Universidad de las Ciencias Informáticas

Facultad 8

ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE LA CAPA DE INTERFAZ DE USUARIO DEL MÓDULO INVESTIGACIÓN EN CIENCIAS FORENSES DEL SIIPOL

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

Autores: Yendy Ponce Castañón
Yasser Chacón Cabrera

Tutores: Ing. Susel Ruiz Durán
Ing. Lesky Alfonso De León

Ciudad de La Habana, Junio 2008
"Año del 50 de la Revolución"

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 firmo la presente a los ____ días del mes de _____ del año _____.

_____ _____

Firma de Yendy Ponce Castañón Firma del Tutor

_____ _____

Firma de Yasser Chacón Cabrera Firma del Tutor

"En los momentos de crisis sólo la creatividad es más importante que el conocimiento."

Albert Einstein

AGRADECIMIENTOS

A nuestros padres, nadie más que ellos hacen realidad el hecho de vernos hechos ingenieros.

A nuestras familias por sabernos guiar en la vida por el camino correcto.

A nuestro Comandante en Jefe Fidel Castro Ruz,

por haber hecho realidad el sueño de ser un profesional.

A nuestros amigos.

A nuestros profesores, indudables creadores de lo que hemos podido llegar a ser.

A nuestros compañeros de grupo y de proyecto.

A tutores, en especial a Lesky Alfonso de León, por su ayuda.

A la UCI por ser nuestro hogar en los últimos cinco años de nuestras vidas.

A todos aquellos que no hemos mencionado, pero que de sobra se deben dar por aludidos por haber contribuido de un modo u otro en nuestra formación y a la consumación de este sueño.

A todos, Gracias.

Le agradezco:

A mi mamá por brindarme todo su amor, su confianza y su apoyo incondicional en los momentos difíciles, por estar siempre cuando la necesito y creer en mí todo el tiempo. Por ser mi todo.

A mami por su inmenso amor y dedicación a lo largo de todos los años de mi vida, por ayudarme y apoyarme en todo y malcriarme todo el tiempo. Por ser muy importante para mí.

A mi papá que me dio su cariño y amor y que siempre quiso lo mejor para mi.

A mi abuela Gladis, que aunque ya no está entre nosotros, sé que estaría muy orgullosa de verme convertida en ingeniera.

A mi tías, Berta por ser mi desahogo aquí en la escuela, aunque solo haya sido por correo, y Mirna por sacarme de tantos apuros.

A mis hermanos, y a mis lindas sobrinas, que quisiera tenerlas más cerca.

Al resto de mi familia por su confianza, dedicación y comprensión.

A mi amigo Maikel que siempre me ha dado su comprensión, su confianza, su cariño y su apoyo incondicional, por ayudarme en todo y ser mi paño de lágrimas.

A Alicia por darme la primera opinión de la tesis.

A mis amigas, Greilan, Ilidian y Diana por aguantarme en mis peores momentos, y por saber que tengo a alguien con quien contar.

A Leyvis Luis, por ayudarme todo el tiempo y complacerme mis antojos sin pensarlo dos veces.

A Ale, por enseñarme que de los golpes se aprende y por ser mi impulso tanto tiempo, a Maritza por ser tan cariñosa.

A mis amigas y amigos dentro y fuera de esta universidad, a Yamila y Boris, aunque ya no hayan terminado con nosotros, a todos mis amigos de la Cujae, al Yose.

A mis compañeros de aula, de los que he aprendido tanto.

A todos los que aportaron algo para contribuir a mi tesis, a Luis Emilio.

A todos los que siempre confiaron en mí y me dieron su apoyo incondicional, gracias.

Yendy

Le agradezco:

A mi mamá, por ayudarme y apoyarme siempre en mis estudios y en todas las decisiones importantes que he tomado en la vida y sobre todo por quererme más que a sí misma.

A mi papá, por darme todo lo necesario para llegar al final, así que ya te puedes retirar y dejar al marabú crecer.

A mi hermanita, porque aunque mami siempre me ha prestado más atención nunca te has puesto celosa y si no fuese por ti no se habría realizado el sueño de graduarme hoy.

A mi tía Lazarita, por ser mi segunda madre y acogerme como a un hijo, por criarme, educarme y estar siempre a mi lado.

A mi novia Daismary, por estar junto a mí en los buenos y malos momentos, por ayudarme a encontrar soluciones cuando más desesperado me encuentro, por ser mi amor lindo.

A mi tía Cira y a su esposo Jorge, por acogerme en su casa con mucha atención y cariño desde el primer momento que llegue a la Habana.

A la familia de mi novia por hacerme sentir como un hijo, por el cariño, el amor y la preocupación que me han mostrado en todos estos años.

Yasser

DEDICATORIA

A mi mamá, a mami y a mi papá.

Yendy

A toda mi familia, en especial a mi mamá Librada y a mi hermanita Yadira.

Yasser

RESUMEN

En la República Bolivariana de Venezuela, El Cuerpo de Investigaciones Científicas, Penales y Criminalísticas (CICPC) es una institución que garantiza la eficiencia en la investigación del delito, mediante su determinación científica. La Coordinación Nacional de Ciencias Forenses (CNCF) es una institución auxiliar de justicia adscripta al CICPC que provee servicios forenses a nivel nacional. Actualmente CICPC cuenta con el Sistema Integrado de Información Policial (SIIPOL) que permite tener centralizada la información de interés para los cuerpos policiales. El sistema no brinda funcionalidades para el manejo o consulta de la información que se genera en las áreas de la CNCF. Dentro del procesos de modernización del SIIPOL se enmarca el presente trabajo como una propuesta de análisis, diseño e implementación del Módulo Investigación en Ciencias Forenses, perteneciente al nuevo sistema, desde el punto de vista de la capa Interfaz de Usuario (UI), que permitirá dar soporte a los principales servicios que brinda la CNCF. Se incluye además dentro del proceso de desarrollo el uso de una metodología de diseño y desarrollo para las Interfaces de Usuario, Metodología para el Diseño y Desarrollo de las Interfaces Web de Usuario (DDWUI). En el presente documento se exponen los resultados de la investigación realizada, se analizan otros sistemas de gestión de información policial, se fundamenta la selección de las metodologías, herramientas y tecnologías utilizadas y se presenta la propuesta de sistema. Por último, se exponen los artefactos generados en el análisis, diseño e implementación.

ÍNDICE

Introducción *1*

Fundamentación Teórica *5*

Introducción **5**

1.1 Sistemas de Gestión de Información Policial **5**

1.2 Análisis de las principales metodologías para el desarrollo de software. Selección de la metodología que apoya la solución del problema **6**

1.2.1 Metodologías Tradicionales 7

1.2.1.1 RUP 7

1.2.1.2 MSF 10

1.2.2 Metodologías Ágiles 12

1.2.2.1 XP 13

1.2.3 Selección de la metodología a utilizar 15

1.3 UML **15**

1.4 Propuesta de sistema **16**

1.5 Análisis de las principales plataformas de desarrollo. Selección de la plataforma de desarrollo que apoya la solución del problema **18**

1.5.1 La plataforma J2EE 19

1.5.2 La plataforma .NET 20

1.5.3 Selección de la plataforma a utilizar 20

1.6 Herramientas a utilizar **21**

1.6.1 IDE de desarrollo 21

1.6.1.1 Eclipse 21

1.6.2 Herramientas Case 22

1.6.2.1 Visual Paradigm 22

1.7 Análisis de los principales Frameworks. Selección del Frameworks a utilizar **23**

1.7.1 Tecnología JSP-Servlet 23

1.7.2 Modelo Vista-Controlador 24

1.7.2.1 Modelo 2 24

1.7.3 Struts 25

1.7.4 JSF 27

1.7.5 Framework a utilizar 28

1.8 Tecnologías del lado del cliente **29**

1.8.1 HTML 29

1.8.2 CSS 29

1.8.3 JavaScript 30
1.8.4 AJAX 30

1.9 Análisis de las metodologías para el diseño y desarrollo de las Interfaces de Usuario. Selección de la metodología a utilizar 31

Conclusiones 32

Desarrollo de la Solución Propuesta 33

Introducción 33

2.1 Modelado del Sistema 33
2.1.1 Modelo de CU del sistema 33
2.1.2 Descripción de los CU del sistema 36

2.2 Diseño de la UI 39
2.2.1 Fases de la metodología a utilizar 39
2.2.2 Resultados Obtenidos 40

2.3 Análisis 43
2.3.1 Modelo de Análisis 43
2.3.1.1 Clases del análisis 44
2.3.2 Diagramas de clases del análisis 44

2.4 Diseño 46
2.4.1 Modelo de Diseño 46
2.4.2 Estructura lógica de paquetes 46
2.4.3 Diagrama de Contrato entre Paquetes 47
2.4.4 Diagrama de clases de diseño 51
2.4.5 Descripción de las Clases 56

2.5 Implementación 72
2.5.1 Modelo de Implementación 72
2.5.2 Subsistema de Implementación 72
2.5.3 Modelo de Despliegue 76

Conclusiones 77

Conclusiones Generales 78

Recomendaciones 79

Referencias Bibliográficas 80

Bibliografía 82

Glosario de Términos 83

ÍNDICE DE FIGURAS

Fig. 1 RUP en Dos Dimensiones.....8
Fig. 2 Fases de la metodología MSF.....11
Fig. 3 Esquema de Trabajo XP. 14
Fig. 4 Cliente-Servidor. 17
Fig. 5 Patrón 3 capas.....18
Fig. 6 Esquema del patrón MVC. 24
Fig. 7 Esquema del patrón Front Controller.....25
Fig. 8 Tecnologías agrupadas bajo el concepto de AJAX. 31
Fig. 9 Diagrama de CU subsistema solicitudes. 34
Fig. 10 Diagrama de CU subsistema control de investigación.....34
Fig. 11 Diagrama de CU subsistema experticias.....35
Fig. 12 Ejemplo del primer prototipo de UI. CU Gestionar Protocolo de Exhumación.....42
Fig. 13 Muestra de la pantalla del CU Protocolo de Exhumación en formato .jpg. 43
Fig. 14 Diagrama de clases del análisis CU Ver Detalles de Experticia. 44
Fig. 15 Diagrama de clases del análisis CU Consultar Detalles de Experticias Asignadas.....45
Fig. 16 Diagrama de clases del análisis CU Consultar Registro Histórico de Experticias/Casos Forenses.....45
Fig. 17 Diagrama de clases del análisis CU Gestionar Protocolo de Exhumación.....45
Fig. 18 Estructura de Carpetas. 46
Fig. 19 Diagrama de contrato entre paquetes CU Ver Detalles de Experticia.....48
Fig. 20 Diagrama de contrato entre paquetes CU Consultar Experticias Asignadas. 48
Fig. 21 Diagrama de contrato entre paquetes CU Consultar Registro Histórico de Experticias/Casos Forenses.....49
Fig. 22 Diagrama de contrato entre paquetes CU Gestionar Protocolo de Exhumación.....50
Fig. 23 Diagrama de clases de diseño CU Ver Detalles de Experticia.....52
Fig. 24 Diagrama de clases de diseño CU Consultar Experticias Asignadas. 53
Fig. 25 Diagrama de clases de diseño CU Consultar Registro Histórico de Experticias/Casos Forenses.....54
Fig. 26 Diagrama de clases de diseño CU Gestionar Protocolo de Exhumación.....55
Fig. 27 Diagrama del Subsistema de Implementación de la Capa UI.....74
Fig. 28 Diagrama de Componentes del Subsistema de Implementación solicitudes.bean. 75
Fig. 29 Diagrama de Componentes del Subsistema de Implementación WebContent.forense.solicitudes.....76
Fig. 30 Diagrama de Despliegue.....77

ÍNDICE DE TABLAS

Tabla 1 Descripción del CU Ver Detalles de Experticia.....36
Tabla 2 Descripción del CU Consultar Experticias Forenses Asignadas.....36
Tabla 3 Descripción del CU Consultar Registro Histórico de Experticias/Casos Forenses.....37
Tabla 4 Descripción del CU Gestionar Protocolo de Exhumación.....38
Tabla 5 Descripción de clases CU Ver Detalles Experticia.....56
Tabla 6 Descripción de clases CU Consultar Experticias.....59
Tabla 7 Descripción de clases CU Consultar Registro Histórico de Experticias/Casos Forenses.....62
Tabla 8 Descripción de clases CU Gestionar Protocolo de Exhumación.....64

INTRODUCCIÓN

La inseguridad y principalmente la criminalidad en cada país, representan la mayor angustia de los ciudadanos, de los 193 países que integran el planeta, la República Bolivariana Venezuela ocupa el puesto número 13 en las estadísticas de mayor criminalidad (1).

En este sentido, para que la República Bolivariana de Venezuela o cualquier otro país tenga adecuados niveles de desarrollo humano, es necesario que el Estado garantice la convivencia y la seguridad de los ciudadanos, fortaleciendo los sistemas de justicia y seguridad ciudadana, de tal forma que los ciudadanos puedan gozar de sus derechos y puedan cumplir con sus deberes (2).

Debido al aumento de la delincuencia, los órganos encargados de la administración de justicia requieren, constantemente, del estudio de las muestras vinculadas a hechos delictivos o a escenas del crimen, para el esclarecimiento de las infracciones cometidas. En la República Bolivariana de Venezuela el **Cuerpo de Investigaciones Científicas, Penales y Criminalísticas (CICPC)** es una institución que integra las investigaciones de tipo científicas y criminalísticas y sus resultados son usados por el Ministerio del Poder Popular para las Relaciones Interiores y Justicia, responsables legales de la investigación. CICPC se caracteriza por su amplia capacidad científica, su alta disponibilidad de recursos y la rapidez de respuestas, que le permite convertirse en una institución de obligada consulta para el enfrentamiento al delito. Pero, aunque cuenta con todos los indicadores antes expuestos, no es capaz de brindar información necesaria sobre servicios forenses.

La **Coordinación Nacional de Ciencias Forenses (CNCF)** es una institución auxiliar de justicia adscripta al CICPC que provee servicios forenses a nivel nacional. Sus principales funciones están dirigidas a servir de apoyo al proceso de investigación y a la organización, administración y supervisión de todas las actividades forenses en servicio del trabajo policial.

CNCF es la institución nacional rectora, acreditada científica y académicamente en las Ciencias Forenses; que intenta brindar un aporte óptimo de peritaje legal, objetivo y fidedigno como medios de prueba en la administración de justicia. Además pretende garantizar los servicios públicos forense a nivel nacional; operativos y actualizados en dependencia de 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.

En la actualidad, CNCF, atraviesa por una situación que limita sus aportes al desarrollo del proceso de investigación. En ocasiones la atención prestada a las cuestiones forense, por los cuerpos policiales o instituciones judiciales, es poca y no utilizan en su totalidad los servicios que brinda la coordinación, muchas veces por el desconocimiento de los mismos. 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 al ente que lo solicite. Suelen perderse los datos recogidos durante la investigación que puedan aportar información de interés a los expertos en el momento de emitir una valoración.

La no existencia de sistemas de comunicación entre las áreas de una misma unidad imposibilita el acceso, de forma rápida, a la información generada en las mismas. Además las unidades no se comunican con sistemas externos, como por ejemplo la Oficina Nacional de Identificación y Extranjería (ONIDEX).

Muchas veces se toman decisiones o se dan respuestas equivocadas producto a la desactualización de la información. Esto se debe, en gran medida, a que la información generada en las diferentes unidades de la medicatura, distribuidas por todo el país, llega retrasada o no llega a las demás unidades que la necesitan.

Actualmente CICPC cuenta con el **Sistema Integrado de Información Policial (SIIPOL)** que permite tener centralizada la información de interés para los cuerpos policiales. Los funcionarios, cuerpos policiales estadales y municipales que tienen acceso al sistema pueden consultar personas buscadas por la justicia, antecedentes penales, delictivos, así como vehículos robados.

Este sistema está desarrollado sobre una tecnología obsoleta: sobre el lenguaje de programación Adabas-Natural y los servidores de base datos SUN 6500, hoy en día fuera de mercado. La información que brinda SIIPOL no siempre cuenta con la calidad y la inmediatez que se necesita, sumado a la complejidad de su uso, provocan que se demoren los procesos de investigación.

El sistema actual no brinda funcionalidades para el manejo o consulta de la información que se genera en las áreas de la CNCF, que es de suma importancia para el esclarecimiento de los delitos.

En el país se han aprobado contratos de negocio con empresas venezolanas relacionados con las Tecnologías de la Informática y la Comunicaciones (TIC). Parte de estos contratos se llevan a cabo en el marco del ALBA, mediante la empresa comercializadora ALBET de la Universidad de las Ciencias Informáticas (UCI), que en estos momentos se encuentra desarrollando el proyecto productivo CICPC encargado de la automatización y modernización del SIIPOL.

Para dar solución a esta problemática, la modelación del sistema del nuevo SIIPOL incluirá el Módulo de Investigación en Ciencias Forenses, donde se agrupan las funcionalidades referidas al trabajo en las medicaturas forenses.

Terminada la fase de modelado del sistema quedan creadas las condiciones para que dé comienzo la etapa de análisis, diseño y la implementación, por lo que el **problema** a resolver es: ¿Cómo garantizar el cumplimiento de los requisitos funcionales y no funcionales; y las pautas de diseño gráfico, asociados al Módulo de Investigación en Ciencias Forenses del sistema SIIPOL desde el punto de vista de la capa UI?

Por tanto, el **objeto de estudio** queda conformado por la modelación del sistema del Módulo de Investigación en Ciencias Forenses del Sistema SIIPOL y el **campo de acción** sea la capa UI de este subsistema.

Lo anteriormente expuesto se orienta al cumplimiento del **objetivo general** de analizar, diseñar e implementar la capa UI para el Módulo de Investigación en Ciencias Forenses; que se adapte a los requisitos funcionales, no funcionales y a las pautas de diseño gráfico especificados para el subsistema.

Los **objetivos específicos** descritos a continuación tributarán al cumplimiento del objetivo general.

- ✓ Analizar y aplicar la metodología para el Diseño y Desarrollo de las Interfaces Web de Usuario (DDWUI) en el Módulo de Investigación en Ciencias Forenses del Sistema SIIPOL.
- ✓ Realizar el diseño de las clases necesarias para el desarrollo de todos los casos de uso (CU) del Módulo de Investigación en Ciencias Forenses del Sistema SIIPOL.
- ✓ Realizar la implementación de todas las clases diseñadas, así como de otras clases que son necesarias para el trabajo con el framework.

Se presenta como **idea a defender** que, si se realiza el análisis, diseño e implementación de la capa UI, del Módulo Investigación en Ciencias Forenses del SIIPOL, aplicando la metodología DDWUI, haciendo uso de la tecnología Java Server Faces (JSF) y cumpliendo con los requisitos funcionales, no funcionales y con las pautas de diseño gráfico, especificados para la aplicación, se podrán garantizar funcionalidades dentro de SIIPOL para el procesado de la información en la CNCF.

A continuación se proponen un conjunto de tareas que ayudarán a la realización de la investigación con mayor calidad.

- ✓ Análisis del Módulo Investigación en Ciencias Forenses del SIIPOL.
- ✓ Descripción y selección de las herramientas, metodologías y tecnologías a utilizar.
- ✓ Aplicación de las pautas de diseño gráfico y la arquitectura propuesta para el trabajo en la capa UI.

- ✓ Definición y realización de los diferentes artefactos para el análisis, el diseño y la implementación.

Como resultado del presente trabajo, el renovado SIIPOL, brindará las funcionalidades para el manejo o consulta de la información que se genera en las áreas de la CNCF. Además, el módulo de Investigación en Ciencias Forenses del SIIPOL, representa una porción importante de todo el sistema, por lo que la confección del mismo significa una fracción importante del pago final, lo que traería ingresos importantes a la economía del país. Por último, el trabajo, servirá de guía para el uso de la tecnología JSF en la universidad y para validar la eficiencia de la metodología DDWUI.

Estructuración del contenido.

El documento está compuesto por 2 capítulos.

El **capítulo 1** "Fundamentación Teórica" está dedicado al estado del arte del tema relacionado con la investigación. Se abordan las tendencias y tecnologías actuales sobre las cuales se apoya la propuesta, así como las distintas metodologías de desarrollo de software, herramientas que serán usadas, lenguajes de programación, etc. Se describen las características de la propuesta de sistema.

