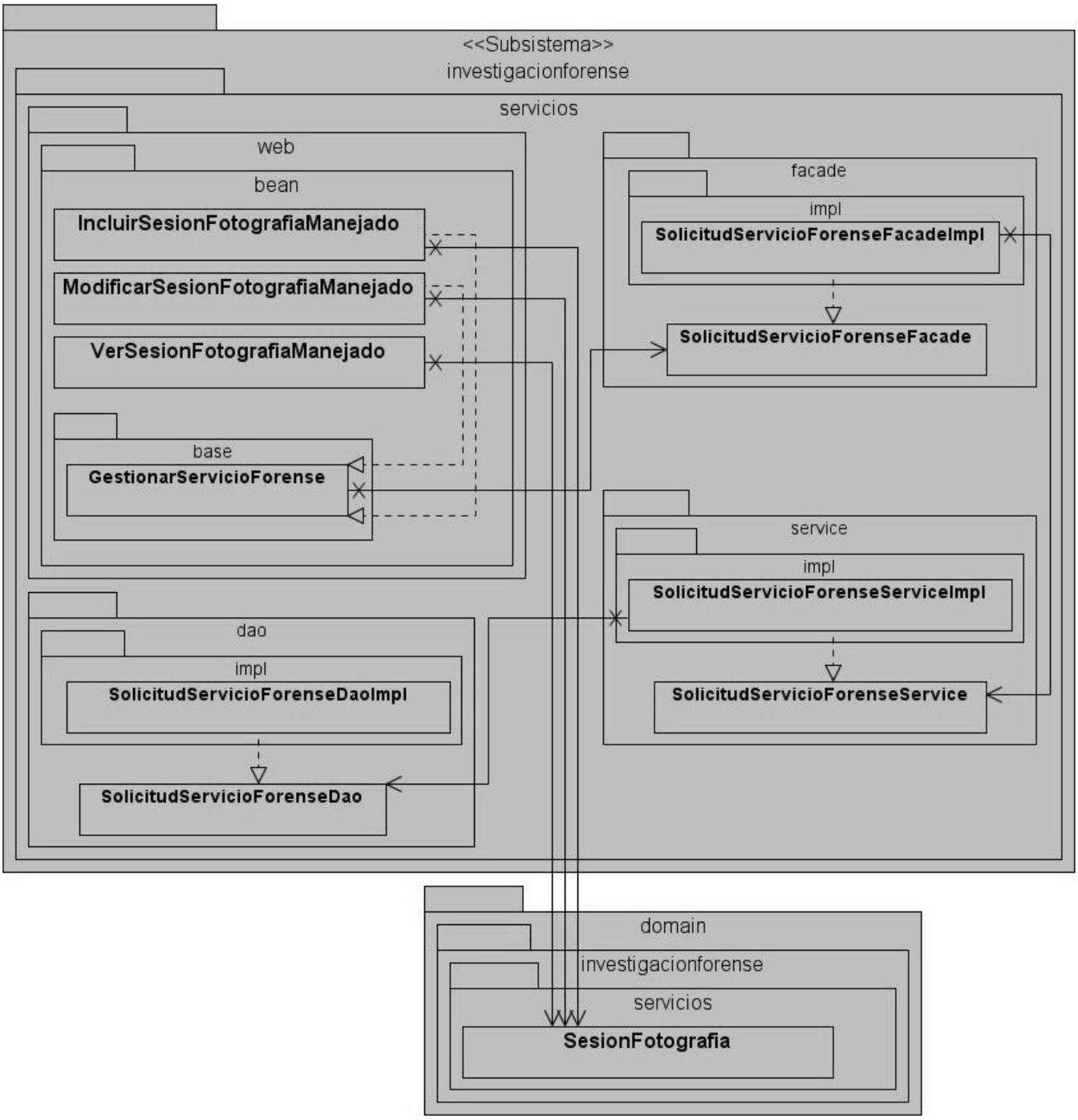

Figura 41. Diagrama de Clases de Diseño. CU Gestionar Sesión de Fotografía.