El **capítulo 2** "Desarrollo de la solución Propuesta" en él se muestran los artefactos y los documentos generados en los flujos de trabajo análisis, diseño e implementación, además de seguir paso a paso las fases de la metodología escogida para el diseño y montaje de las interfaces del módulo.

CAPÍTULO 1

FUNDAMENTACIÓN TEÓRICA

Introducción

En el capítulo se realiza una investigación sobre los sistemas de gestión de información policial. Se hace un estudio de las principales metodologías de desarrollo de software y se realizan comparaciones entre las mismas, con el fin de justificar la seleccionada para el desarrollo del sistema. Se realiza un análisis, descripción y comparación de las tendencias, tecnologías y herramientas en las que se apoya la propuesta de solución. Por último, se describe brevemente la propuesta de sistema.

1.1 Sistemas de Gestión de Información Policial

A nivel internacional existe una tendencia creciente a la integración de los sistemas de gestión de información policial, esto permite centralizar la información generada por los órganos encargados del enfrentamiento al delito. La centralización de la información y la integración de los diferentes frentes, eliminan los molestos y tardíos trámites de solicitud de información necesaria para el esclarecimiento de los hechos delictivos, que constantemente son necesarios entre los órganos o instituciones que participan en la investigación (3).

Como ejemplos de sistemas de información policial se pueden mencionar:

Helios: Es una aplicación informática integral para la gestión policial que brinda instalaciones personalizadas para cada cliente, manteniendo un alto nivel de servicio. Presenta una máxima seguridad en la protección de los datos y control de acceso. Además cuenta con una plataforma única de gestión del programa (realizar tareas policiales desde una única aplicación informática), así como un diseño de pantallas visual e intuitivo (4).

Seguridad Ciudadana y Gestión Policial STEGPOL: La idea principal del Proyecto STEGPOL, que actualmente opera en Chile, es integrar a las diferentes entidades como Carabineros, Investigaciones, Ministerio del Interior y Municipios, entre otros, en una Plataforma Nacional Común de Información que permita el intercambio de datos y que sirva de apoyo a la gestión operacional

regional o comunal, donde dichas instituciones estén interconectadas entre sí, en especial con diferentes Divisiones o Departamentos de Carabineros (5).

ePatrol: Plataforma definitiva para agilizar, simplificar y automatizar el trabajo diario de la policía local, gestiona los procedimientos administrativos, en toda su duración, para las unidades administrativas implicadas. Se basa en software libre y no requiere del pago de licencias de soporte. Además, permite su uso a través de una plataforma web, por lo que no necesita de ningún tipo de instalación, ni distribución (6).

Como resultado del estudio realizado no se encontró ningún sistema de gestión policial que brindara funcionalidades para consultar información referente a servicios forenses, pero sí se pudo conocer de la profesionalidad y desarrollo que han alcanzado las instituciones que brindan estos servicios. Se puede citar a Cuba, como ejemplo, como afirmó Luís Alberto Kuitko, profesor titular de la especialidad en Argentina, **"***La Medicina Legal cubana está muy bien ubicada y es competente a nivel internacional".* Anualmente en Cuba se investigan unos dos mil cadáveres y quince mil personas vivas (7).

Lo anteriormente expuesto da una idea de la importancia y significación del desarrollo del nuevo SIIPOL, especialmente el módulo Investigación en Ciencias Forenses, que garantizará funcionalidades dentro de SIIPOL para el procesado de la información en la CNCF.

1.2 Análisis de las principales metodologías para el desarrollo de software. Selección de la metodología que apoya la solución del problema

La metodología, será la encargada de elaborar "el plano" sobre el cual se apoyará el equipo de desarrollo. El impacto de elegir la mejor para un determinado proyecto, es trascendental, para el éxito del producto.

Todo desarrollo de software es riesgoso y difícil de controlar, pero si no se lleva una metodología de por medio, se obtienen clientes insatisfechos con el resultado y desarrolladores aún más insatisfechos (8). Es muy importante, cuando los proyectos que se van a desarrollar son de mayor envergadura, seleccionar la correcta de las tantas existentes. En la actualidad, se encuentran congregadas en dos grandes grupos, Metodologías Tradicionales y Metodologías Ágiles.

1.2.1 Metodologías Tradicionales

Al inicio el desarrollo de software era artesanal en su totalidad. La ausencia de procesos formales, lineamientos claros, determinaron que se importara la concepción y fundamentos de metodologías existentes en otras áreas de la producción industrial. Así surgen, a finales de los sesenta, las llamadas metodologías tradicionales para el desarrollo de software. Esta nueva etapa de adaptación contenía el desarrollo dividido en etapas de manera secuencial.

Entre las principales metodologías tradicionales se encuetran Rational Unified Process (RUP) y Microsoft Solution Framework (MSF), entre otras, que centran su atención en guiar el proceso de desarrollo de un software teniendo en cuenta elementos importantes para evitar en lo posible su ambigüedad. Dentro de estos elementos se encuentran la determinación del alcance real de la aplicación, especificación en detalle de las funcionalidades y características del sistema, la planificación y la estimación mediante el cálculo del esfuerzo valorando para ello el tiempo, el costo y los requisitos que especifican el alcance del producto. Otra de las características importantes dentro de este enfoque, son los altos costos al implementar un cambio y la falta de flexibilidad en proyectos donde el entorno es volátil.

Las metodologías tradicionales se focalizan en documentación, planificación y procesos (plantillas, técnicas de administración, revisiones, etc.) (9).

A continuación se analizan RUP y MSF.

1.2.1.1 RUP

RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del Lenguaje Unificado de Modelado (UML), y trabajo de muchas metodologías utilizadas por los clientes. La versión que se ha estandarizado vio la luz en 1998 y se conoció en sus inicios como Proceso Unificado de Rational 5.0; de ahí las siglas con las que se identifica a este proceso de desarrollo.

Como RUP es un proceso, en su modelación define como sus principales elementos:

Trabajadores (“quién”): Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.

Actividades (“cómo”): Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.

Artefactos (”qué”): Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.
Flujo de actividades (“cuándo”): Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

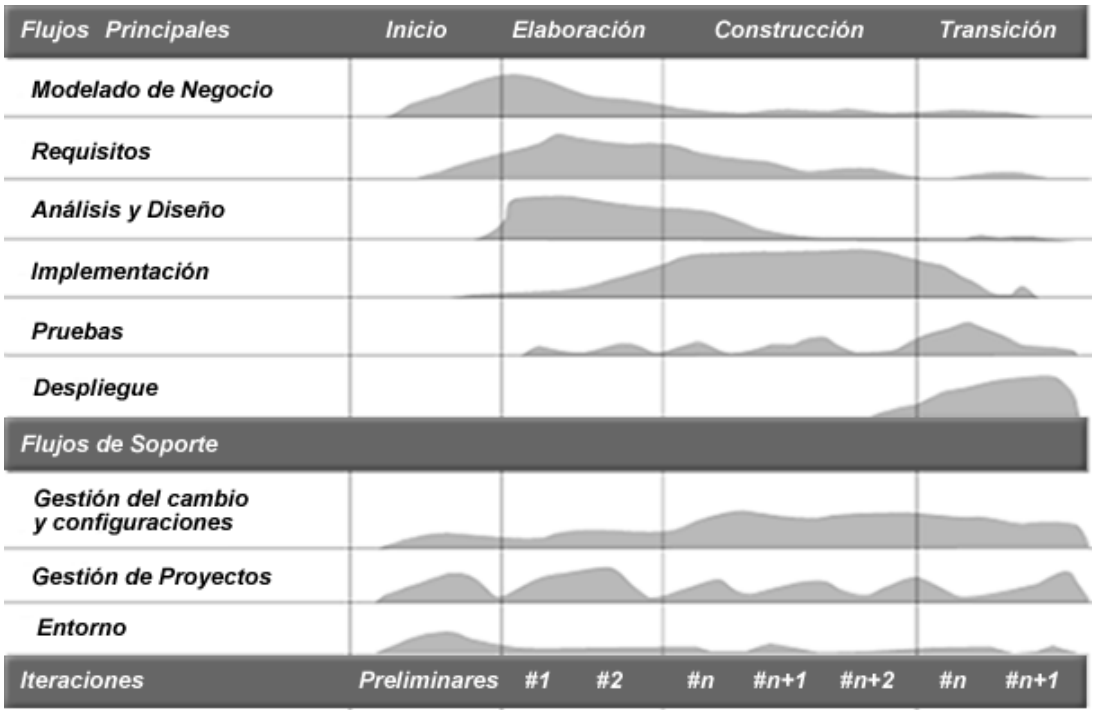

Fig. 1 RUP en Dos Dimensiones.

Flujos de trabajo:

- ✓ **Modelamiento del negocio:** Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- ✓ **Requerimientos:** Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- ✓ **Análisis y diseño:** Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.

- ✓ **Implementación:** Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- ✓ **Prueba (Testeo):** Busca los defectos a lo largo del ciclo de vida.
- ✓ **Instalación:** Produce release del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.
- ✓ **Administración del proyecto:** Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- ✓ **Administración de configuración y cambios:** Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.
- ✓ **Ambiente:** Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

Fases:

- ✓ **Conceptualización (Concepción o Inicio):** Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los CU del sistema.
- ✓ **Elaboración:** Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los CU 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 al alcance definido.
- ✓ **Construcción:** Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene uno o varios release del producto que han pasado las pruebas. Se ponen estos release a consideración de un subconjunto de usuarios.
- ✓ **Transición:** El release ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores (10).

RUP propone que el sistema se realice de una forma iterativa e incremental, teniendo en cuenta los CUs para dirigir el proceso de implementación de las funcionalidades, y una arquitectura base bien definida previa al desarrollo.

1.2.1.2 MSF

MSF es un compendio de las mejores prácticas en cuanto a administración de proyectos se refiere. Más que una metodología rígida de administración de proyectos, MSF es una serie de modelos que puede adaptarse a cualquier proyecto de tecnología de información.

Fases:

- ✓ **Visión y Alcances:** La fase de visión y alcances trata uno de los requisitos fundamentales para el éxito del proyecto, la unificación del equipo detrás de una visión común. El equipo debe tener una visión clara de lo que se quiere lograr para el cliente y ser capaz de indicarlo en términos que motivarán a todo el equipo y al cliente. Se definen los líderes y responsables del proyecto, adicionalmente se identifican las metas y objetivos a alcanzar; estas últimas se deben respetar durante la ejecución del proyecto en su totalidad, y se realiza la evaluación inicial de riesgos del proyecto.
- ✓ **Planificación:** Es en esta fase es cuando la mayor parte de la planeación para el proyecto es terminada. El equipo prepara las especificaciones funcionales, realiza el proceso de diseño de la solución, y prepara los planes de trabajo, estimaciones de costos y cronogramas de los diferentes entregables del proyecto.
- ✓ **Desarrollo:** Durante esta fase el equipo realiza la mayor parte de la construcción de los componentes (tanto documentación como código), sin embargo, se puede realizar algún trabajo de desarrollo durante la etapa de estabilización en respuesta a los resultados de las pruebas. La infraestructura también es desarrollada durante esta fase.

Fig. 2 Fases de la metodología MSF.

- ✓ **Estabilización:** En esta fase se conducen pruebas sobre la solución, las pruebas de esta etapa enfatizan el uso y operación bajo condiciones realistas. El equipo se enfoca en priorizar y resolver errores y preparar la solución para el lanzamiento.
- ✓ **Implantación:** Durante esta fase el equipo implanta la tecnología base y los componentes relacionados, estabiliza la instalación, traspasa el proyecto al personal soporte y operaciones, y obtiene la aprobación final del cliente (9).

Modelo de roles:

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación.

- ✓ **Modelo de Arquitectura del Proyecto:** Diseñado para acortar la planificación del ciclo de vida. Este modelo define las pautas para construir proyectos empresariales a través del lanzamiento de versiones.
- ✓ **Modelo de Equipo:** Este modelo ha sido diseñado para mejorar el rendimiento del equipo de desarrollo. Proporciona una estructura flexible para organizar los equipos de un proyecto.

Puede ser escalado dependiendo del tamaño del proyecto y del equipo de personas disponibles.

- ✓ **Modelo de Proceso:** Diseñado para mejorar el control del proyecto, minimizando el riesgo, y aumentar la calidad acortando el tiempo de entrega. Proporciona una estructura de pautas a seguir en el ciclo de vida del proyecto, describiendo las fases, las actividades, la liberación de versiones y explicando su relación con el Modelo de Equipo.
- ✓ **Modelo de Gestión del Riesgo:** Diseñado para ayudar al equipo a identificar las prioridades, tomar las decisiones estratégicas correctas y controlar las emergencias que puedan surgir. Este modelo proporciona un entorno estructurado para la toma de decisiones y acciones valorando los riesgos que puedan provocar.
- ✓ **Modelo de Diseño del Proceso:** Diseñado para distinguir entre los objetivos empresariales y las necesidades del usuario. Proporciona un modelo centrado en el usuario para obtener un diseño eficiente y flexible a través de un enfoque iterativo. Las fases de diseño conceptual, lógico y físico proveen tres perspectivas diferentes para los tres tipos de roles: los usuarios, el equipo y los desarrolladores.
- ✓ **Modelo de Aplicación:** Diseñado para mejorar el desarrollo, el mantenimiento y el soporte, proporciona un modelo de tres niveles para diseñar y desarrollar aplicaciones software. Los servicios utilizados en este modelo son escalables, y pueden ser usados en un solo ordenador o incluso en varios servidores (11).

1.2.2 Metodologías Ágiles

Luego de varias opiniones tanto a favor como en contra de las metodologías tradicionales se genera un nuevo enfoque denominado, métodos ágiles, que se basa en dos aspectos puntuales, el retrasar las decisiones y la planificación adaptativa; permitiendo potenciar aún más el desarrollo de software a gran escala.

Como resultado de esta nueva teoría se crea un Manifiesto Ágil cuyas principales ideas son:

- ✓ Los individuos y las interacciones entre ellos son más importantes que las herramientas y los procesos empleados.
- ✓ Es más importante crear un producto software que funcione que escribir documentación exhaustiva.
- ✓ La colaboración con el cliente debe prevalecer sobre la negociación de contratos.

- ✓ La capacidad de respuesta ante un cambio es más importante que el seguimiento estricto de un plan (12).

Entre los principales métodos ágiles se encuentra eXtreme Programming (XP), Scrum, Iconix, Cristal Methods, AUP entre otras. Seguidamente se describe XP.

1.2.2.1 XP

XP intenta minimizar el riesgo de fallo del proceso por medio de la disposición permanente de un representante *competente* del cliente a disposición del equipo de desarrollo. El representante debería estar en condiciones de contestar rápida y correctamente a cualquier pregunta del equipo de desarrollo de forma que no se retrase la toma de decisiones, de ahí lo de *competente*.

XP define *UserStories* (historias de usuarios) como base del software a desarrollar. Estas historias las escribe el cliente y describen escenarios sobre el funcionamiento del software, que no solo se limitan a la *Graphical User Interface (GUI)* si no también pueden describir el modelo, dominio, etc. A partir de las UserStories y de la arquitectura perseguida se crea un plan de *releases* entre el equipo de desarrollo y el cliente.

Para cada release se discutirán los objetivos de la misma con el representante del cliente y se definirán las iteraciones (de pocas semanas de duración) necesarias para cumplir con los objetivos de la release. El resultado de cada iteración es un programa que se transmite al cliente para que lo juzgue. En base a su opinión se definen las siguientes iteraciones del proyecto y si el cliente no está contento se adaptará el plan de releases e iteraciones hasta que el cliente de su aprobación y el software este a su gusto.

Junto a los UserStories se encuentran los escenarios de pruebas que describen el escenario contra el que se comprueba la realización de las UserStories. Las UserStories y los casos de pruebas son la base sobre la que se asienta el trabajo del desarrollador.

Como primer paso de cada iteración se escribirán las pruebas, de tal forma que puedan ser ejecutadas automáticamente, de manera que pueda comprobarse la corrección del software antes de cada release. Esto es de vital importancia en XP debido a su apuesta por las iteraciones cortas que generan software que el cliente puede ver y por la refactorización para mejorar el código constantemente, que hacen más que deseable una cantidad considerables de test lo más automatizables posible. Así pues, la funcionalidad concreta del software solo se escribe cuando las pruebas para su corrección estén preparadas.

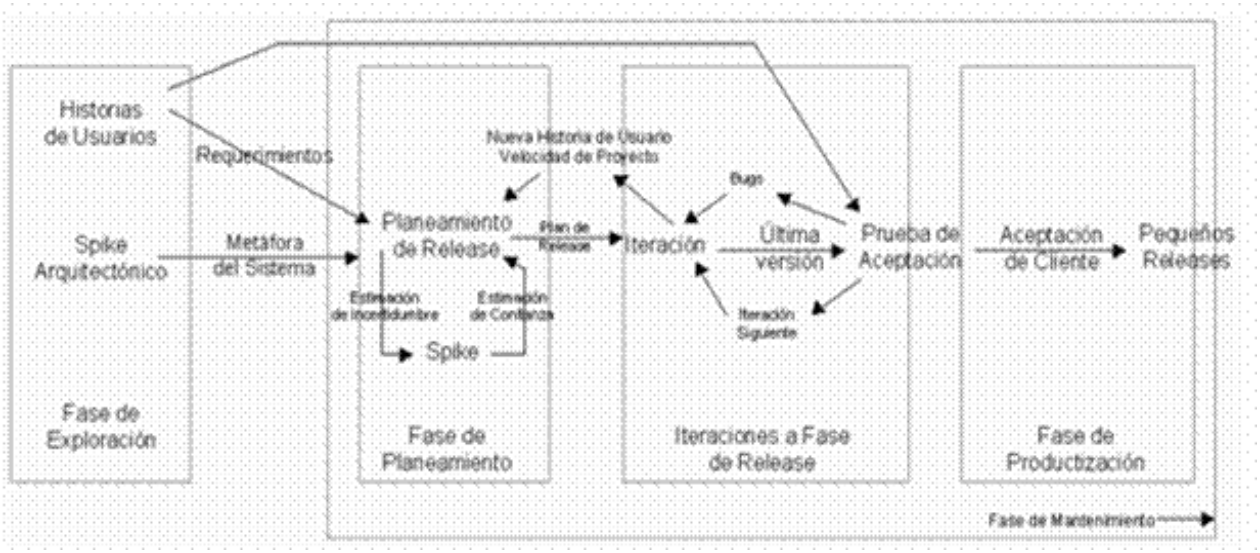

Fig. 3 Esquema de Trabajo XP.

La codificación del software en XP se produce siempre en parejas (dos programadores, un ordenador), por lo que se espera que la calidad del mismo suba en el mismo momento de escribirlo. Al contrario que muchos otros métodos, el código pertenece al equipo en completo, no a un programador o pareja, de forma que cada programador puede cambiar cualquier parte del código en cualquier momento si así lo necesita, dejándose en todo caso las mejoras orientadas al rendimiento para el final. Las parejas no se mantienen para todo el proyecto si no que rotan cíclicamente a lo largo del mismo, tanto en los componentes de la misma como en las partes del software que desarrollan, así cada componente del equipo aprende como trabaja el resto. El objetivo ideal sería que cada componente del equipo trabaje al menos una vez con cada uno de los demás integrantes y con cada componente software, de forma que el conocimiento de la aplicación completa lo posea el equipo entero y no unos pocos miembros.

En XP se programará solo la funcionalidad que es requerida para la release actual. Es decir, una gran flexibilidad y capacidad de configuración solo será implementada cuando sea necesaria para cumplir los requerimientos del release. Se sigue un diseño evolutivo con la siguiente premisa: conseguir la funcionalidad deseada de la forma más sencilla posible. De ahí una variación educada del famoso *Keep It Simple Stupid* (*KISS*), *Keep things as simple as possible* (mantén las cosas tan sencillas como sea posible). Este diseño evolutivo hace que no se le dé apenas importancia al análisis como fase independiente, puesto que se trabaja exclusivamente en función de las necesidades del momento (13).

1.2.3 Selección de la metodología a utilizar

Usar alguna de las metodologías ágiles no sería muy recomendado, debido a la magnitud y complejidad del software a desarrollar, ya que estas, para proyectos de gran envergadura pierden de vista el proceso de desarrollo y con él el control del avance real del proyecto. Debido a la distancia que existe entre los clientes y los desarrolladores, y sumado a que existe un presupuesto y un tiempo de entrega, es más rentable en el desarrollo, mantener el equipo del proyecto en el país, por tanto, el cliente o el usuario no se puede convierte en miembro del equipo.

De las dos metodologías tradicionales mencionadas no se usará MSF debido a que los precios de licencias, capacitación y soporte por parte de Microsoft son caros, por tanto, se selecciona RUP.

RUP, junto a UML, constituyen la metodología estándar más usada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos. RUP es más apropiada para proyectos grandes, dado que requiere un equipo de trabajo capaz de administrar un proceso complejo en varias etapas. En proyectos pequeños, es posible que no sea viable 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. El control de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción.

1.3 UML

UML es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas.

Pueden ser artefactos: un modelo, una descripción que comprende el desarrollo de software que se basen en el enfoque orientado a objetos, utilizándose también en el diseño web. UML usa procesos de otras metodologías, aprovechando la experiencia de sus creadores, eliminó los componentes que resultaban de poca utilidad práctica y añadió nuevos elementos.

El lenguaje UML tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático: desde el análisis con los CUs, el diseño con los diagramas de clases, objetos, etc., hasta la implementación y configuración con los diagramas de despliegue.

UML es desde finales de 1997, un lenguaje de modelado orientado a objetos estándar, de acuerdo con el Object Management Group (OMG), siendo utilizado diariamente por grandes organizaciones como: Microsoft, Oracle, Rational (14).

1.4 Propuesta de sistema

Una aplicación web es un sistema informático que los usuarios utilizan accediendo a un servidor web a través de internet o de una intranet. Las aplicaciones web son populares debido a la practicidad del navegador web como cliente ligero, lo que reduce las necesidades de recursos de las estaciones de trabajo. Se destaca también la facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software.

El capital humano para el mantenimiento del sistema se reduce, pues no es necesaria la instalación ni configuración individualizada. Es importante destacar que facilita la implementación de los mecanismos de seguridad, como la autentificación de Internet Protocol (IP) y el monitoreo de los accesos. Permite tener la información centralizada, por lo que no es necesario realizar réplicas periódicamente. En cuanto al trabajo con información, con sistemas web se obtienen mejores resultados, pues las intranets son una vía efectiva y barata de proporcionar información a toda la organización para la cual se efectúa.

Pero la mayor ventaja de los sistemas web se encuentra en la facilidad para determinar la ubicación exacta de cualquier documento, más aún si estos se encuentran en formato digital.

La Figura 4 muestra la estructura de la aplicación web que dará solución a la propuesta para el sistema de gestión SIIPOL.

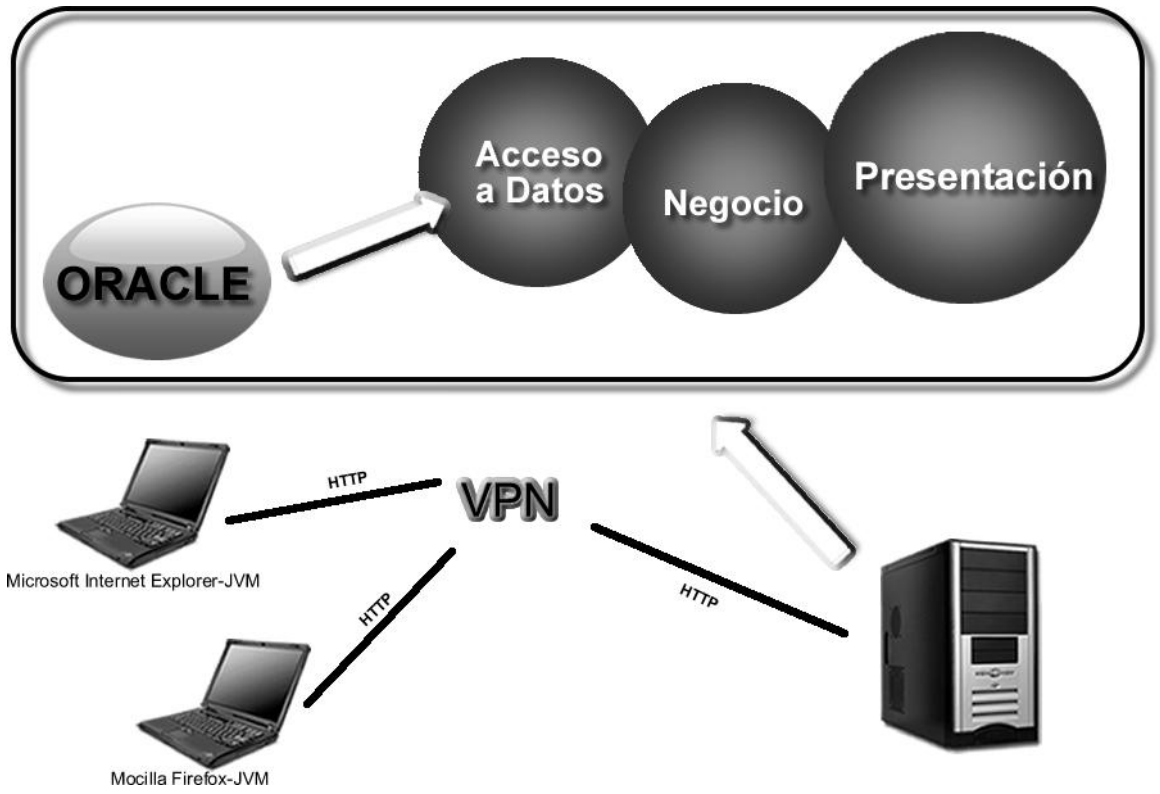

Fig. 4 Cliente-Servidor.

La Figura 5 muestra la arquitectura usada para el desarrollo del sistema SIIPOL, basada en el estilo n capas, usando el patrón 3 capas.

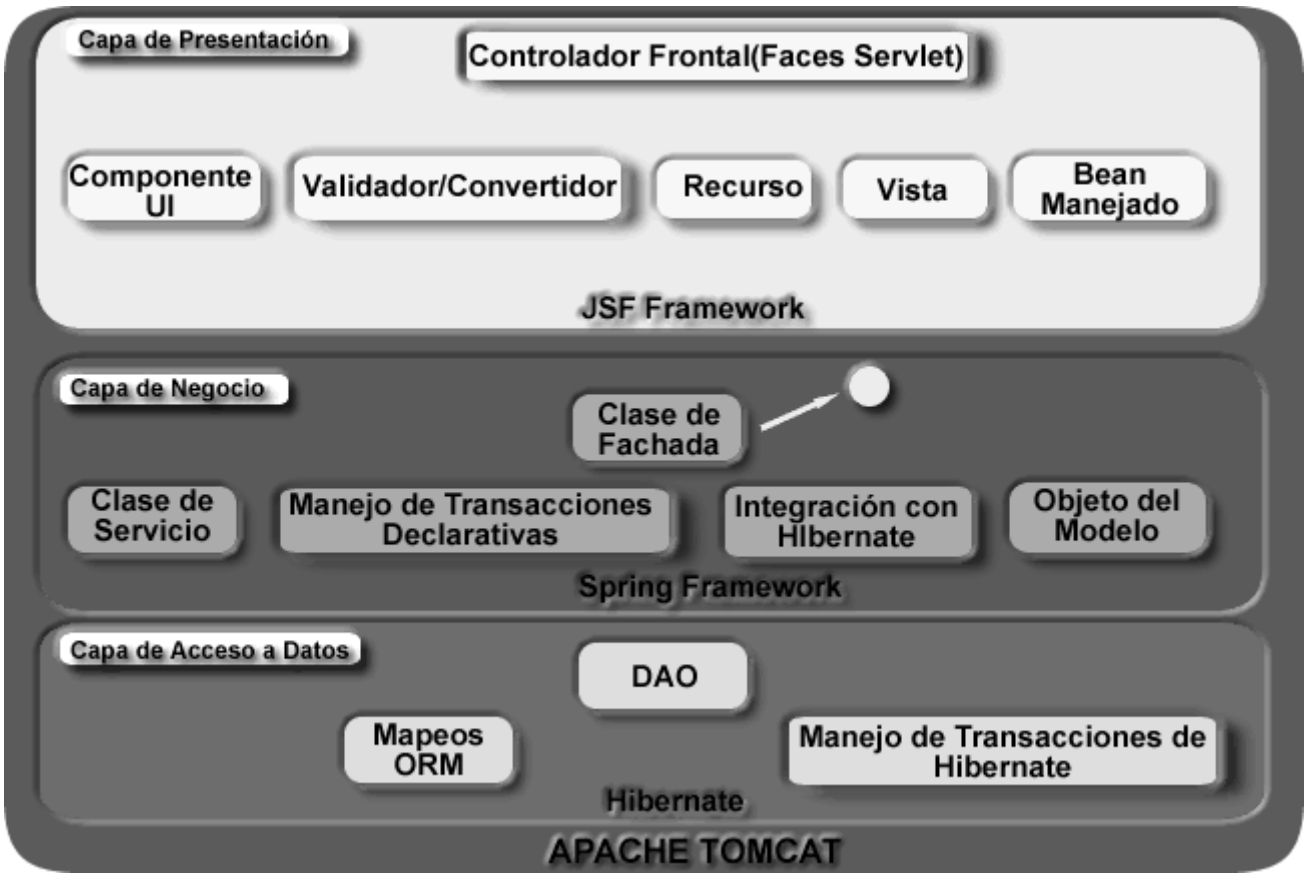

Fig. 5 Patrón 3 capas.

1.5 Análisis de las principales plataformas de desarrollo. Selección de la plataforma de desarrollo que apoya la solución del problema

Muchas son las alternativas de desarrollo para una aplicación web: PHP, páginas ASP, ColdFusion, etc. y la mayoría de ellas están actualmente en un estado tecnológico excelente. Hay algo que las diferencia y es su orientación real hacia un modelo de desarrollo de componentes empresarial. Esto quiere decir que la plataforma tecnológica tiene que servir y propiciar, desde el entorno de ejecución, hasta el entorno de desarrollo.

Las plataformas que existen para el desarrollo de este tipo de aplicaciones son: J2EE (Java 2 Interprise Edition) de Sun Microsystems y .NET de Microsoft.

1.5.1 La plataforma J2EE

Antes de entrar en el análisis de J2EE es necesario tener una breve idea acerca de lo que es Java.
Java es un lenguaje de programación orientado a objeto desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.
Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible (15).
La plataforma J2EE 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 J2EE. Un componente de la 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. La plataforma 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.
J2EE brinda, entre otras, las siguientes ventajas:
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.
Organismo de control: J2EE 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.
Madurez: creada en el año 1997, J2EE ya tiene varios años de vida y una amplia cantidad de proyectos importantes a sus espaldas.
Soluciones libres: sobre la plataforma J2EE es posible crear arquitecturas basadas por completo en productos de software libre. No solo eso, sino que los arquitectos de software disponen de muchas soluciones libres para cada una de las partes de su arquitectura (16).

1.5.2 La plataforma .NET

Primeramente es necesario conocer que es Microsoft.NET.

Microsoft.NET es un entorno de ejecución común que sale al mercado en el año 2000, por supuesto, desarrollado por Microsoft. Es un conjunto de nuevas tecnologías que podrían resumirse en las siguientes: Plataforma .NET, SDK de la plataforma .NET, Visual Studio.NET, Servicios Web y Servidores para empresas.

La plataforma .NET es una capa de software que se coloca entre el Sistema Operativo (SO) y el programador y que abstrae los detalles internos del SO. Las características fundamentales de esta plataforma son:

Portabilidad: Debido a la abstracción del programador respecto al SO, una aplicación .NET puede ser ejecutada en cualquier SO de cualquier máquina que disponga de una versión de la plataforma.

Multilenguaje: Estas aplicaciones pueden escribirse en cualquiera de los múltiples lenguajes que ofrece .NET (Visual C#.Net, Visual Basic.NET...). En lugar de compilarse a código máquina, se compila a un lenguaje intermedio llamado Microsoft Intermediate Language o MSIL (Lenguaje Intermedio de Microsoft).

Interoperabilidad: La interoperabilidad entre diferentes segmentos de código escritos en diferentes lenguajes es total.

Microsoft define la plataforma .NET como “un entorno para la construcción, desarrollo y ejecución de servicios web y otras aplicaciones que consiste en tres partes fundamentales: el Common Language Runtime (entorno de ejecución), las Framework Classes (clases de la plataforma) y ASP.NET (17).

1.5.3 Selección de la plataforma a utilizar

Con el estudio de estas dos plataformas se ha llegado a la conclusión de que ambas tienen un fin común pero metodologías sustancialmente diferentes a la hora de abordar problemas, como la portabilidad, seguridad, etc. En el aspecto funcional ambas plataformas guardan varias similitudes; se programa en un lenguaje que luego se compila a un código intermedio. Este código se ejecutará en un “entorno de ejecución” que transformará el lenguaje intermedio a código propio de la máquina en la que se corre la aplicación.

Las dos plataformas tienen diferencias entre las que se pueden mencionar las siguientes:

.NET es un producto de Microsoft, mientras que J2EE es un conjunto de especificaciones que definen un estándar desarrollado por Sun Microsystems secundada por muchas más empresas como IBM, HP, BEA, ORACLE, etc.

.NET cuenta con un ambiente de desarrollo llamado Visual Studio .NET, mientras que en J2EE cada empresa ofrece su entorno de desarrollo, tales como NetBeans de Sun Microsystems, Visual Café de WebGain, Visual Age for Java de IBM, IBM WebSphere, Eclipse, entre otros.
En J2EE se puede crear una arquitectura para un software empresarial de altas prestaciones con soluciones completamente libres, lo cual, sobre .NET es completamente imposible.
Luego del análisis de ambas plataformas y de las necesidades del cliente, desarrollar el software en una plataforma, que se ajustara a los términos legales venezolanos, se toma la decisión de utilizar J2EE.

1.6 Herramientas a utilizar

1.6.1 IDE de desarrollo

Integrated Development Environment (IDE) es un programa compuesto por un conjunto de herramientas para un programador. Son múltiples los IDE que existen en el mercado ofreciendo entornos adecuados para el trabajo sobre la plataforma J2EE, tales como, NetBeans de Sun Microsystems, Visual Café de WebGain, Visual Age for Java de IBM, Eclipse de la Fundación Eclipse, entre otros. En la selección del IDE se analizaron NetBeans y Eclipse.
Cuando comenzó la preparación del proyecto la versión estable que existía era NetBeans 5.0 (surge en enero 2006). Esta versión no brindaba un soporte eficiente para el trabajo con Struts y JSF, sumado a que eran pocos, en la UCI, los que conocían a fondo la herramienta. Por estos motivos, la selección fue Eclipse. Se puede agregar, además, que el Proyecto Sistema de Gestión Penitenciaria (SIGEP), de la Facultad 4, ha montado con éxito toda su arquitectura usando Eclipse.

1.6.1.1 Eclipse

Eclipse es un IDE multiplataforma libre para crear aplicaciones clientes de cualquier tipo. La primera y más importante aplicación que ha sido realizada con este entorno es el afamado IDE Java llamado *Java Development Toolkit* (JDT). Eclipse emplea módulos (en inglés *plug-in*) para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Asimismo, a través de los módulos, libremente disponibles, es posible añadir: control de versiones con Subversión, vía Subclipse; integración con Hibernate, vía

Hibernate Tools. Además se puede desarrollar código javascripts, integrándolo con Aptana y trabajar con JSF, integrándolo con Exadel.

1.6.2 Herramientas Case

Se puede definir a las Computer Aided Software Engineering (CASE) como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software (18).

En el mundo existen numerosas herramientas CASE pero las dos más usadas son Rational Rose y Visual Paradigm. Cuando comenzó la preparación del proyecto, la versión de Rational Rose, no incluía soportes de generación de código Java (actualmente existe el IBM Rational Rose Developer para Java). Por esta razón, se selecciona la otra herramienta más usada: Visual Paradigm.

1.6.2.1 Visual Paradigm

Visual Paradigm, al igual que Rational Rose, soporta el ciclo de vida completo del desarrollo de software, está diseñada para un gran número de usuarios, incluyendo ingenieros de sistemas, analistas de sistemas, analistas de negocio, arquitectos de sistemas. Visual Paradigm se integra con IDE (Eclipse, JBuilder, NetBeans, IntelliJ IDEA, JDeveloper and WebLogic Workshop) para soportar la fase de implementación. La transición desde el análisis al diseño y después a la implementación es cuidadosamente integrada. Incluye además, un conjunto de herramientas que soportan Object-Relational Mapping (ORM), como Hibernate, lo cual incluye generación de código completamente orientado a objetos, listo para usar librerías, para obtener y modificar registros de una gran variedad de gestores de base de datos, y la sincronización entre los diagramas de clases y los diagramas de entidad-relación (ERD). Presenta además generación de código e ingeniería inversa. Permite generar los Enterprise Java Beans (EJB) y así mismo los descriptores de despliegue para varios servidores de aplicación. Distinto a muchas herramientas de modelado, pueden extenderse sus diagramas hechos desde el Visio y de Rational Rose.

1.7 Análisis de los principales Frameworks. Selección del Frameworks a utilizar

Recientemente, el interés en reutilizar software ha ido cambiado, de la reutilización de componentes simples a diseño de sistemas enteros o estructuras de aplicaciones. Un sistema software que pudiera ser reutilizado para la creación de aplicaciones completas es llamado framework.

Los frameworks deberían permitir la producción fácil de un conjunto de sistemas específicos pero similares, dentro de un cierto dominio, comenzando desde una estructura genérica. Por lo anteriormente expresado, se puede afirmar que son arquitecturas genéricas integradas por un extensible conjunto de componentes.

Antes de mencionar los frameworks analizados para trabajar en la capa de presentación se debe hablar de la Tecnología JSP- Servlet y del patrón de diseño Model-View-Controller (modelo-vista-controlador).

1.7.1 Tecnología JSP-Servlet

Los Servlets no son más que clases Java que implementan la interfaz HttpServlet del API Java Servlet. Esta interfaz define un conjunto de métodos cíclicos que pueden anularse para proporcionar respuestas dinámicas a solicitudes HTTP. Un servidor de aplicación compatible con J2EE proporciona a los servlets un entorno en el que residir y administrar las solicitudes entrantes.

El mayor problema con el que se enfrentaron los servlets fue que requerían al desarrollador Java ensamblar la salida con formato HTML desde dentro del código Java utilizando series de sentencias *out.println( )*. De este modo no sólo se generaba código innecesario, sino que también se hacía complicado crear una UI amigable. Si el usuario deseaba utilizar una herramienta para diseñar visualmente una página necesitaba copiar el HTML en su código Java y usar las llamadas al método *out.println( )* en cada línea. De hecho, no existía una distinción clara entre el código de la aplicación y la UI. Para cambiar uno, invariablemente era necesario modificar el otro.

Java Server Pages (JSP) fue concebido como un modo de separar el contenido propiamente dicho de la presentación del mismo. Una página JSP suele ser una página HTML con etiquetas especiales para añadir código Java. La página se compila dinámicamente en un servlet (en segundo plano) y se ejecuta como tal. Esto hace posible escribir HTML puro (y utilizar herramientas HTML) sin que importe el código Java existente en la página (19).

1.7.2 Modelo Vista-Controlador

Model-View-Controller, más conocido como MVC, es un patrón de diseño clásico utilizado con frecuencia en aplicaciones que necesiten la habilidad de mantener múltiples vistas de la información.

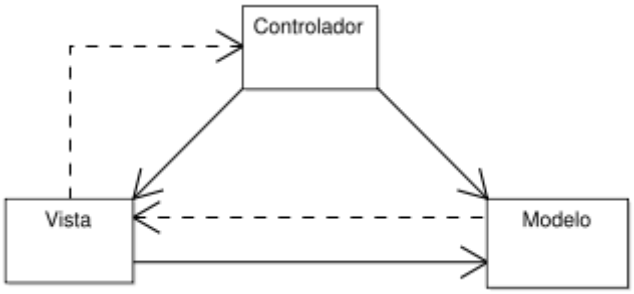

Fig. 6 Esquema del patrón MVC.