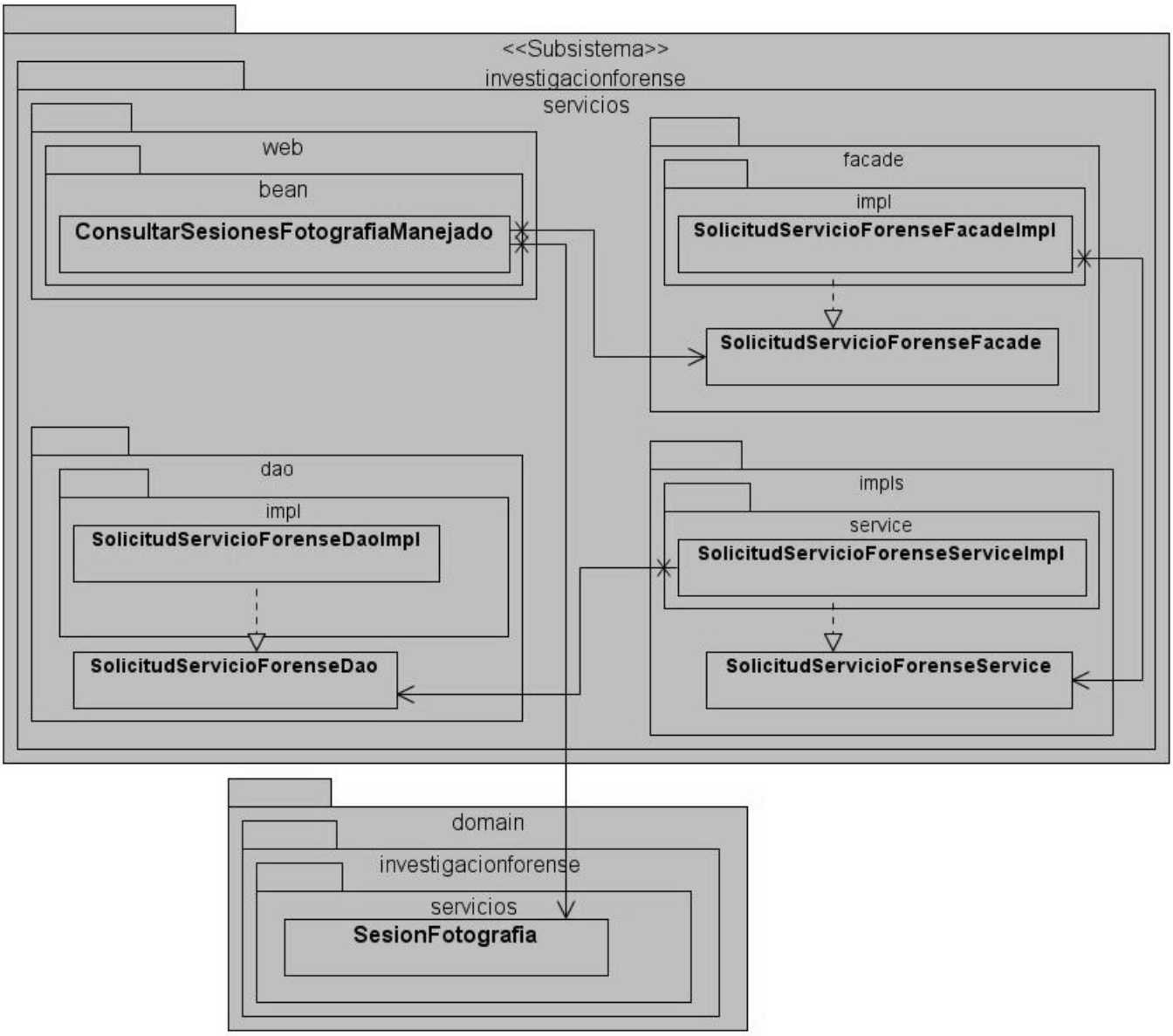

Figura 42. Diagrama de Clases de Diseño. CU Consultar Sesión de Fotografía.