El Modelo: Es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.

La Vista: Es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa con el Modelo a través de una referencia al propio Modelo.

El Controlador: Es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo.

1.7.2.1 Modelo 2

El Modelo 2 es una variación específicamente para J2EE realizada por Sun Microsystems al clásico patrón MVC, el cual introdujo el patrón Front Controller.

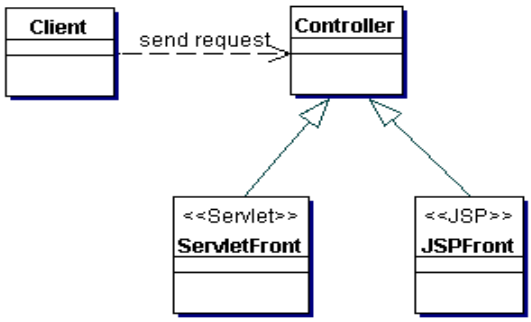

Fig. 7 Esquema del patrón Front Controller.

El patrón Front Controller es un objeto que acepta todos los requerimientos de un cliente y los direcciona a manejadores apropiados. Podría dividir la funcionalidad en dos objetos diferentes: el Front Controller y el Dispatcher. En ese caso, el Front Controller acepta todos los requerimientos de un cliente y realiza la autenticación, y el Dispatcher direcciona los requerimientos a manejadores apropiados.

1.7.3 Struts

Struts es un subproyecto del proyecto Apache, un framework libre de código abierto para la creación de aplicaciones web en Java. Se puede decir que Struts ha sido el framework que ofreció un camino para el MVC en Java para desarrollos web. El núcleo de Struts es una capa flexible de control basada en las tecnologías estándares. Este framework soporta, como se menciona anteriormente, arquitecturas basadas en la metodología del Modelo 2.

El framework provee tres componentes primordiales:

- ✓ Un manipulador de solicitudes provisto por la aplicación.
- ✓ Un manipulador de respuestas que transfiere el control a otro recurso que completa la respuesta.
- ✓ Una librería de etiquetas, o tags, que ayuda a los desarrolladores a crear aplicaciones interactivas basadas en formularios con páginas JSP del lado del servidor.

Los ficheros eXtensible Markup Language (XML) están siendo comúnmente utilizados como archivos de configuración de las aplicaciones y Struts no es una excepción. Los dos ficheros XML de configuración para Struts son el WEB.XML, el cual es usado en la configuración de la aplicación web y el STRUTS-CONFIG.XML, el cual es utilizado para configurar todas las acciones que llevará a cabo la aplicación en Struts.

Al utilizar Struts primeramente los desarrolladores se encuentran con los beneficios del MVC. Sin embargo, Struts ofrece un significativo número de ventajas además de lo anteriormente dicho. He aquí un sumario de las mismas:

- ✓ Configuración centralizada basada en archivos: Muchos de los valores de Struts están representados en XML y archivos de propiedades, lo que significa que muchos cambios puede ser realizados sin necesidad de modificar o recompilar código Java; es suficiente editar un simple archivo. Esto, además, permite a los desarrolladores enfocarse en sus tareas específicas.
- ✓ Librerías de etiquetas o tags: Permite que se puedan crear interfaces JSP rápidamente y de menor peso para el servidor.
- ✓ Struts es libre y de código abierto: Puede ser utilizado libremente y su código puede ser modificado en dependencia de las necesidades del usuario, además, permite ver como fue programado.
- ✓ Todos los componentes y clases permiten ser heredados y son reemplazables.
- ✓ Validación de formularios: Struts contiene capacidades para chequear y validar la entrada de valores.
- ✓ Amplio tiempo de vida: Dado a que Struts existe hace varios años, es posible y no difícil encontrar desarrolladores experimentados en dicho framework (20).

Además de tener muchas otras ventajas, Struts puede convertirse en algo bien complejo, lo que trae sus inconvenientes, algunos de los cuales son citados a continuación:

- ✓ Una gran curva de aprendizaje: Es necesario tener amplio conocimiento acerca de las APIs de Java dedicadas a la web. Esta inconveniencia es específicamente significativa para proyectos pequeños con desarrolladores poco experimentados, en los que se puede emplear mucho más tiempo estudiando Struts que desarrollando el proyecto.
- ✓ Enfoque rígido: El lado opuesto del beneficio de que Struts tiene un apoyo consistente a la metodología del MVC es que Struts hace muy difícil (pero no imposible) al desarrollador utilizar otras metodologías.
- ✓ Poca transparencia: El código de las aplicaciones en Struts es difícil de comprender a simple vista y difícil de comparar y optimizar.

Además de esto, Struts no manipula eventos del lado del cliente, lo que provoca que tengan que ser implementados por el usuario con otro lenguaje (22).

1.7.4 JSF

JSF es el estándar presentado por Sun Microsystems para la capa de presentación web. Se rige, como una evolución natural, de los frameworks actuales hacia un sistema de componentes. Es un framework sencillo que aporta los componentes básicos de las páginas web, además permite crear componentes más complejos, tales como menús, árboles, pestañas, etc. Hoy por hoy existe la disponibilidad de diferentes implementaciones de JSF, tanto comerciales como libres y de código abierto, así como librerías de componentes adicionales que amplían la funcionalidad del framework. La principal característica de JSF es la simplificación en el desarrollo de interfaces de usuario en J2EE.

Además de esto, **JSF incluye**:

- ✓ Un conjunto de APIs para representar componentes de una UI y administrar su estado, manejar eventos, validar la entrada de datos, definir un esquema de navegación de las páginas y dar soporte para internacionalización y accesibilidad.
- ✓ Un conjunto por defecto de componentes de UI.
- ✓ Dos librerías de etiquetas personalizadas para JSP que permiten expresar una interfaz JSF dentro de una página JSP.
- ✓ Un modelo de eventos del lado del servidor.
- ✓ Administración de estados.
- ✓ Java Beans o Beans manipulados.
- ✓ Lenguaje unificado de expresiones, tanto para JSP como para JSF, las que al final compilan como JSPs.

Ventajas para el desarrollador:

- ✓ Básicamente el modelo de componentes de JSF hace muy simple el código.
- ✓ Puede usarse un lenguaje basado en expresiones en cualquier lugar y en cualquier momento.
- ✓ La habilidad para manipular el árbol de componentes de UI en Java antes de que la página se construya en el cliente hace el dinamismo de las páginas realmente simple.
- ✓ No se requiere de un entorno de desarrollo específico para construir páginas JSF basadas en la tecnología de JSP, aunque el uso de un entorno de desarrollo apropiado simplificaría mucho las cosas.
- ✓ Manipulación de eventos del lado del servidor.

Desventajas:

- ✓ JSF es una tecnología muy nueva y aún no ha alcanzado el grado de madurez de otras tecnologías, lo que hace que los desarrolladores sientan recelos con respecto a ella.
- ✓ La documentación existente no es abundante y es bien difícil contactar personal capacitado en la tecnología.
- ✓ No tiene implementados mecanismos de seguridad (22).

1.7.5 Framework a utilizar

En el proyecto CICPC se decide usar JSF por las siguientes razones:
Al igual que Struts, JSF pretende normalizar y estandarizar el desarrollo de aplicaciones web. Hay que tener en cuenta que JSF es posterior a Struts y, por lo tanto, se ha nutrido de la experiencia de este, algunas de sus deficiencias. De hecho el creador de Struts (Craig R. McClanahan) también es el líder de la especificación de JSF (21).
Una de las grandes ventajas de la tecnología JSF es que ofrece una clara separación entre el comportamiento y la presentación. Las aplicaciones web construidas con tecnología Struts conseguían parcialmente esta separación. Sin embargo, una aplicación Struts no puede mapear peticiones HTTP al manejo de eventos específicos de los componentes o manejar elementos UI como objetos con estado en el servidor.
Separar la lógica de negocio de la presentación también permite que cada miembro del equipo de desarrollo de la aplicación web se centre en su parte asignada del proceso diseño, y proporciona un modelo sencillo de programación para enlazar todas las piezas. Por ejemplo, personas sin experiencia en programación pueden construir páginas JSF usando las etiquetas de componentes UI que esta tecnología ofrece, y luego enlazarlas con código de la aplicación sin escribir ningún script. Por último, el soporte de JSF en IDEs como Eclipse, Netbeans, etc. es mucho mejor (22).
En el proyecto se utiliza MyFaces (MyFaces Core 1.15 y MyFaces Tomahawk 1.14), las implementaciones libres y abiertas de JSF de la fundación Apache, brindando una mayor funcionalidad al usuario. También se utiliza RichFaces (RichFaces 3.0) que es una biblioteca de componentes que permite la integración fácil de capacidades AJAX en el desarrollo de la aplicación web.

1.8 Tecnologías del lado del cliente

1.8.1 HTML

HyperText Markup Language (HTML) es un lenguaje de composición de documentos y especificación de ligas de hipertexto que define la sintaxis y coloca instrucciones especiales que no muestra el navegador, aunque si le indica como desplegar el contenido del documento, incluyendo texto, imágenes y otros medios soportados. HTML también le indica cómo hacer un documento interactivo a través de ligas especiales de hipertexto, las cuales conectan diferentes documentos, ya sea en su computadora o en otras, así como otros recursos de internet, como FTP y Gopher (23).

Aunque HTML es un lenguaje que utilizan los ordenadores y los programas de diseño, es muy fácil de aprender y escribir por parte de las personas. Por tanto, HTML es un lenguaje muy sencillo que permite preparar documentos web insertando una serie de marcas (tags) que controlan los diferentes aspectos de la presentación y comportamiento de sus elementos.

1.8.2 CSS

Cascading Style Sheets (CSS) es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas (24).

Los beneficios de usar CSS son dobles. Por un lado, se evita hacer a los archivos demasiado pesados (excluyendo el largo código requerido para las tablas anidadas y el añadido de características gráficas), y se define el "estilo visual" de un sitio entero sin necesidad de hacerlo etiqueta por etiqueta, para cada una de las páginas.

Por otro lado, se trabaja con estándares, y se separa hasta cierto punto la estructura, el código de la presentación, logrando una manera más nítida de trabajar, y lo que es más: en un documento sencillo CSS, se define lo que se llamaría una "plantilla gráfica" para todo un sitio, cualquier cambio hecho a un estilo CSS, se reflejará en todos los elementos que sean referidos a éste, automáticamente, con sólo editar un sencillo documento CSS.

1.8.3 JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios (25).

Con Javascript se puede crear efectos especiales en las páginas y definir interactividades con el usuario. El navegador del cliente es el encargado de interpretar las instrucciones Javascript y ejecutarlas para realizar estos efectos e interactividades, de modo que el mayor recurso, y tal vez el único, con que cuenta este lenguaje es el propio navegador.

Javascript es el siguiente paso, después de HTML, que puede dar un programador de la web que decida mejorar sus páginas y la potencia de sus proyectos. Es un lenguaje de programación bastante sencillo y pensado para hacer las cosas con rapidez, a veces con ligereza. Incluso las personas que no tengan una experiencia previa en la programación podrán aprender este lenguaje con facilidad y utilizarlo en toda su potencia con sólo un poco de práctica.

Javascript es un lenguaje con muchas posibilidades. Permite la programación de pequeños scripts, pero también de programas más grandes, orientados a objetos, con funciones, estructuras de datos complejas, etc. Además, pone a disposición del programador todos los elementos que forman la página web, para que pueda acceder a ellos y modificarlos dinámicamente. Con Javascript el programador se convierte en el verdadero dueño y controlador de cada cosa que ocurre en la página, cuando la está visualizando el cliente.

1.8.4 AJAX

El término AJAX es un acrónimo de *Asynchronous JavaScript* + *XML*, que se puede traducir como "JavaScript asíncrono + XML". AJAX es una combinación de tres tecnologías ya existentes:

Las tecnologías que forman AJAX son:

- ✓ XHTML y CSS: Para crear una presentación basada en estándares.
- ✓ DOM: Para la interacción y manipulación dinámica de la presentación.
- ✓ XML, XSLT y JSON: Para el intercambio y la manipulación de información.
- ✓ XMLHttpRequest: Para el intercambio asíncrono de información.
- ✓ JavaScript: Para unir todas las demás tecnologías.

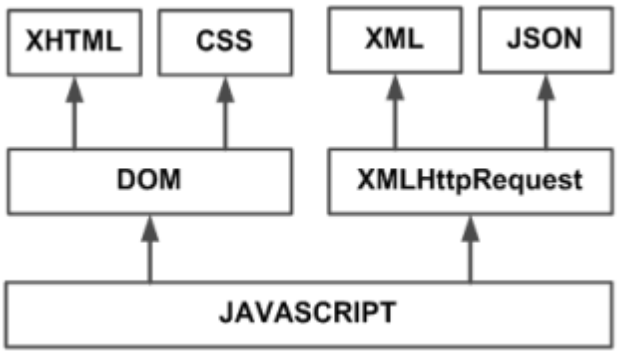

Fig. 8 Tecnologías agrupadas bajo el concepto de AJAX.

AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano.

Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. La nueva capa intermedia de AJAX mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor (25).

1.9 Análisis de las metodologías para el diseño y desarrollo de las Interfaces de Usuario. Selección de la metodología a utilizar

En la actualidad existen algunas metodologías definidas para el diseño y montaje de interfaces de usuario, pero ninguna cumple con las expectativas que se necesita en el proyecto CICPC. Se pueden mencionar las siguientes.

- ✓ Interfaces de Usuarios Inteligentes.
- ✓ Metodología y Técnicas en Proyectos Software para la Web.
- ✓ Diseño Centrado en el Usuario.

Dichas metodologías centran su atención, por lo general, en la adaptación de los sistemas a los usuarios, esto tiene suma importancia, sobre todo cuando se quiere facilitar la interacción del usuario con el sistema. De las existentes se pueden tomar algunos aspectos, pero ninguna es la que se busca, ya que el objetivo principal no es construir un sistema inteligente de los que son capaces de

adaptarse a cada tipo de usuario, pues SIIPOL es un sistema que será utilizado por un grupo de usuarios específicos.

En la UCI, no existe ni se ha creado, una metodología que estandarice el diseño y desarrollo del montaje de UI. Generalmente, son los propios programadores los que realizan esta función, sin tener ningún tipo de especialización en el tema, por lo que pocas veces se entregan los prototipos funcionales, cumpliendo con los requisitos del cliente o con las pautas definidas por los diseñadores. Por estas razones, el proyecto decidió realizar una metodología que reúna todos los requisitos necesarios.

La metodología que se va a implementar, ha sido desarrollada por un equipo del proyecto CICPC, esta se ocupa de estandarizar las buenas prácticas en el diseño y desarrollo del montaje de UI, para optimizar el trabajo en la capa de UI en las aplicaciones web. Su nombre es DDWUI.

Conclusiones

Como conclusiones del capítulo se comprobó la utilidad de los sistemas de gestión policial en la actualidad. Con el objetivo de justificar la selección de las metodologías, herramientas y tecnologías, sobre las cuales se apoya la propuesta, se concluyó que son las apropiadas, acorde con las características del sistema a desarrollar.

CAPÍTULO 2

DESARROLLO DE LA SOLUCIÓN PROPUESTA

Introducción

En el presente capítulo se desarrollarán distintos diagramas, como el de clases del análisis, clases de diseño y el de contrato entre paquetes, que se generan en el flujo de trabajo Análisis y Diseño. Se describe, paso a paso, las fases de la metodología escogida para el diseño y montaje de las interfaces del módulo, para luego proseguir con el flujo de trabajo Implementación, donde se estructuran los componentes del subsistema de implementación de la capa UI.

2.1 Modelado del Sistema

2.1.1 Modelo de CU del sistema

SIIPOL es un sistema compuesto por varios módulos funcionales, cada uno con un objetivo y una función independiente, lo cual no implica que no exista una estrecha interacción entre los mismos. El **Módulo Investigación en Ciencias Forenses**, es el encargado de gestionar toda la información asociada a las investigaciones forenses, desde la solicitud y realización de experticias, control del proceso de investigación, manejo de la información asociada a cadáveres y otras. Se divide a su vez en tres submódulos más:

Control de Investigación Forense: Este submódulo agrupa todas las funcionalidades para controlar la información referente al proceso de investigación forense, está concebido para ser utilizado principalmente por los directivos de CICPC, pues desde él se podrá acceder a toda la información relacionada con las experticias realizadas en uno o varios despachos.

Experticias: Este submódulo agrupa todas las funcionalidades necesarias para la realización de los informes de experticias forense, desde él se pueden ver las nuevas solicitudes de experticia forense asignadas a un funcionario determinado hasta el envío del informe y su almacenamiento para posteriores consultas.

Solicitudes: Es el encargado de hacer posible la realización de una solicitud de experticia forense y de manejar toda la lógica asociada a este proceso.

El módulo cuenta con un total de veintiún CU, repartidos entre los tres subsistemas. Se presenta a continuación el modelo de CU.

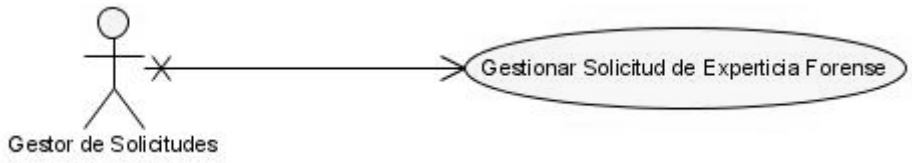

Fig. 9 Diagrama de CU subsistema solicitudes.

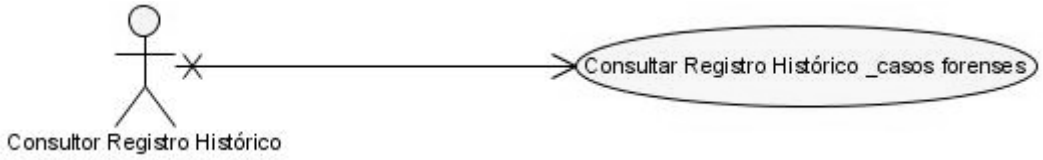

Fig. 10 Diagrama de CU subsistema control de investigación.

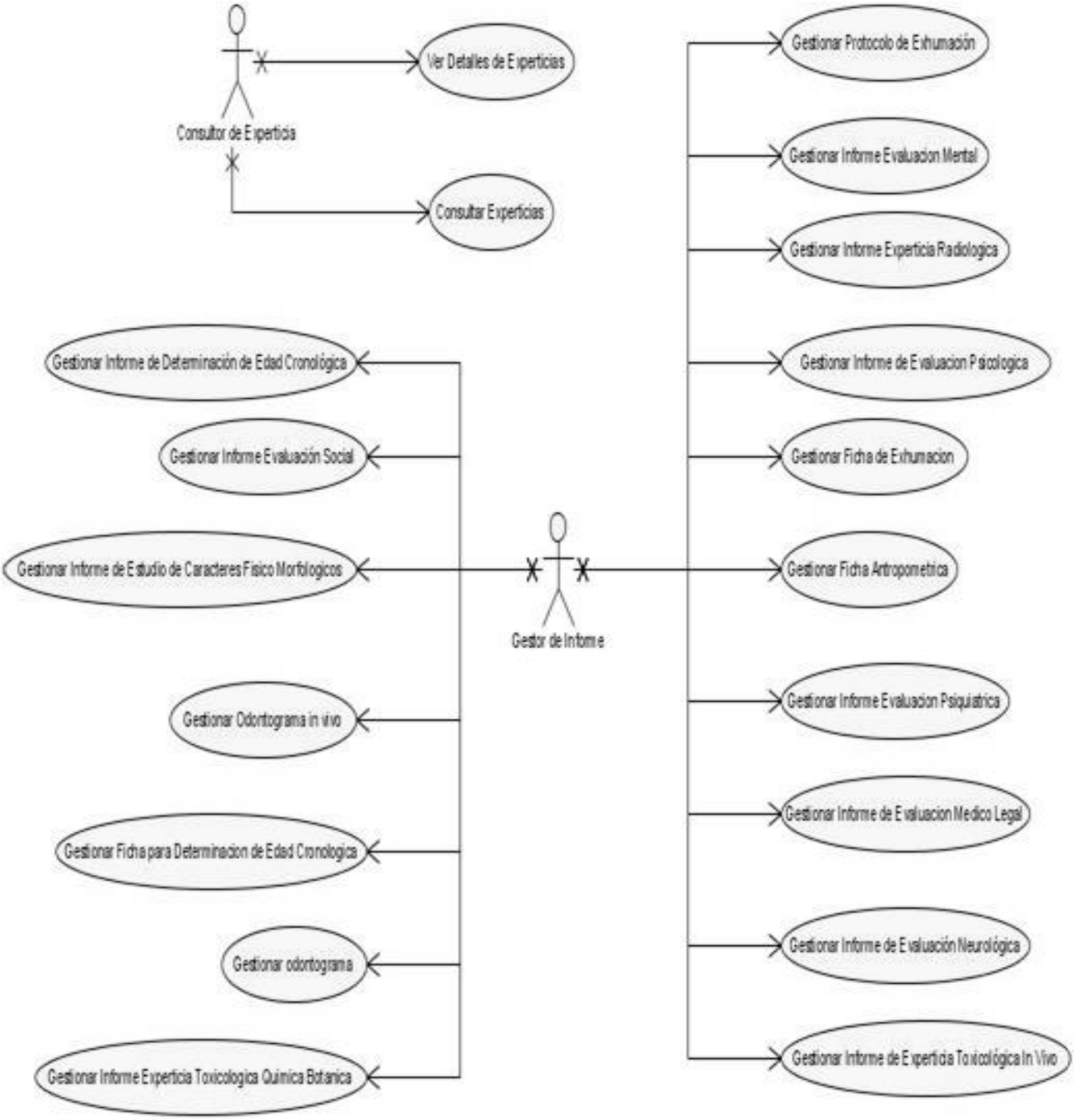

Fig. 11 Diagrama de CU subsistema experticias.

2.1.2 Descripción de los CU del sistema

A continuación se presenta la descripción de algunos de los CU más significativos del Módulo de Investigación en Ciencias Forenses. La totalidad de las descripciones pueden ser vistas en el **Anexo # 1.**

Tabla 1 Descripción del CU Ver Detalles de Experticia.

Nombre del CU	Ver Detalles de Experticia
Actor (es)	Consultor de Experticias
Descripción	El caso de uso se inicia cuando el actor selecciona la opción que le permite ver toda la información relativa a la realización de una experticia. El sistema muestra los datos de la Solicitud de Experticia Forense, así como un listado con todos los Informes de Experticia Forense que se han generado durante la realización de la experticia. El sistema permite ver una vista previa de la información de la experticia, con posibilidad de imprimir y exportar a PDF.
Referencia	1. RF Ver datos de una Solicitud de Experticia Forense. 2. RF Ver datos de la Evidencia y/o Persona asociadas a una Solicitud de Experticia Forense. 3. RF Ver el identificador del Acta Procesal y/o Expediente Tanatológico asociados a una Solicitud de Experticia Forense. 4. RF Ver listado de Informes de Experticia Forense asociados a una Solicitud de Experticia Forense. 5. RF Imprimir o exportar a PDF la información de una experticia.

Tabla 2 Descripción del CU Consultar Experticias Forenses Asignadas.

Nombre del CU	Consultar Experticias Forenses Asignadas
Actor (es)	Consultor de Experticias

Descripción	El caso de uso inicia cuando el actor accede a la opción de consultar las Solicitudes de Experticia Forense que le han sido asignadas. El sistema muestra un listado de Solicitudes de Experticia Forense que están asignadas al actor y que aún no han sido concluidas. El actor introduce uno o varios criterios de búsqueda y el sistema muestra una lista que solamente incluye los elementos que satisfagan esos criterios.
Referencia	1. Mostrar un listado de Solicitudes de Experticia Forense asignadas. 2. Ordenar las Solicitudes de Experticia Forense por diferentes criterios 3. Filtrar las Solicitudes de Experticia Forense a partir de sus datos. 4. Imprimir o Exportar a PDF. 5. Mantener informado al usuario del resultado de las operaciones

Tabla 3 Descripción del CU Consultar Registro Histórico de Experticias/Casos Forenses.

Nombre del CU	Consultar Registro Histórico de Experticias/Casos en Forenses
Actor (es)	Consultor de Registro Histórico de Experticias / Casos Forenses
Descripción	El caso de uso inicia cuando el actor accede a la opción de consultar la información relativa a Experticias realizadas en su Despacho. Introduce uno o varios criterios de búsqueda y el sistema muestra una lista con los Informes y/o Solicitudes de Experticia Forense procesadas en el Despacho y que satisfacen los criterios seleccionados. El actor puede imprimir o exportar a formato PDF dicho listado.
Referencia	1. Buscar Solicitudes de Experticia Forense dado criterios 2. Mostrar un listado de Solicitudes de Experticia Forense ordenado por un criterio 3. Ordenar un listado de Solicitudes de Experticia Forense dado un criterio

	4. Imprimir o Exportar a PDF un listado de Solicitudes de Experticia Forense. 5. Mantener informado al usuario del resultado de las operaciones.

Tabla 4 Descripción del CU Gestionar Protocolo de Exhumación.

Nombre del CU	Gestionar Protocolo de Exhumación
Actor (es)	Gestor de Informes
Descripción	El caso de uso inicia cuando el actor indica que desea realizar una acción sobre un Protocolo de Exhumación determinado. El actor puede incluir un nuevo informe, ver los detalles de un informe existente o modificar los mismos. En caso de que se seleccione la opción de incluir un Protocolo de Exhumación, el sistema dará la posibilidad de insertar los datos del mismo. Si el actor elige la opción de ver un Protocolo de Exhumación, el sistema muestra el contenido del mismo, así como los comentarios de revisión asociados a él y brinda la funcionalidad de generar una vista previa del informe, con posibilidad de imprimir o exportar a formato PDF. Si el actor elige la opción de modificar un Protocolo de Exhumación, el sistema muestra el contenido del mismo, permitiendo la edición de los datos y una vez realizados los cambios, guarda las modificaciones.
Referencia	1. Ver datos de Protocolo de Exhumación. 2. Mostrar las Imágenes asociadas a un Protocolo de Exhumación. 3. Ampliar una Imagen asociada a un Protocolo de Exhumación. 4. Mostrar los Comentarios de Revisión asociados a un Protocolo de Exhumación, ordenados de forma descendente según la fecha y hora de su creación. 5. Incluir Protocolo de Exhumación. 6. Modificar datos de Protocolo de Exhumación. 7. Asociar una Imagen a un Protocolo de Exhumación.

	8. Disociar una Imagen de un Protocolo de Exhumación. 9. Imprimir o exportar a PDF un Protocolo de Exhumación. 10. Validar la integridad de los datos introducidos por el usuario. 11. Mantener informado al usuario del resultado de las operaciones.

2.2 Diseño de la UI

2.2.1 Fases de la metodología a utilizar

La metodología a utilizar es DDWUI, de forma general se explicará cómo funciona al ser aplicada.

La primera tarea es crear el equipo de montadores de UI, el cual debe estar conformado por personas que les guste el diseño y que tengan aptitud para el mismo. Dicho equipo será el responsable de todo lo que tenga que ver con UI a partir de ese momento, especializándose así en este campo. Le sigue la etapa de capacitación a los integrantes del equipo, donde deben brindarse cursos afines a su trabajo en herramientas como: Photoshop, Corel Draw, HTML, CSS, JavaScript, Ajax, con el fin de perfeccionar sus habilidades. Se aconseja realizar varios casos de estudio antes de comenzar a trabajar en el proyecto real, para familiarizarse con el proceso.

La metodología que se utilizará consta de cinco fases:

- ✓ Estudio y Modelado del Sistema.
- ✓ Arquitectura de la Información.
- ✓ Diseño Gráfico.
- ✓ Elaboración de los Recursos de Montaje de Interfaces.
- ✓ Desarrollo de las UI.

En el Estudio y Modelado del Sistema es donde se obtiene una primera presentación de cómo será la UI, en esta fase como indica su nombre, es donde se modela el sistema, se hace el levantamiento de requisitos, o sea aquí se estudia el entorno donde está enmarcado el negocio desde el punto de vista de la UI. Se agrupan los requisitos no funcionales de usabilidad y UI, con el objetivo de obtener información subliminar acerca de la preferencia de los clientes.

La persona responsable de esta fase es el analista y se obtiene como resultado el documento con las especificaciones de los CU.

En la segunda fase, o sea, Arquitectura de la Información, se establece la organización de los contenidos dentro de las interfaces. A partir de un estudio realizado a la audiencia, se definen por parte de los diseñadores y arquitecto de información pautas a seguir, que son estudiadas por los diseñadores de UI.

El responsable de la fase es el arquitecto de información y como resultado se obtiene el documento explicativo de pautas de arquitectura de información y un prototipo de UI.

En Diseño Gráfico, basado en los principios de diseño web, se realiza un boceto de las interfaces. En esta fase la persona responsable es el diseñador asignado al proyecto, y como resultado se obtiene un documento explicativo de las pautas para el montaje de las interfaces, que incluye un archivo .psd y varios .jpg para el apoyo.

En la cuarta fase, se construyen elementos que serán utilizados en el desarrollo de las UI, a través del documento que contiene las pautas de montaje, se definen las clases de estilo que se irán a utilizar y se propone un prototipo funcional básico del sistema. En esta fase los responsables son los montadores UI, y como resultado se obtiene el escritorio de trabajo, que no es más que una página .jsp que contiene regiones que no cambiaran en ninguna interfaz.

Por último en la fase de Desarrollo de las UI, como lo indica su nombre, es donde se desarrollan las interfaces, apoyándose en las especificaciones de los CU y utilizando las hojas de estilos predefinidas, obteniéndose las páginas .jsp, las que son entregadas a los programadores para que prosigan su trabajo.

2.2.2 Resultados Obtenidos

A continuación se muestran los resultados obtenidos en el montaje de la interfaces de usuario, se realizarán siguiendo dicha metodología. Será aplicada a cada uno de los CU del Módulo Investigación en Ciencias Forenses, se expondrán los resultados obtenidos realizando una iteración completa de la metodología, tomando como ejemplo el CU Gestionar Protocolo de Exhumación que pertenece al submódulo de *experticias.*

En proyecto CICPC el equipo de montadores UI se conformó desde los inicios del mismo, los integrantes no fueron escogidos al azar sino que se analizó que tuvieran aptitudes y les gustara el

diseño. Inicialmente estaba compuesto por aproximadamente diez estudiantes, más tarde el proyecto quedó dividido en subsistemas y por cada equipo existían aproximadamente dos montadores UI.

A continuación los montadores recibieron cursos de las herramientas de trabajo propuestas en la metodología, los cuales fueron impartidos por profesores de diseño y programación web, con el fin de capacitarlos.

Para adquirir práctica se realizaron tres casos de estudio, estos fueron Sistema de Ficha Técnica del Estudiante, Sistema para controlar el Tiempo de Máquina de los Laboratorios y Sistema para la entrega de Materiales Docentes.

En la primera fase de la metodología, Estudio y Modelado del Sistema, los analistas del proyecto realizaron el modelamiento del negocio y el levantamiento de los requisitos, a partir del análisis del entorno donde se encuentra enmarcado el negocio. Como resultado se obtuvieron las especificaciones de CU (**Anexo #1**).

En la segunda fase, Arquitectura de la Información, los arquitectos de la información realizaron el estudio de la audiencia y las necesidades que tenían. A partir de dicho estudio quedaron definidos los contenidos que se incluirían en el producto y la organización de los mismos. Considerando los factores de usabilidad y accesibilidad, quedó organizado el sistema de navegación, el de etiquetado y el diseño conceptual. Se elaboró también el documento explicativo de Pautas de Arquitectura de Información y se construyeron los prototipos de UI para cada CU**.** Seguidamente se muestra cómo quedó conformado el prototipo de interfaz gráfica del CU Gestionar Protocolo de Exhumación.

Fig. 12 Ejemplo del primer prototipo de UI. CU Gestionar Protocolo de Exhumación.

En la fase de Diseño Gráfico, después de estudiado el documento explicativo de Pautas de Arquitectura de Información y analizado el comportamiento de la aplicación, quedaron conformadas las interfaces del sistema por parte del diseñador gráfico, aplicándose así los cambios necesarios. Quedó elaborado el documento general de pautas para los montadores UI Manual de Pautas SIIPOL. Se obtuvo también el archivo.psd con el diseño de las interfaces y finalmente las muestras de las pantallas en formato .jpg, en las que se apoyaron los montadores**.**

Fig. 13 Muestra de la pantalla del CU Protocolo de Exhumación en formato .jpg.

En la fase de Elaboración de recursos de montaje, los montadores UI construyeron el escritorio de trabajo para la aplicación y quedó conformada la hoja de estilos, teniendo en cuenta los artefactos antes obtenidos.

En la quinta y última fase, Desarrollo de UI, los montadores se encargaron de construir las interfaces del CU Protocolo de Exhumación, haciendo uso del prototipo de UI, la especificación del CU y de la hoja de estilos.

2.3 Análisis

2.3.1 Modelo de Análisis

El Modelo de Análisis contiene clases de análisis y sus objetos organizados en paquetes que colaboran. En la construcción del modelo de análisis se tienen que identificar las clases que describen la realización de los CU, los atributos y las relaciones entre ellas. Con dicha información se

construye el diagrama de clases del análisis, que por lo general se descompone para agrupar las clases en paquetes. Esta descomposición tiene impacto por lo general en el diseño e implementación de la solución (27).

2.3.1.1 Clases del análisis

Se centran en los requisitos funcionales y son evidentes en el dominio del problema porque representan conceptos y relaciones del dominio. Tienen atributos y entre ellas se establecen relaciones de asociación, agregación/composición, generalización/especialización y tipos asociativos. RUP propone clasificar a las clases en:

Las clases Interfaz son aquellas que permiten al actor interactuar con el sistema, pueden ser una pantalla, reporte, formulario etc.

Las clases de Control se utilizan para representar coordinación, secuencia, transacciones y control de otros objetos y se usan con frecuencia para encapsular el control de un CU en concreto. También se utilizan para representar cálculos complejos, como la lógica del negocio, se pueden identificar en los CU como verbos.

Las clases Entidad representan una información o un dato persistente; suelen mostrar una estructura de datos lógica y contribuyen a comprender de qué información depende el sistema (27).

2.3.2 Diagramas de clases del análisis

Para CU del sistema se realizó un diagrama de clases del análisis. A continuación se presenta una muestra de los mismos, los restantes diagramas se encuentran en el **Anexo # 2**.

Fig. 14 Diagrama de clases del análisis CU Ver Detalles de Experticia.

Fig. 15 Diagrama de clases del análisis CU Consultar Detalles de Experticias Asignadas.

Fig. 16 Diagrama de clases del análisis CU Consultar Registro Histórico de Experticias/Casos Forenses.

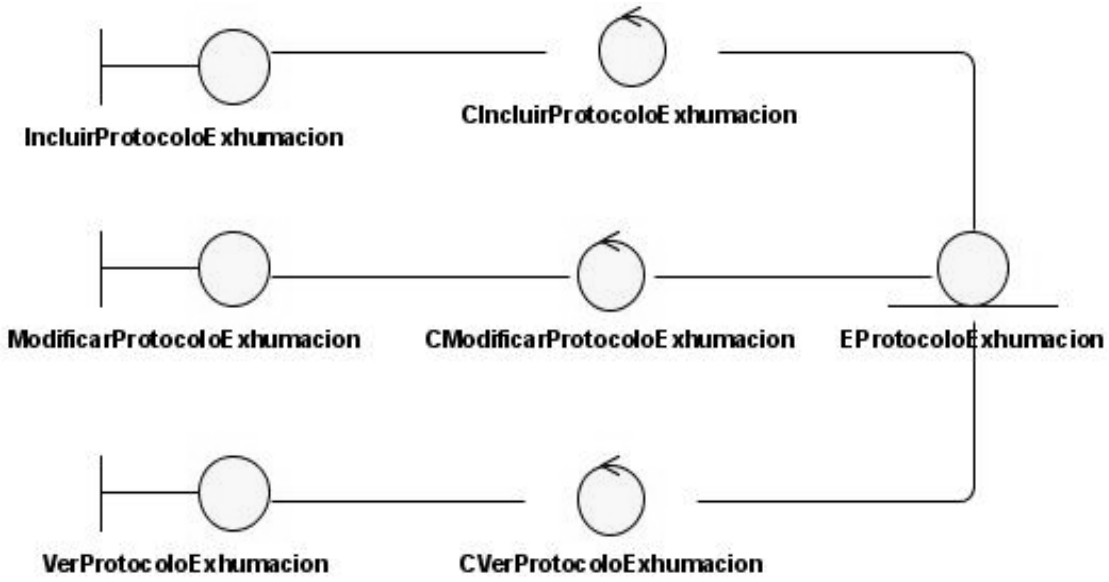

Fig. 17 Diagrama de clases del análisis CU Gestionar Protocolo de Exhumación.

2.4 Diseño

2.4.1 Modelo de Diseño

El modelo de diseño es un modelo de objetos que describe la realización de los CU, y sirve como una abstracción del modelo de implementación y el código fuente. Es usado como una entrada inicial en las actividades de implementación y prueba (28).

2.4.2 Estructura lógica de paquetes

A partir de la herramienta de modelado escogida para el desarrollo del proyecto, Visual Paradigm, se generara el código correspondiente para el IDE Eclipse, para lograrlo esta herramienta se basa en los componentes presentes en el Repositorio de Clases. Esta vista contendrá exactamente la estructura de carpetas del proyecto, definida por el arquitecto principal en el documento Guía de estilo de código para el proyecto CICPC.

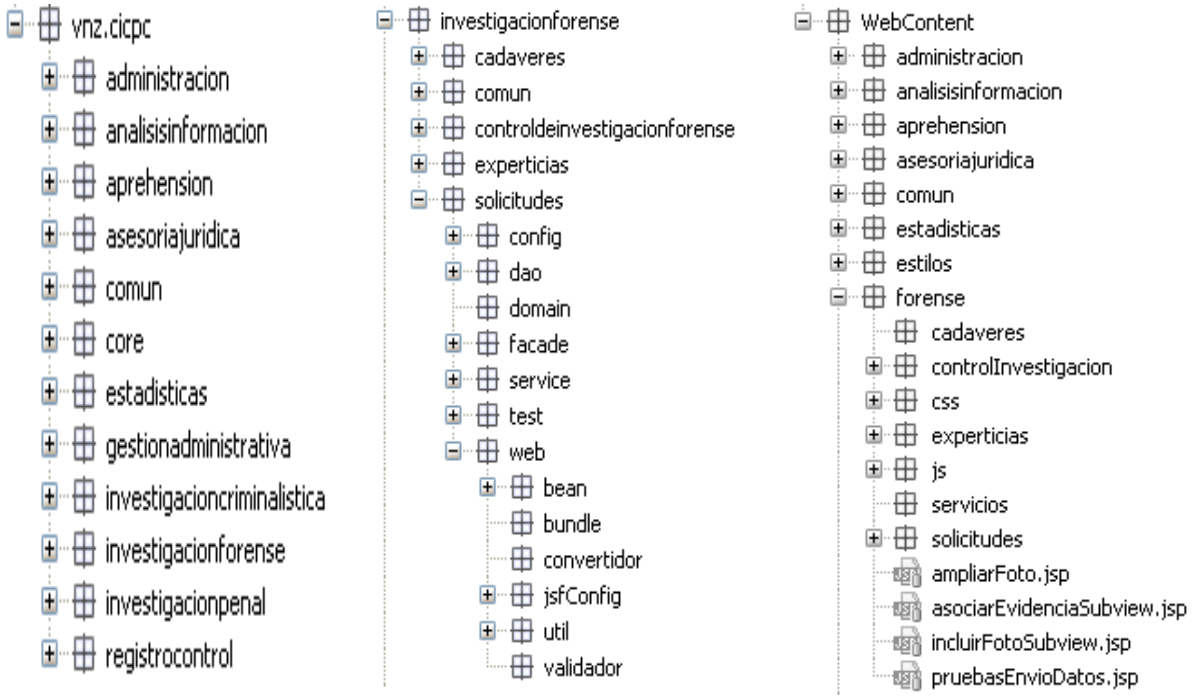

Fig. 18 Estructura de Carpetas.