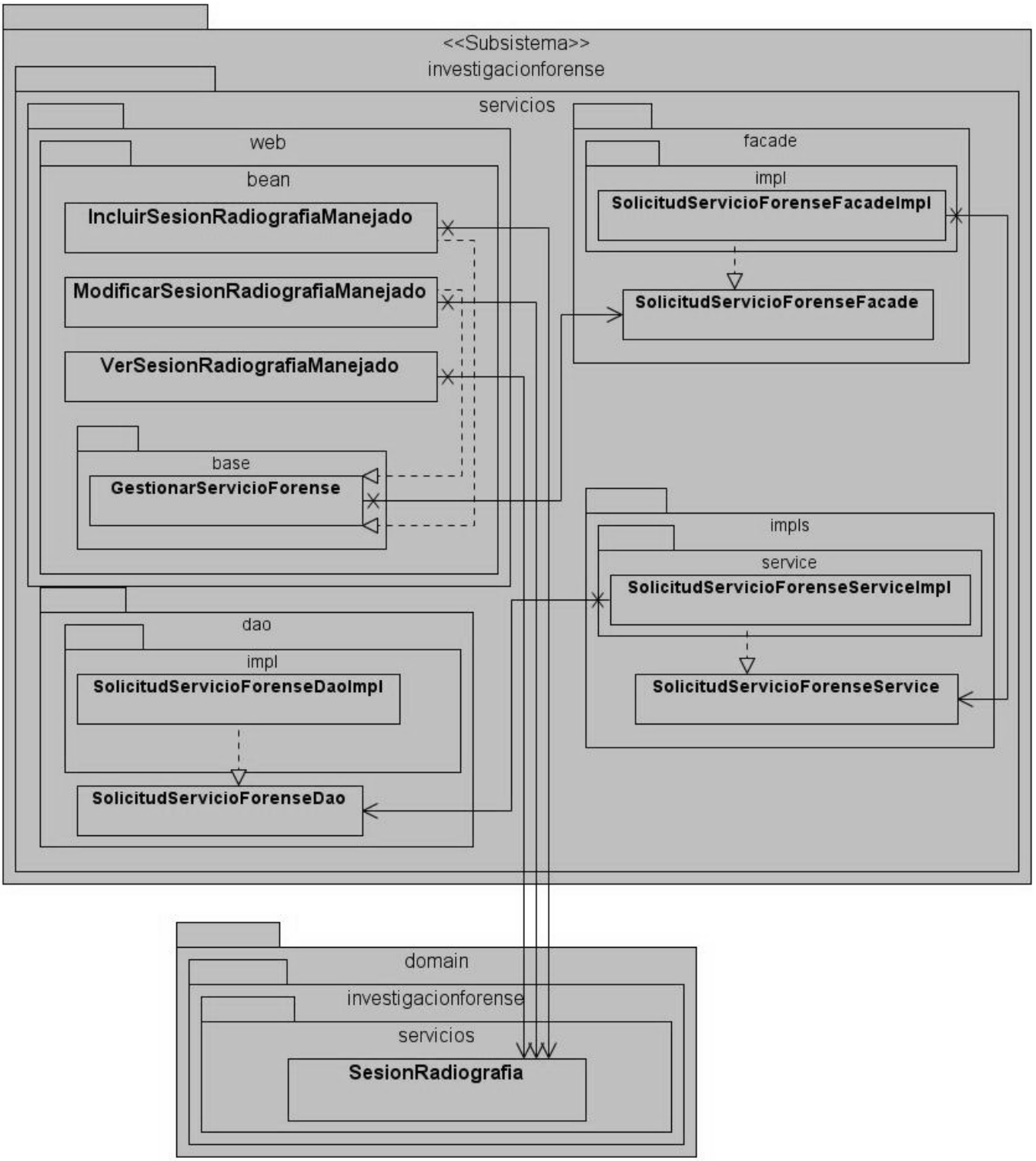

Figura 43. Diagrama de Clases de Diseño. CU Gestionar Sesión de Radiografía.