2.4.3 Diagrama de Contrato entre Paquetes

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

Para cada CU del sistema se realizó un Diagrama de Contrato Entre Paquetes. Con el objetivo de lograr una mayor claridad y comprensión de los mismos, a continuación se explica el significado de los siguientes mensajes:

cargarDatosInicializacion: en este mensaje se engloban todos los métodos encargados de interactuar con la fachada con el objetivo de cargar los datos necesario para la inicialización de la página, como por ejemplo, los métodos necesarios para obtener los nomencladores.

Convertir y Mostrar Datos: en este mensaje abarca todo el proceso que se hace para convertir (formatear) los datos, necesario para la inicialización de la página.

Validar y Convertir Datos: este mensaje comprende todas las validaciones que se hacen para validar los datos (del lado del cliente y del lado del servidor) y el proceso que hace JSF para convertir (formatear) los datos.

A continuación se presenta una muestra de los diagramas de contrato entre paquetes, los restantes diagramas se encuentran en el **Anexo # 3**.

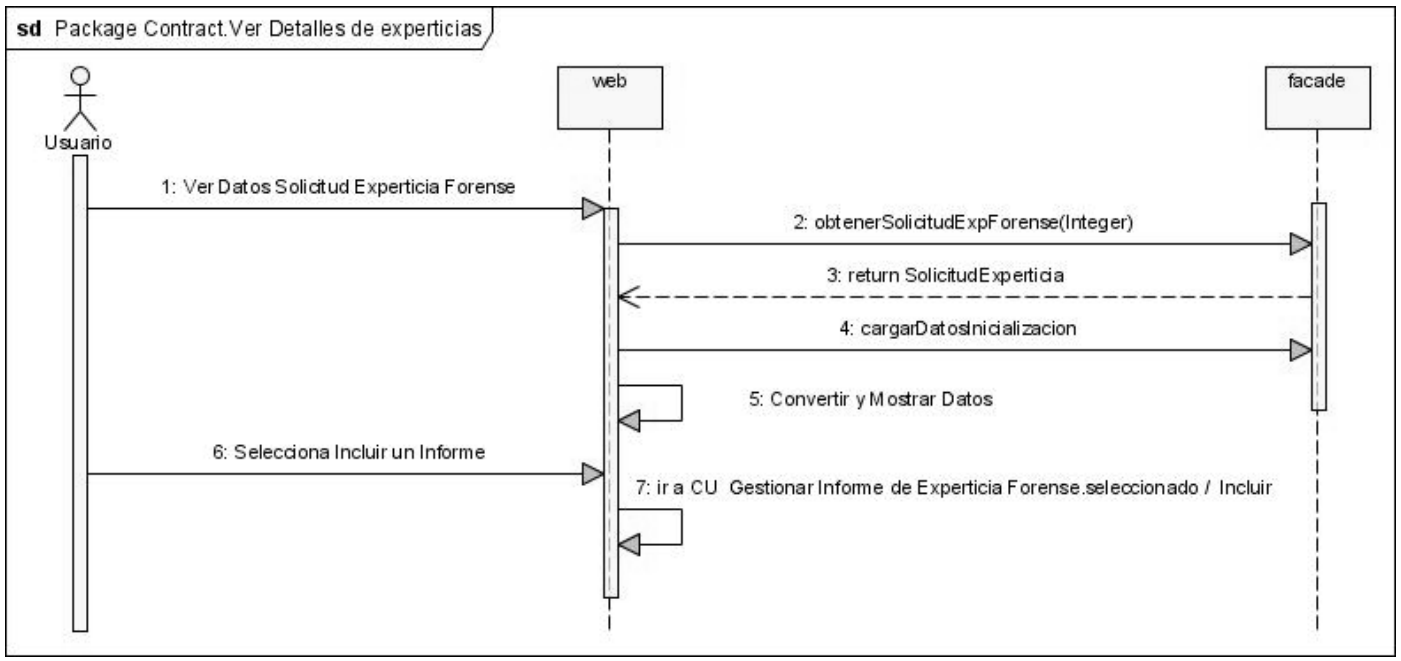

Fig. 19 Diagrama de contrato entre paquetes CU Ver Detalles de Experticia.

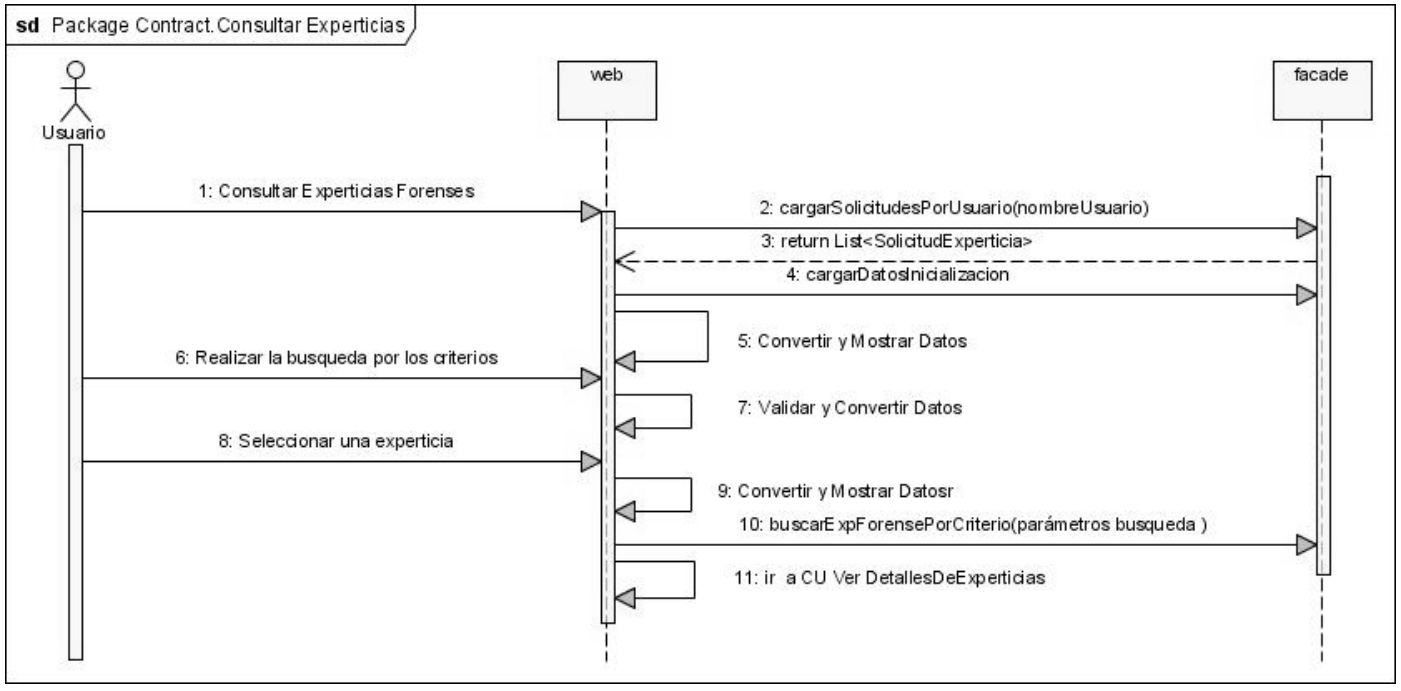

Fig. 20 Diagrama de contrato entre paquetes CU Consultar Experticias Asignadas.

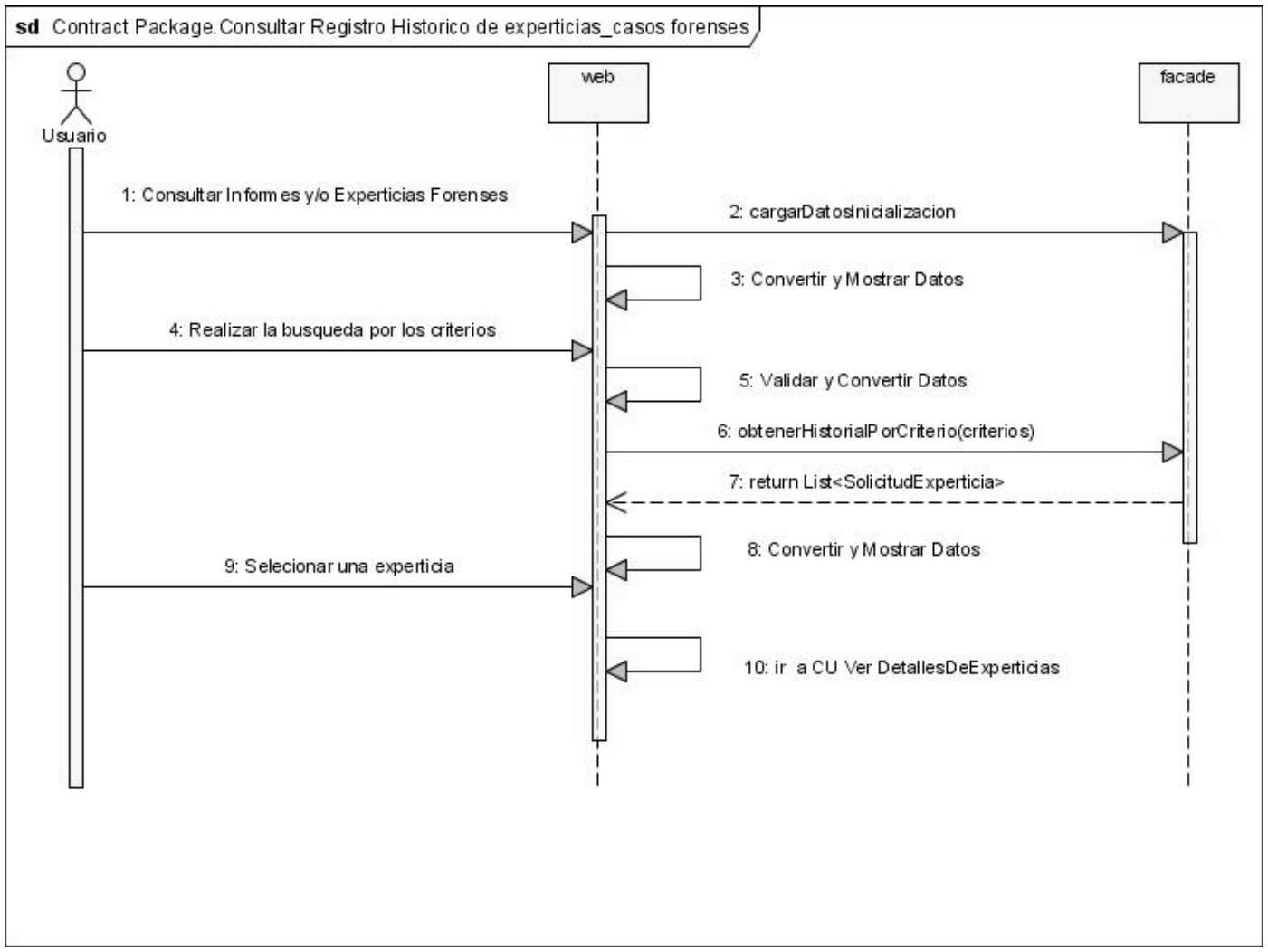

Fig. 21 Diagrama de contrato entre paquetes CU Consultar Registro Histórico de Experticias/Casos Forenses.

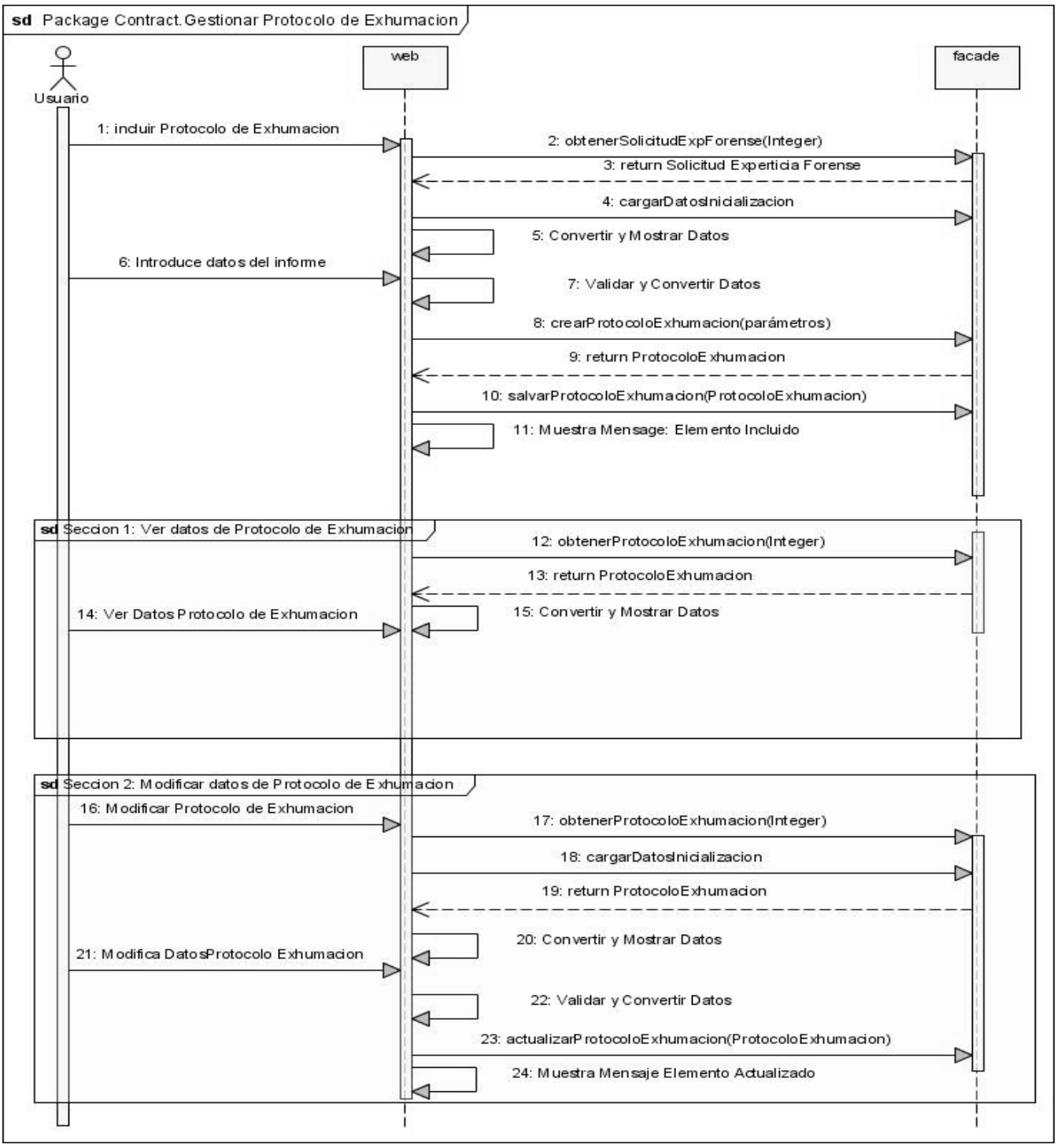

Fig. 22 Diagrama de contrato entre paquetes CU Gestionar Protocolo de Exhumación.

2.4.4 Diagrama de clases de diseño

Los diagramas de clases son los más utilizados en el modelado de sistemas orientados a objetos. Un diagrama de clases muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Los diagramas de clases se utilizan para modelar la vista de diseño estática de un sistema, esto incluye modelar el vocabulario del sistema, modelar las colaboraciones o modelar esquemas.

Los diagramas de clases son importantes no sólo para visualizar, especificar y documentar modelos estructurales, sino también para construir sistemas ejecutables, aplicando ingeniería directa e inversa (28).

Para cada CU del sistema se realizó un diagrama de clases de diseño. Con el objetivo de lograr una mayor claridad y comprensión, se tomaron los siguientes acuerdos:

En la carpeta *facade* se han colocado todas las fachadas con las cuales interactúa el CU correspondiente, esto no quiere decir que todas las fachadas se encuentran en esa misma carpeta, sino que se encuentran en la carpeta *facade* de su módulo o submódulo correspondiente, ejemplo, la fachada ExperticiaForenseFacade se halla dentro de este directorio de carpetas *vnz.cicpc .investigacionforense. experticias.facade.*

La carpeta convertidor se encuentra dentro del directorio de carpetas vnz.cicpc. investigacionforense.comun.web.

La carpeta *bean* perteneciente a los CUs del submódulo *experticias* se encuentra dentro del directorio de carpetas *vnz.cicpc.investigacionforense. experticias.web.*

La carpeta *bean* perteneciente al CU Gestionar Solicitud de Experticia Forense se encuentra dentro del directorio de carpetas vnz.cicpc.investigacionforense.solicitudes.

La carpeta *bean* perteneciente al CU Consultar Registro Histórico de Experticias/Casos en Forenses se encuentra dentro del directorio de carpetas *vnz.cicpc.investigacionforense. controldeinvestigacionforense.web.*

Las carpetas experticias, *controlInvestigacion, solicitudes y js* se encuentran dentro del directorio de carpetas *WebContent.forense.*

La clase FacesServlet pertenece al framework JSF.

A continuación se presenta una muestra de los diagramas de clase de diseño, los restantes diagramas se encuentran en el **Anexo # 4.**

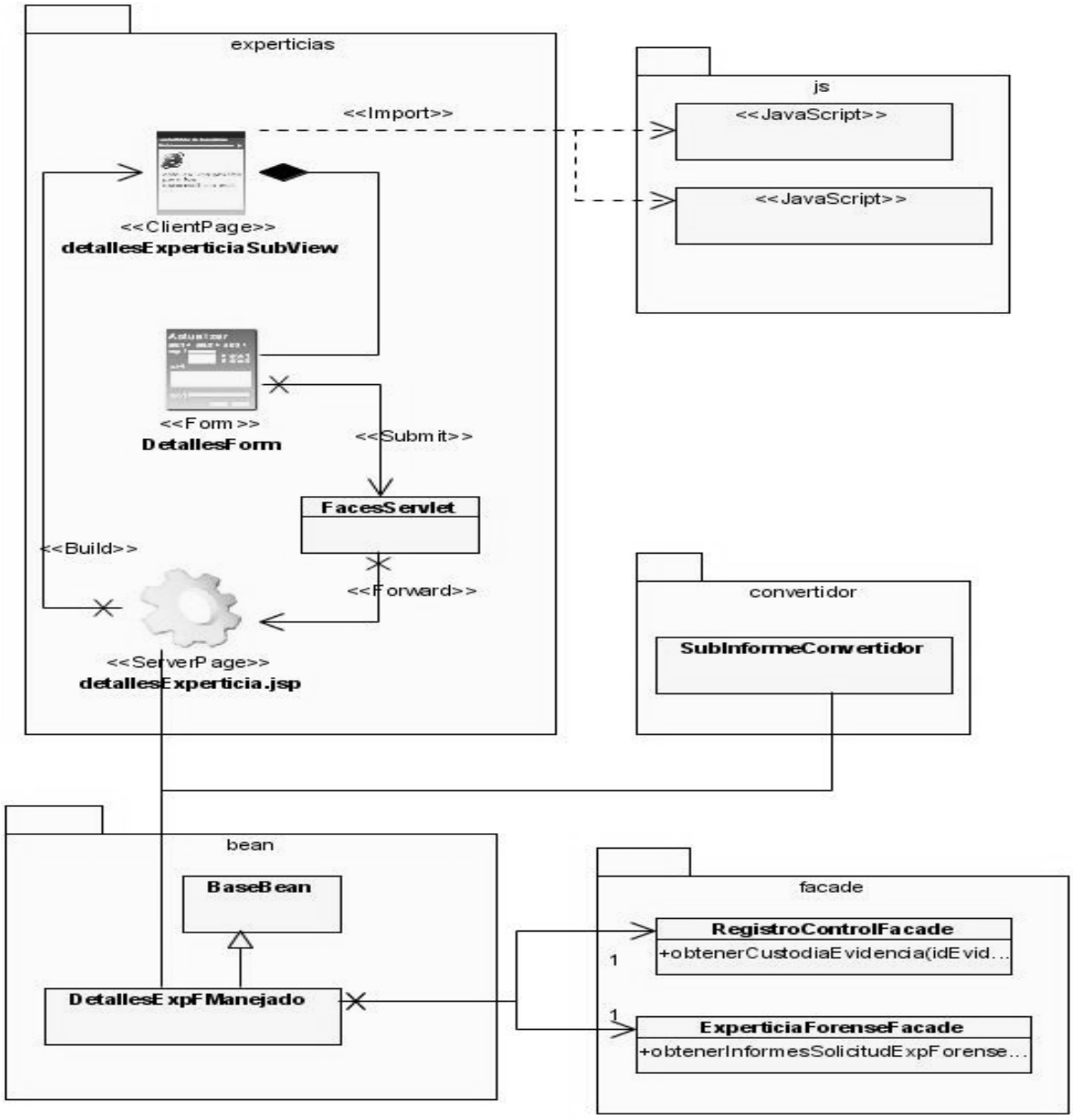

Fig. 23 Diagrama de clases de diseño CU Ver Detalles de Experticia.