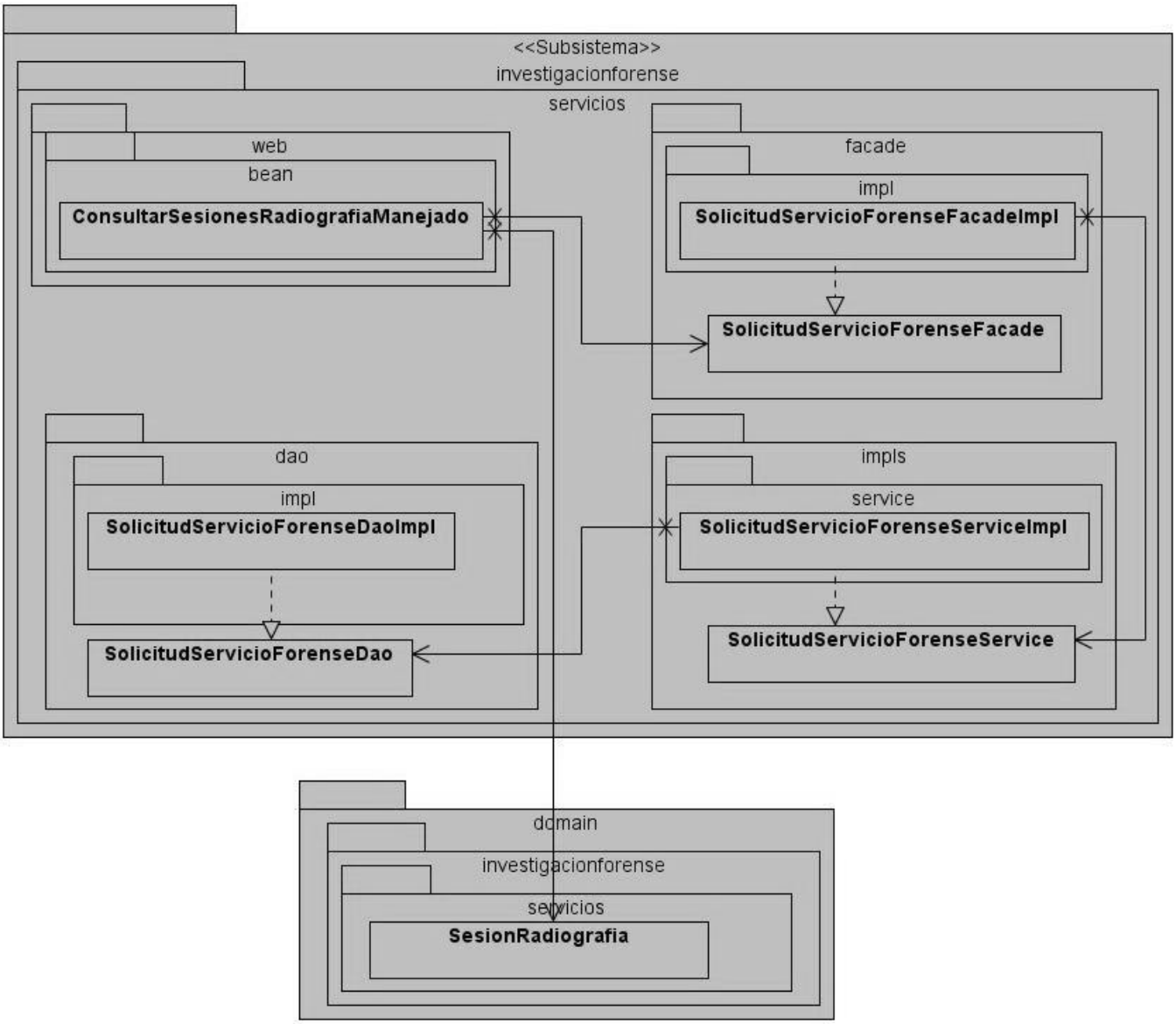

Figura 44. Diagrama de Clases de Diseño. CU Consultar Sesión de Radiografía.

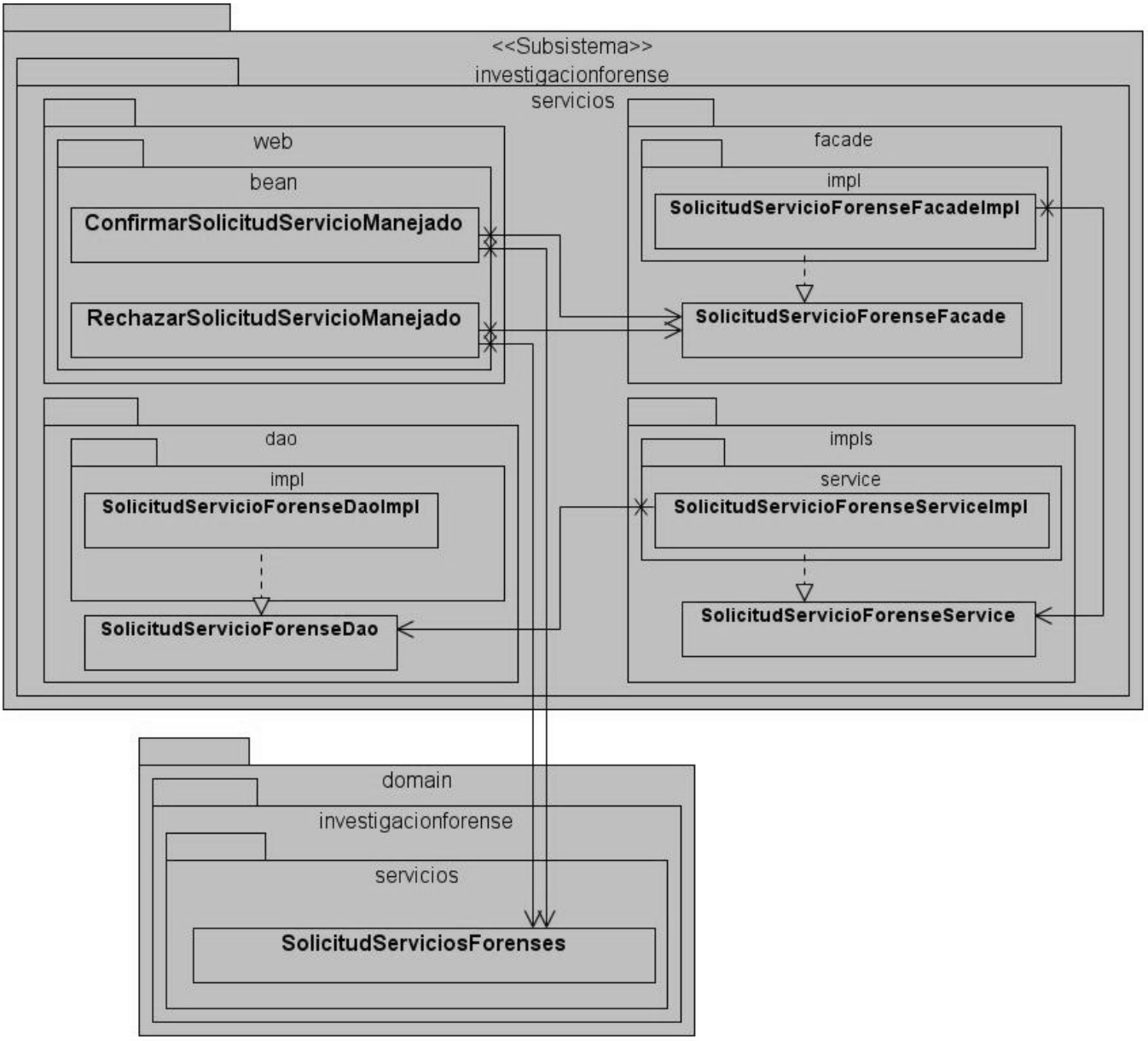

Figura 45. Diagrama de Clases de Diseño. CU Confirmar Solicitud de Servicio.

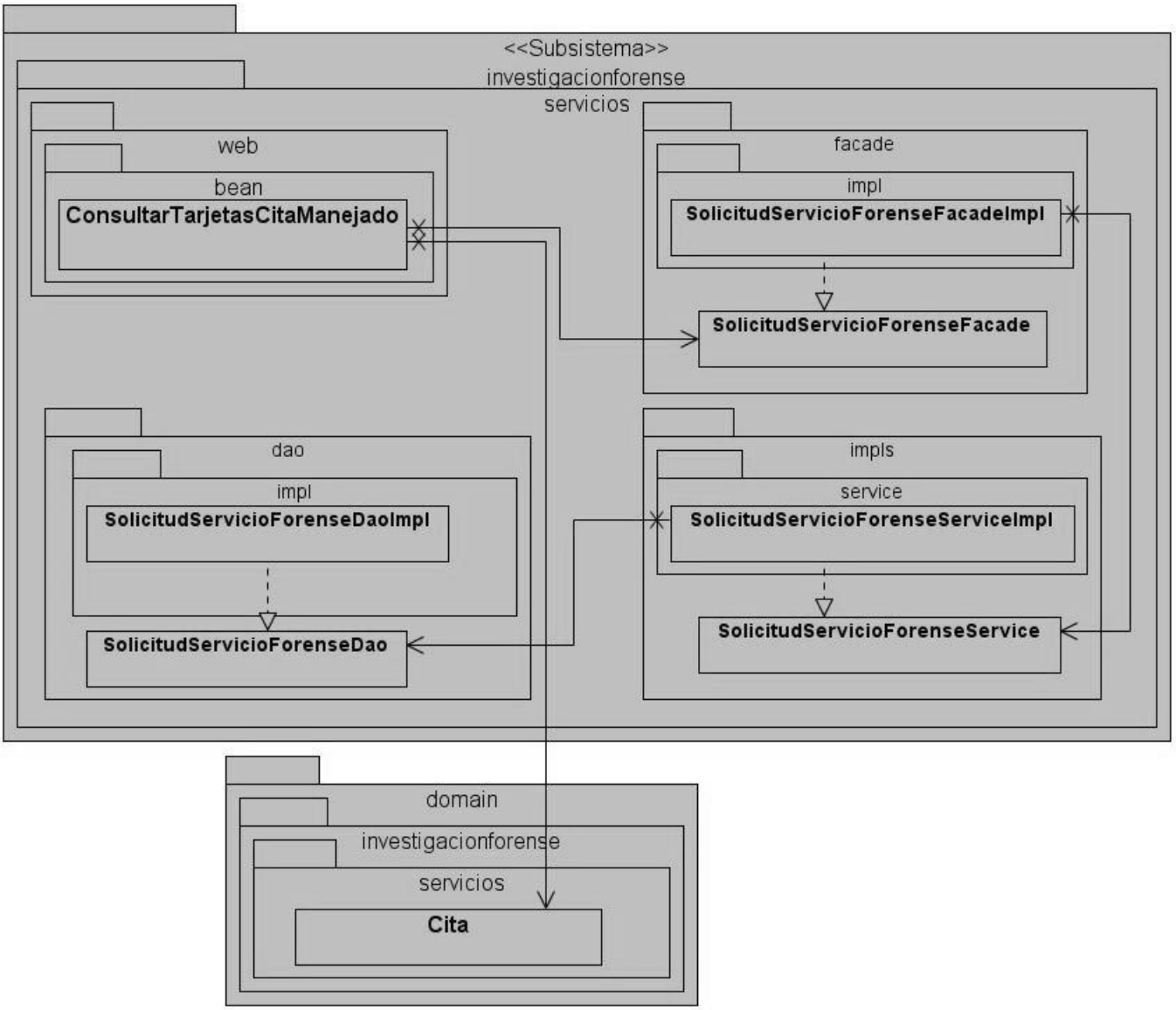

Figura 46. Diagrama de Clases de Diseño. CU Consultar Citas.