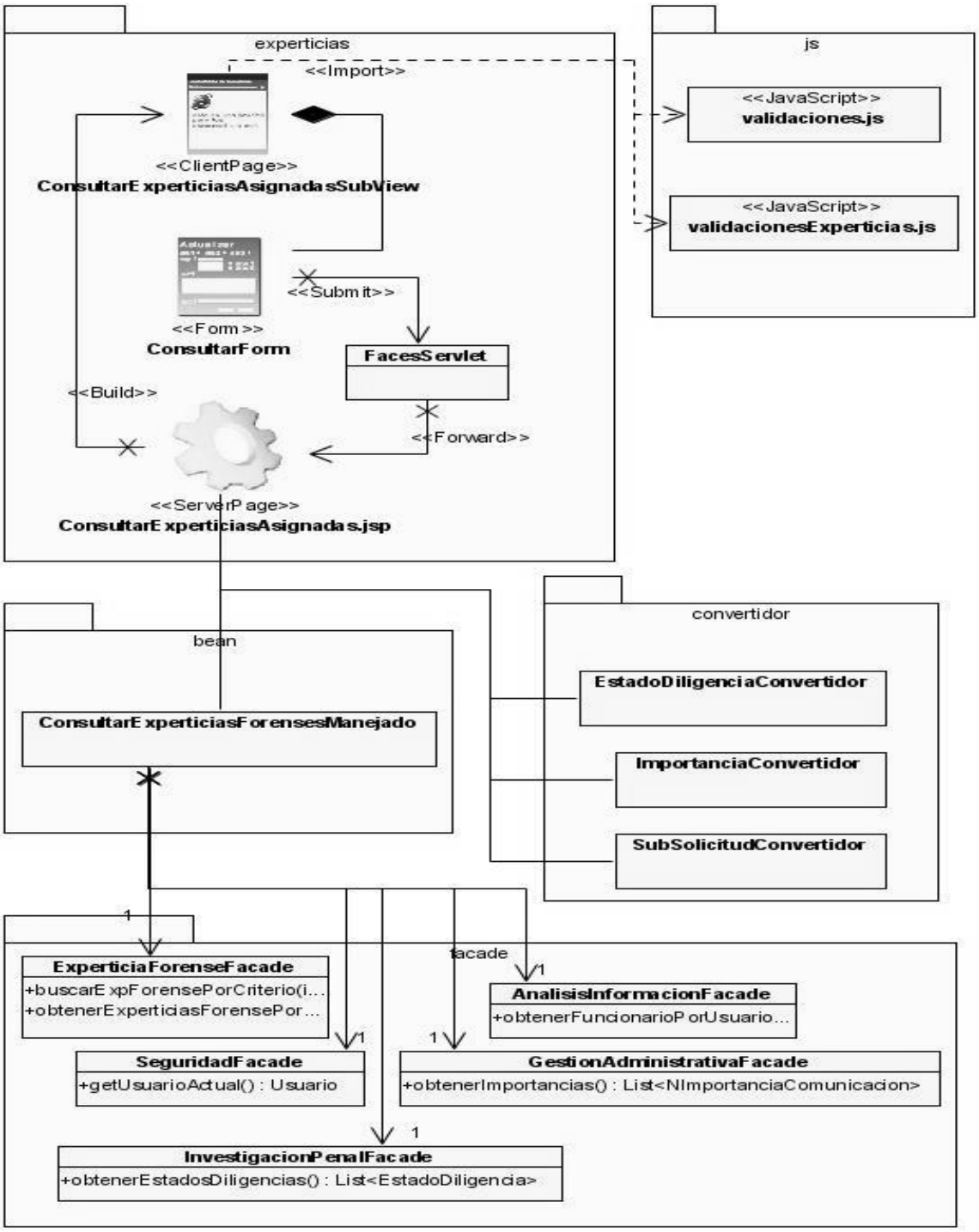

Fig. 24 Diagrama de clases de diseño CU Consultar Experticias Asignadas.

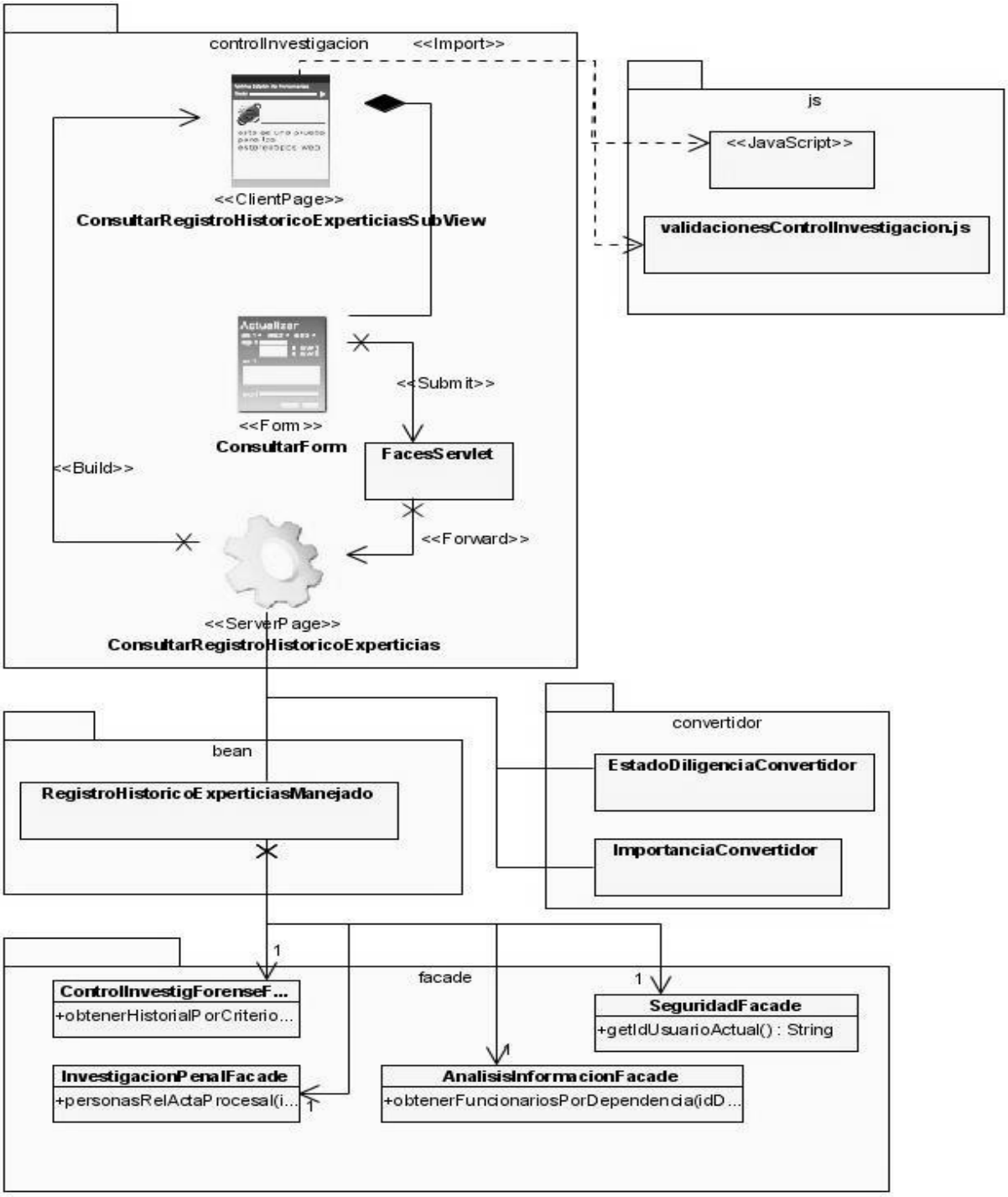

Fig. 25 Diagrama de clases de diseño CU Consultar Registro Histórico de Experticias/Casos Forenses.

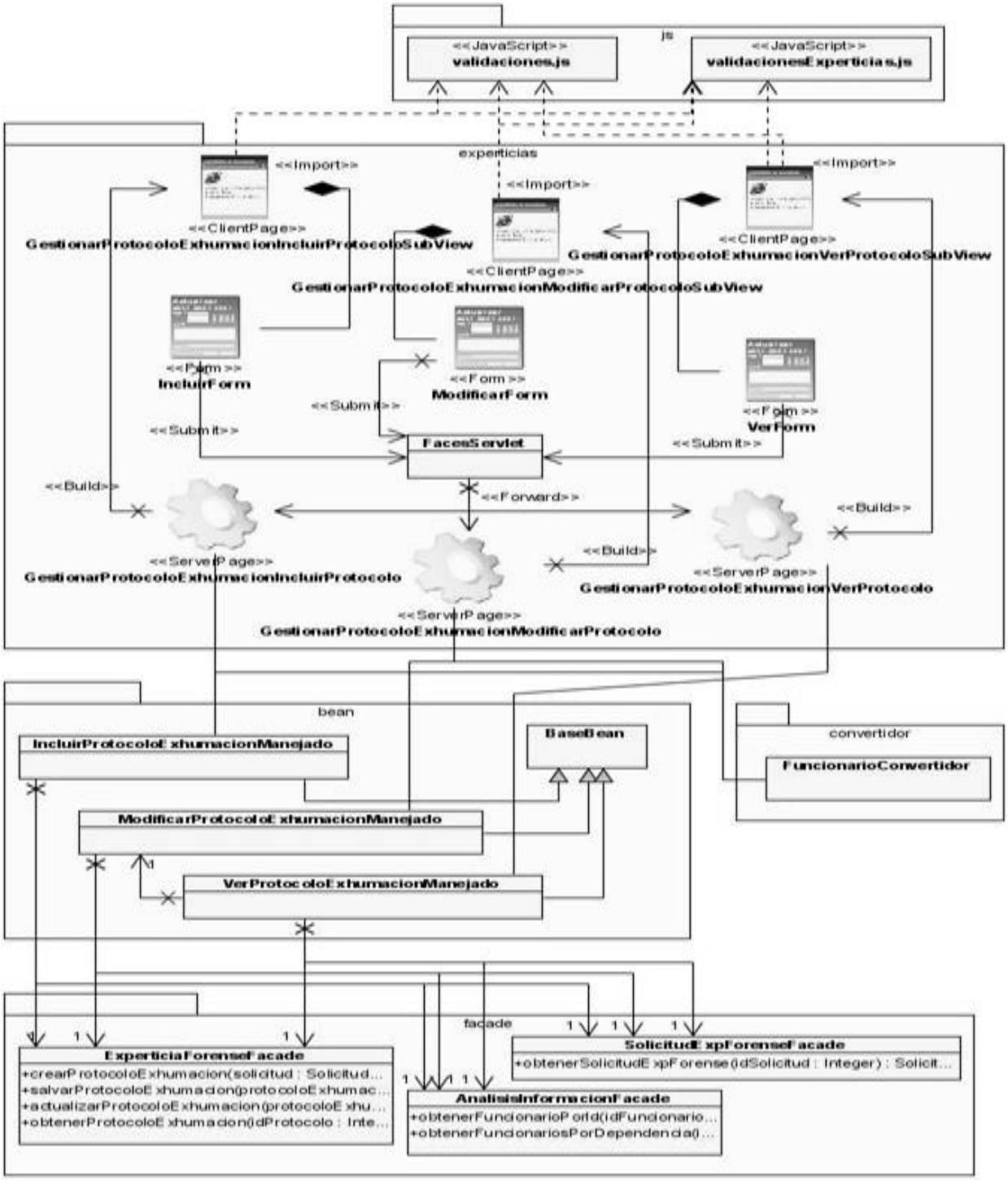

Fig. 26 Diagrama de clases de diseño CU Gestionar Protocolo de Exhumación.

2.4.5 Descripción de las Clases

A continuación se representa la descripción de clases de los CU analizados en el documento, en el **Anexo # 5** se pueden encontrar algunas descripciones más.

Tabla 5 Descripción de clases CU Ver Detalles Experticia.

Nombre: DetallesExpFManejado	
Tipo de clase: Controladora	
Atributo	**Tipo**
investigacionPenalFacade	InvestigacionPenalFacade
experticiaForenseFacade	ExperticiaForenseFacade
idSolicitud	Integer
evidencia	Evidencia
datosPersona	DatosPersona
solicitud	SolicitudExperticia
funcionarioSolicitante	Funcionario
fechaAsignacion	Date
fechaConclusion	Date
fechaRegsitro	Date
direccionEvidencia	String
funcSolicitante	String
ocupacionesPersona	String
subInforme	SubInforme

listTiposInforme	List<SubInforme>
itemstiposInformes	List<SelectItem>
listaDatosInforme	List<DatosInforme>
funcionarios	List<FuncionarioDiligencia>
listaInformes	List<InformeExperticiaForense>
tablaInformesAsociados	UIData
arCreador	AccionRevision
arRevisor	AccionRevision
arSupervisor	AccionRevision
arCorrector	
Para cada responsabilidad:	
Nombre:	verInforme(ActionEvent e)
Descripción:	Esta función se usa para ir a la vista de mostrar los datos del informe seleccionado
Nombre:	adicionarInformeAsociado(Integer idInforme)
Descripción:	Esta función se usa para adicionar un nuevo informe a la lista de informes
Nombre:	actualizarEliminarInformeAsociado(Integer idInforme, boolean eliminar)
Descripción:	Esta función se usa para actualizar un informe asociado a la solicitud, eliminar es true, se elimina el informe
Nombre:	crearNuevoInforme(ActionEvent e)

Descripción:	Esta función se usa para navegar a la vista Incluir del informe seleccionado
Nombre:	verDetallesDeExperticias(String urlAnterior, Integer idSolicitud)
Descripción:	Esta función se usa cuando se va a llamar CU ver Detalles de Experticias
Nombre:	setIdSolicitud(Integer idSolicitud)
Descripción:	Esta función se usa para cargar los datos de la solicitud y llamar a los métodos que cargan los datos para la inicialización
Nombre:	cargarTiposInforme()
Descripción:	Esta función se usa para cargar los nomencladores tipo de informe
Nombre:	cargarTelefonosPersona(Persona p)
Descripción:	Esta función se usa para cargar los teléfonos asociados a una persona
Nombre:	cargarDireccionHabitaPersona(Persona p)
Descripción:	Esta función se usa para cargar las direcciones de tipo habita asociadas a una persona
Nombre:	cargarDireccionEvidencia(Direccion d)
Descripción:	Esta función se usa para formatear la dirección asociada a la evidencia
Nombre:	cargarOcupacionesPersona(Persona persona)
Descripción:	Esta función se usa para cargar las ocupaciones asociadas a una persona
Nombre:	cargarDatosInformes()

Descripción:	Esta función se usa para cargar los informes asociados a la solicitud
Nombre:	crearDatosInforme(InformeExperticiaForense informe)
Descripción:	Esta función se usa para crear, formatear, los Datos de un informe
Nombre:	obtenerInforme(Integer id)
Descripción:	Esta función se usa para obtener un informe de la lista de informes
Nombre:	cancelar(ActionEvent e)
Descripción:	método para realizar la acción cancelar

Tabla 6 Descripción de clases CU Consultar Experticias.

Nombre: ConsultarExperticiasForensesManejado	
Tipo de clase: Controladora	
Atributo	**Tipo**
experticiaForenseFacade	ExperticiaForenseFacade
numeroSolicitud	String
actaProcesal	String
dependencia	String
nombreUsuario	String
credencial	String
fechaInicio	Date
fechaFin	Date

estadosSolicitud	SelectItem[]
tiposExperticia	SelectItem[]
importancias	SelectItem[]
tipoSeleccionado	SubSolicitud
estadoSeleccionado	EstadoDiligencia
importancia	NImportanciaComunicacion
tablaSolicitudesAsociadas	UIData
solicitudesCorrespodientes	List<SolicitudExperticiaForense>
idSolicitudSeleccionada	Integer
busquedaActiva	boolean
permitirBuscarInicio	boolean
DetallesExpFManejado	detallesExpFManejado
Para cada responsabilidad:	
Nombre:	seleccionarAsociada(ActionEvent e)
Descripción:	Esta funcion se usa para selecionar una experticia
Nombre:	buscarPorCriterios(ActionEvent e)
Descripción:	Esta funcion se usa para realizar la busqueda por los criterios
Nombre:	nuevaBusqueda(ActionEvent e)
Descripción:	Esta función se usa para inicializar los parámetros de búsqueda

Nombre:	abrirCerrarBusqueda(ActionEvent e)
Descripción:	Esta función se usa para abrir o cerrar una búsqueda
Nombre:	cancelar(ActionEvent e)
Descripción:	método para realizar la acción cancelar
Nombre:	inicializarCriterios()
Descripción:	Esta función se usa para inicializar los criterios de búsqueda
Nombre:	cargarTiposSolicitud()
Descripción:	Esta función se usa para cargar el nomenclador tipo de solicitud
Nombre:	cargarImportancias()
Descripción:	Esta función se usa para cargar el nomenclador importancia
Nombre:	cargarEstados()
Descripción:	Esta función se usa para cargar el nomenclador estados de la solicitud
Nombre:	cargarDependenciasInternas()
Descripción:	Esta función se usa para cargar las dependencias internas del CICPC
Nombre:	cargarIdDespachoUsuario()
Descripción:	Esta función devuelve el identificador del despacho a que pertenece el usuario
Nombre:	cargarUsuarioSesion()
Descripción:	Esta función se usa para obtener el usuario de la sesión
Nombre:	cargarSolicitudesPorUsuario(String usuario)

Descripción:	Esta función se usa para obtener las solicitudes asociadas al usuario

Tabla 7 Descripción de clases CU Consultar Registro Histórico de Experticias/Casos Forenses.

Nombre: RegistroHistoricoExperticiasManejado	
Tipo de clase: Controladora	
Atributo	**Tipo**
controlInvestigForenseFacade	ControlInvestigForenseFacade
elementosEncontrados	List<SolicitudExperticiaForense>
estadosSolicitud	SelectItem[]
tiposExperticia	SelectItem[]
despachos	SelectItem[]
importancias	SelectItem[]
tipoSeleccionado	SubSolicitud
estadoSeleccionado	EstadoDiligencia
importancia	NImportanciaComunicacion
despacho	String
identificador	String
noActa	String
noExpediente	String

credencialSolicitante	String
credencialResponsable	String
fechaInicio	Date
fechaFin	Date
idSolicitudSeleccionada	Integer
idDespachoSesion	Integer
tablaElementos	UIData
detallesExpFManejado	DetallesExpFManejado
Para cada responsabilidad:	
Nombre:	buscar(ActionEvent e)
Descripción:	Esta función se usa para realizar la búsqueda por los criterios
Nombre:	seleccionar(ActionEvent e)
Descripción:	Esta función se usa para mostrar una solicitud o un expediente seleccionado
Nombre:	nuevaBusqueda(ActionEvent e)
Descripción:	Esta función se usa para reiniciar los parámetros para hacer una nueva búsqueda
Nombre:	cancelar(ActionEvent e)
Descripción:	método para realizar la acción cancelar
Nombre:	inicializarCriterios()

Descripción:	Esta función se usa para inicializar los criterios de búsqueda
Nombre:	cargarTiposSolicitud()
Descripción:	Esta función se usa para cargar el nomenclador tipo de solicitud
Nombre:	cargarImportancias()
Descripción:	Esta función se usa para cargar el nomenclador importancia
Nombre:	cargarEstados()
Descripción:	Esta función se usa para cargar el nomenclador estados de la solicitud
Nombre:	cargarDependenciasInternas()
Descripción:	Esta función se usa para cargar las dependencias internas del CICPC
Nombre:	cargarIdDespachoUsuario()
Descripción:	Esta función devuelve el identificador del despacho a que pertenece el usuario
Nombre:	cargarUsuarioSesion()
Descripción:	Esta función se usa para obtener el usuario de la sesión

Tabla 8 Descripción de clases CU Gestionar Protocolo de Exhumación.