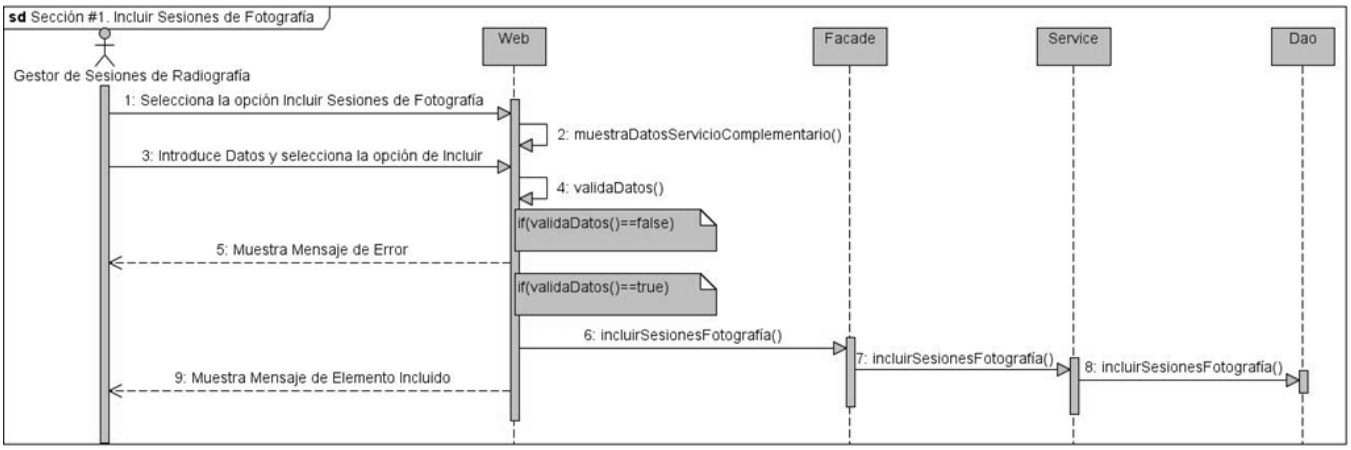

Figura 47. Diagrama de Contrato entre Paquetes. CU Gestionar Sesión de Fotografía. Sesión Incluir.

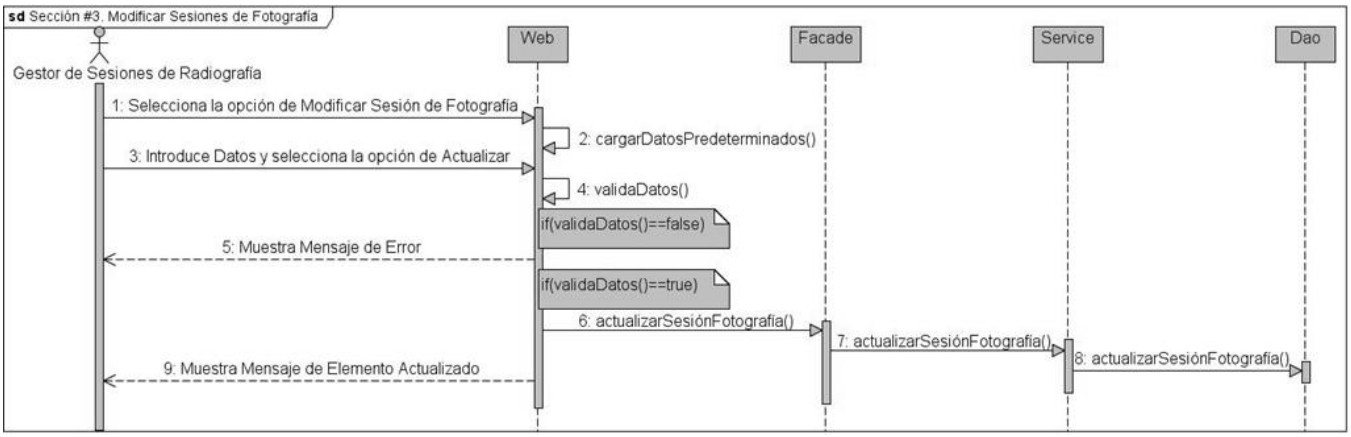

Figura 48. Diagrama de Contrato entre Paquetes. CU Gestionar Sesión de Fotografía. Sesión Modificar.

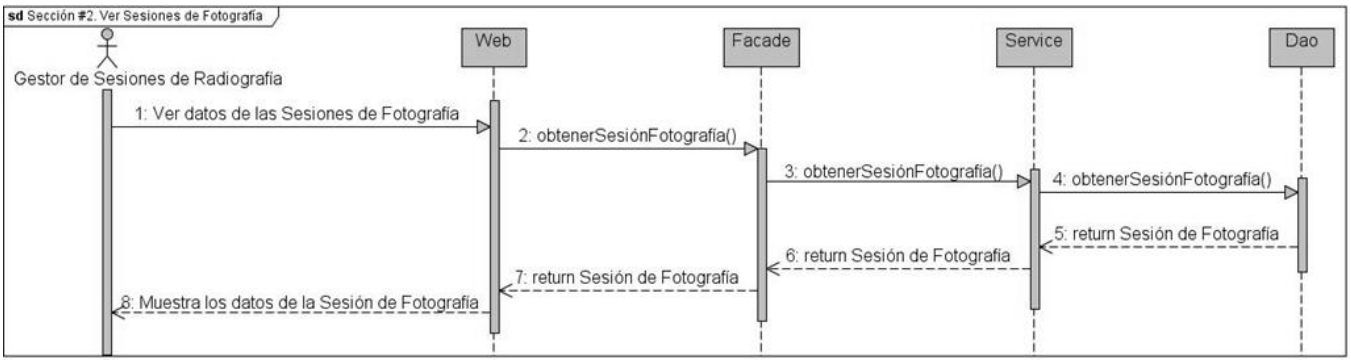

Figura 49. Diagrama de Contrato entre Paquetes. CU Gestionar Sesión de Fotografía. Sesión Ver.

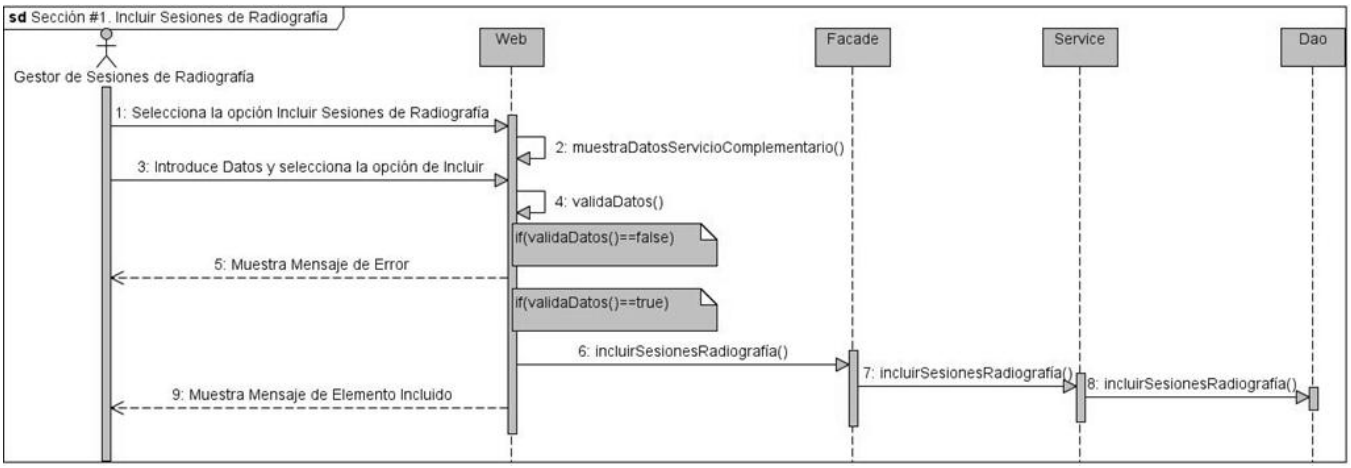

Figura 50. Diagrama de Contrato entre Paquetes. CU Gestionar Sesión de Radiografía. Sesión Incluir.

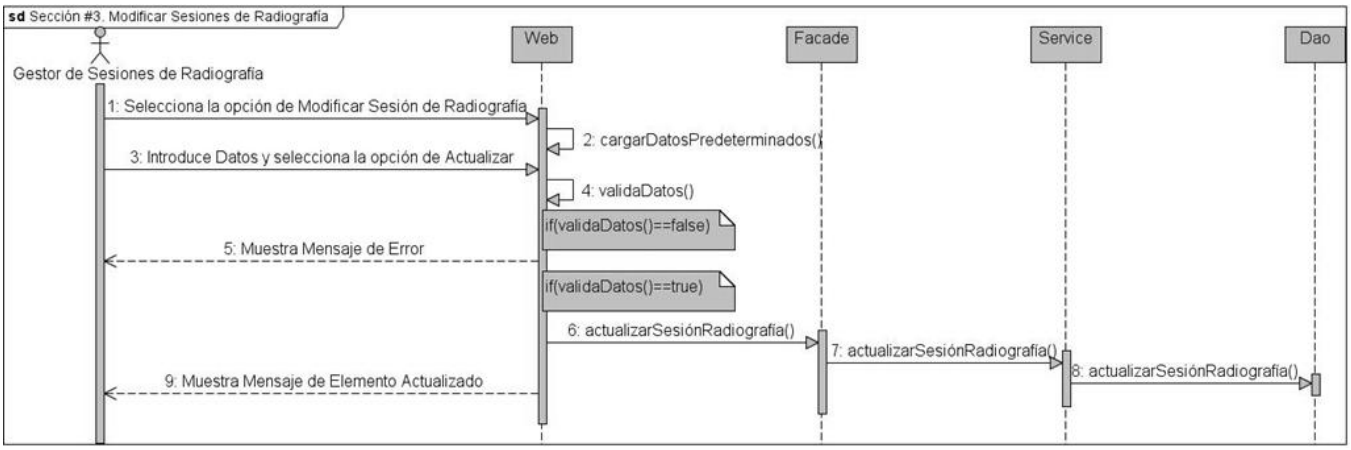

Figura 51. Diagrama de Contrato entre Paquetes. CU Gestionar Sesión de Radiografía. Sesión Modificar.

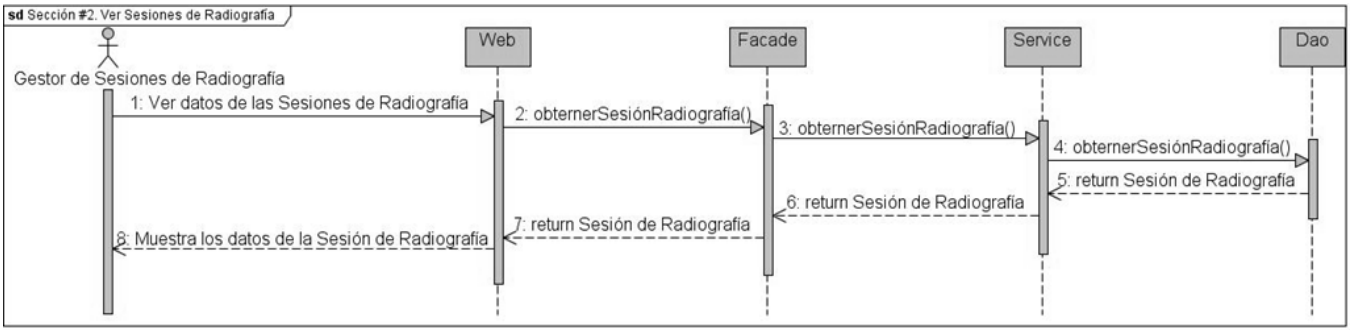

Figura 52. Diagrama de Contrato entre Paquetes. CU Gestionar Sesión de Radiografía. Sesión Ver.

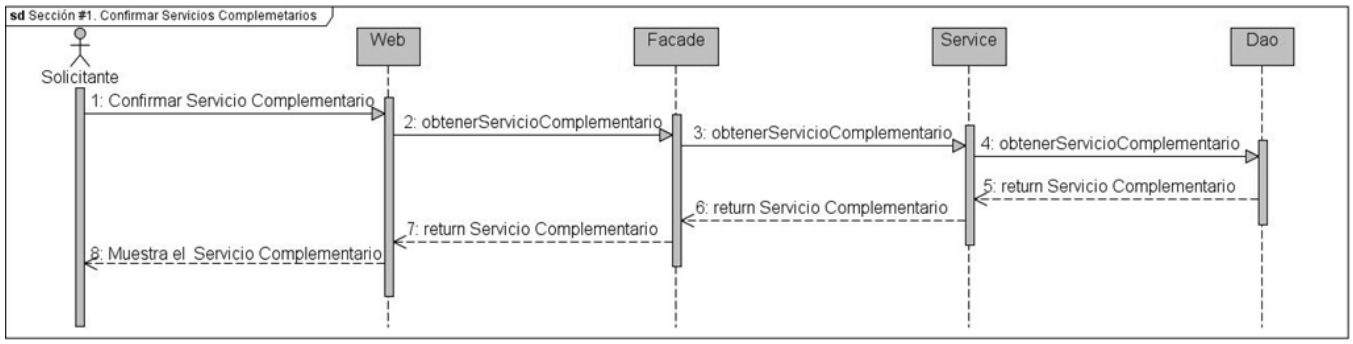

Figura 53. Diagrama de Contrato entre Paquetes. CU Confirmar Solicitud de Servicio. Sesión Confirmar.

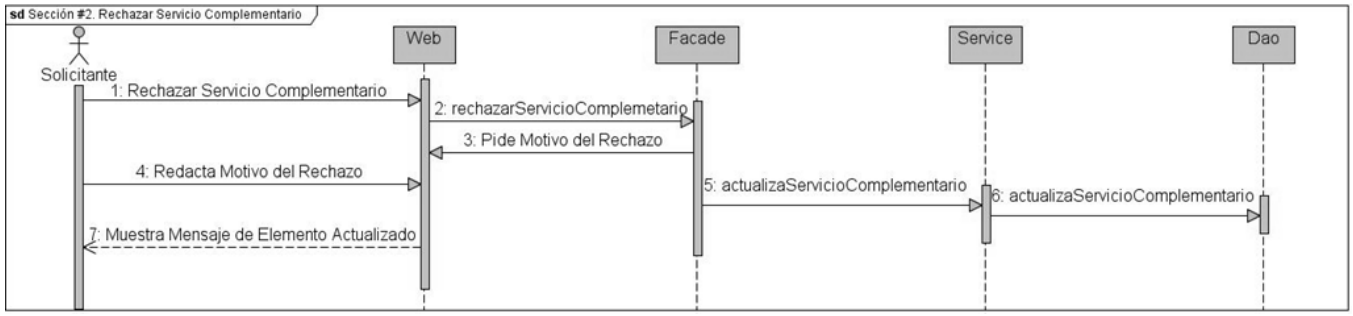

Figura 54. Diagrama de Contrato entre Paquetes. CU Confirmar Solicitud de Servicio. Sesión Rechazar.