Nombre: IncluirProtocoloExhumacionManejado	
Tipo de clase: Controladora	
Atributo	**Tipo**

expForenseFacade	SolicitudExpForenseFacade
experticiaForenseFacade	ExperticiaForenseFacade
idSolIncluirProtocolo	Integer
protocoloExhumacion	ProtocoloExhumacion
solicitudExperticia	SolicitudExperticia
comunicacion	Comunicacion
expertoCargo	Funcionario
credencialExp	String
fechaEstudio	Date
buttonGuardarIncluir	boolean
guardarIncluir	boolean
experto	Funcionario
credencial	String
listaExpReacionados	List<Funcionario>
tablaExpReacionados	UIData
itemsExpertos	SelectItem[]
conclusionExp	String
imagenesAsociadas	List<Foto>
tablaImagen	UIData

Foto	fotoSeleccionada
posActualFoto	int
imagenSubfuncionManejado	ImagenSubfuncionManejado
Para cada responsabilidad:	
Nombre:	setIdSolIncluirProtocolo(Integer idSolIncluirProtocolo)
Descripción:	set del id solicitud de Experticia Forense se inicializan los datos predeterminados
Nombre:	limpiar()
Descripción:	reiniciar los atributos
Nombre:	SelectItem[] cargarExpertos()
Descripción:	crear la selección de expertos del departamento
Nombre:	ScargarSolicitudExperticia(Integer idSolicitud)
Descripción:	obtener la solicitud
Nombre:	disociar(ActionEvent e)
Descripción:	para realizar la acción disociar un experto relacionado
Nombre:	asociar(ActionEvent e)
Descripción:	asociar un experto relacionado
Nombre:	incluir(ActionEvent e)
Descripción:	método para realizar la acción crear el Protocolo Exhumación cuando se va a guardar Concluir

Nombre:	guardar(ActionEvent e)
Descripción:	método para realizar la acción crear el Protocolo Exhumación cuando se va Guardar
Nombre:	validar()
Descripción:	método para validar en el lado del servidor
Nombre:	cancelar(ActionEvent e)
Descripción:	método para realizar la acción cancelar
Nombre:	verImagen(ActionEvent e)
Descripción:	Esta función se usa para ver una imagen de la lista de imágenes asociadas
Nombre:	asociarImagen(ActionEvent e)
Descripción:	Esta función se usa para asociar una imagen de la lista de imágenes asociadas
Nombre:	disociarImagen(ActionEvent e)
Descripción:	Esta función se usa para eliminar una imagen a la lista de imágenes

Nombre: ModificarProtocoloExhumacionManejado	
Tipo de clase: Controladora	
Atributo	**Tipo**
expForenseFacade	SolicitudExpForenseFacade

experticiaForenseFacade	ExperticiaForenseFacade
idProtocoloExhumacion	Integer
protocoloExhumacion	ProtocoloExhumacion
solicitudExperticia	SolicitudExperticia
comunicacion	Comunicacion
expertoCargo	Funcionario
credencialExp	String
fechaEstudio	Date
buttonGuardarIncluir	boolean
guardarIncluir	boolean
experto	Funcionario
credencial	String
listaExpReacionados	List<Funcionario>
tablaExpReacionados	UIData
itemsExpertos	SelectItem[]
conclusionExp	String
imagenesAsociadas	List<Foto>
tablaImagen	UIData
Foto	fotoSeleccionada

posActualFoto	int
imagenSubfuncionManejado	ImagenSubfuncionManejado
Para cada responsabilidad:	
Nombre:	setIdSolIncluirProtocolo(Integer idSolIncluirProtocolo)
Descripción:	set del id solicitud de Experticia Forense se inicializan los datos predeterminados
Nombre:	limpiar()
Descripción:	reiniciar los atributos
Nombre:	SelectItem[] cargarExpertos()
Descripción:	crear la selección de expertos del departamento
Nombre:	SolicitudExperticia cargarSolicitudExperticia(Integer idSolicitud)
Descripción:	obtener la solicitud
Nombre:	cargarExpertoACargo(ProtocoloExhumacion protocoloExhumacion)
Descripción:	cargar los experto a cargo al Protocolo Exhumacion
Descripción:	para realizar la acción disociar un experto relacionado
Nombre:	asociar(ActionEvent e)
Descripción:	asociar un experto relacionado
Nombre:	incluir(ActionEvent e)
Descripción:	método para realizar la acción crear el Protocolo Exhumación cuando se va guardar Concluir

Nombre:	guardar(ActionEvent e)
Descripción:	método para realizar la acción crear el Protocolo Exhumación cuando se va Guardar
Nombre:	validar()
Descripción:	método para validar en el lado del servidor
Nombre:	cancelar(ActionEvent e)
Descripción:	método para realizar la acción cancelar
Nombre:	verImagen(ActionEvent e)
Descripción:	Esta función se usa para ver una imagen de la lista de imágenes asociadas
Nombre:	asociarImagen(ActionEvent e)
Descripción:	Esta función se usa para asociar una imagen de la lista de imágenes asociadas
Nombre:	disociarImagen(ActionEvent e)
Descripción:	Esta función se usa para eliminar una imagen a la lista de imágenes

Nombre: VerProtocoloExhumacionManejado	
Tipo de clase: Controladora	
Atributo	**Tipo**
idProtocoloExhumacion	Integer

protocoloExhumacion	ProtocoloExhumacion
solicitudExperticia	SolicitudExperticia
comunicacion	Comunicacion
listaExpReacionados	List<Funcionario>
encargado	Funcionario
listaFuncionarioDiligencia	List<FuncionarioDiligencia>
imagenesAsociadas	List<Foto>
tablaImagen	UIData
Foto	fotoSeleccionada
posActualFoto	int
imagenSubfuncionManejado	ImagenSubfuncionManejado

Para cada responsabilidad:	
Nombre:	setIdSolIncluirProtocolo(Integer idSolIncluirProtocolo)
Descripción:	set del id solicitud de Experticia Forense se inicializan los datos predeterminados
Nombre:	limpiar()
Descripción:	reiniciar los atributos
Nombre:	cargarExpertoACargo(ProtocoloExhumacion protocoloExhumacion)
Descripción:	cargar los experto acargo al Protocolo Exhumación

Nombre:	cargarExpAsociados()
Descripción:	cargar los expertos asociados al Protocolo Exhumación
Nombre:	cargarSolicitudExperticia(Integer idSolicitud)
Descripción:	obtener la solicitud
Nombre:	cancelar(ActionEvent e)
Descripción:	método para realizar la acción cancelar
Nombre:	verImagen(ActionEvent e)
Descripción:	Esta función se usa para ver una imagen de la lista de imágenes asociadas

2.5 Implementación

2.5.1 Modelo de Implementación

El Modelo de Implementación describe como las clases, se implementan en términos de componentes, como ficheros de código fuente, ejecutables, etc. El Modelo de Implementación describe también como se organizan los componentes de acuerdo con los mecanismos de estructuración y modelación disponible en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y como dependen los componentes uno de otros.

2.5.2 Subsistema de Implementación

Una colección de componentes y otros subsistemas de implementación usados para estructurar el modelo de implementación y dividirlos en pequeñas partes que pueden ser integradas y probadas de forma separada. Los subsistemas de implementación incluyen dependencias y otras informaciones. También podrían incluir modelos claves del subsistema (diagramas de componentes, modelo de

despliegue). Además un subsistema puede implementar las interfaces que representan la funcionalidad que exportan en forma de operaciones (29).

El diagrama que se presenta a continuación muestra la relación entre los Subsistema de Implementación en los que queda estructurado la capa UI.

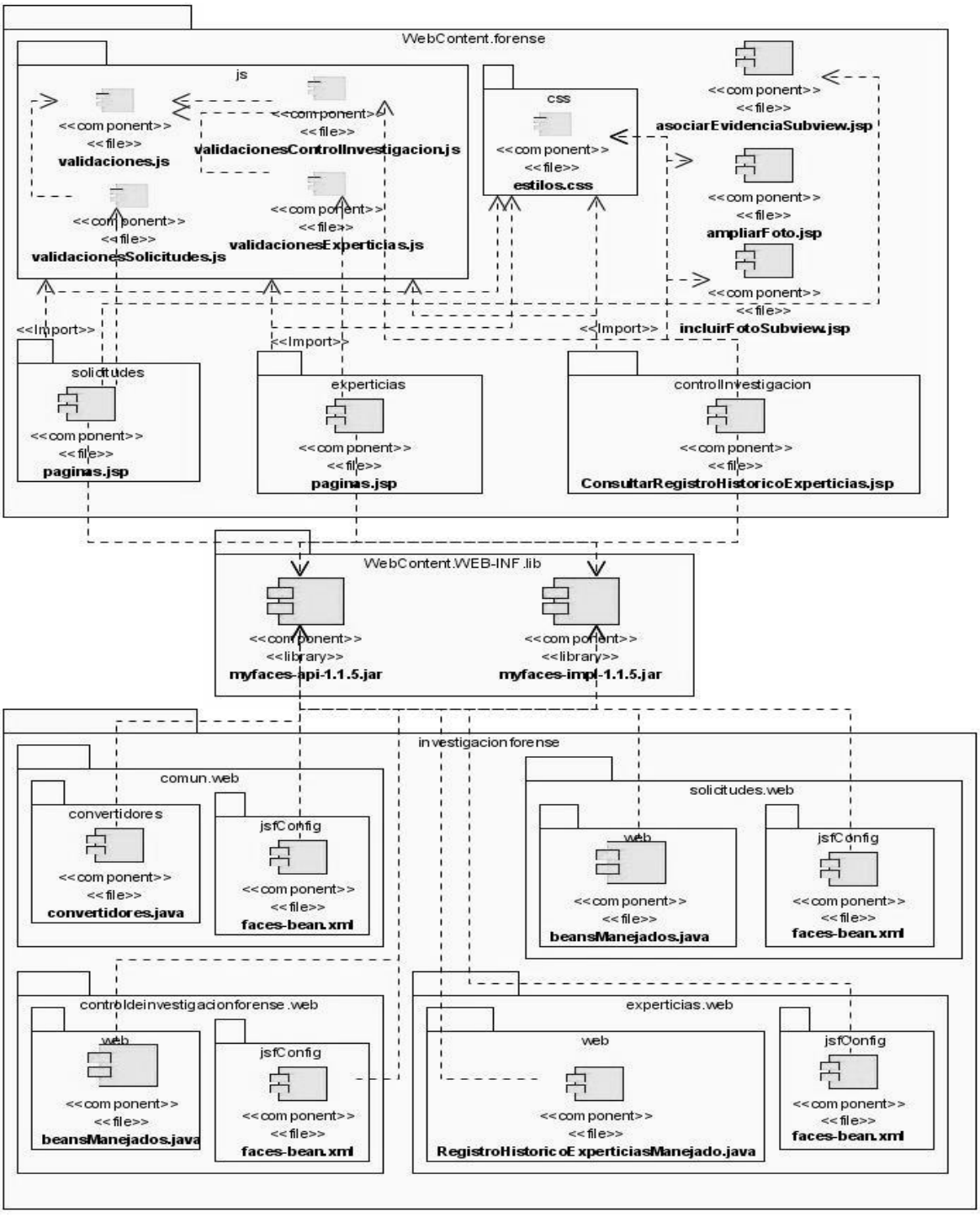

Fig. 27 Diagrama del Subsistema de Implementación de la Capa UI.

El componente *beansManejados.java* representa a todos los beans manejados del Subsistema de Implementación correspondiente. Lo mismo ocurre con el componente *paginas.jsp* y con *convertidores.java*. A continuación se representa el diagrama de componente del subsistema de Implementación *solicitudes.bean* y *WebContent.forense.solicitudes*. En el **Anexo # 6** se encuentran los restantes diagramas.

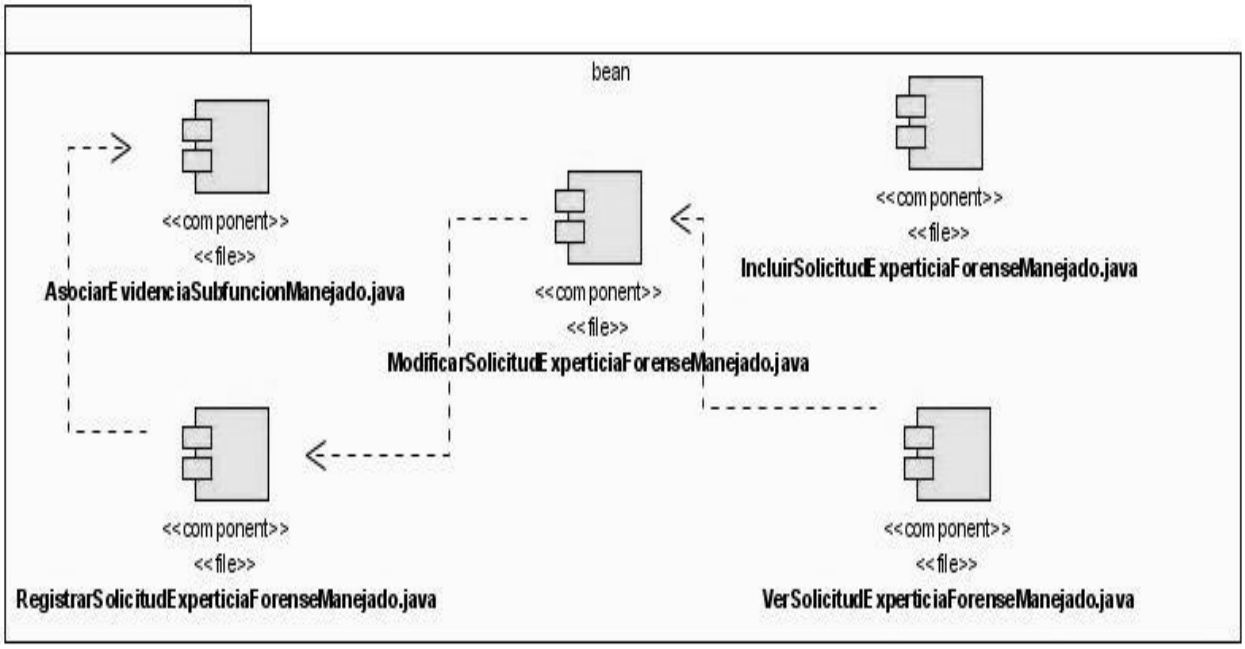

Fig. 28 Diagrama de Componentes del Subsistema de Implementación *solicitudes.bean.*

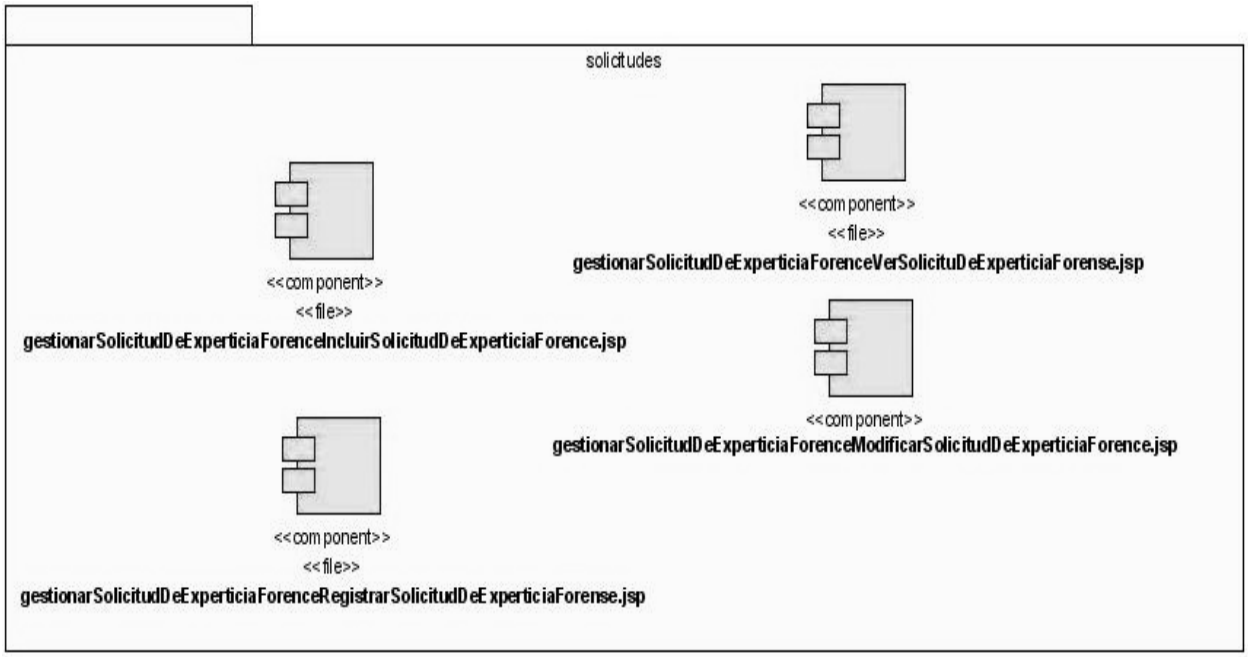

Fig. 29 Diagrama de Componentes del Subsistema de Implementación *WebContent.forense.solicitudes.*

2.5.3 Modelo de Despliegue

EL Modelo de Despliegue describe cómo y dónde el sistema será puesto en funcionamiento. Las estaciones de trabajo, dispositivos y procesadores son reflejados como nodos y su estructura interna puede ser representada adicionando otros nodos o artefactos (29).

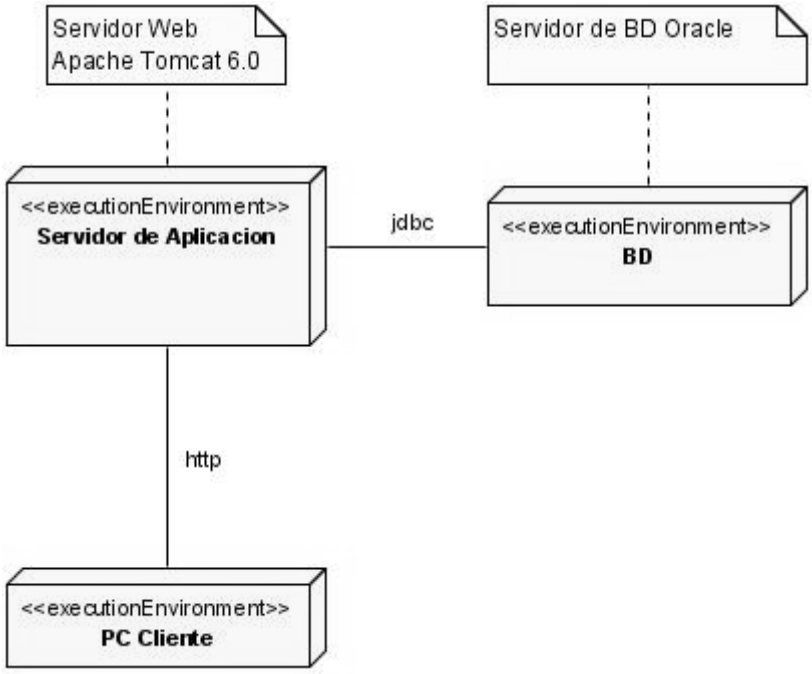

Fig. 30 Diagrama de Despliegue.

Conclusiones

En el capítulo se realizó el análisis, diseño e implementación de la capa UI del módulo de Investigación en Ciencias Forenses del SIIPOL. Se identificaron y siguieron pautas de diseño a lo largo del proceso de diseño y desarrollo de las UI, apoyados en la metodología escogida, lo que facilitó el resultado final. En la etapa de análisis y diseño se construyeron los diagramas de clases del análisis, diagramas de clases del diseño y diagramas de contrato entre paquetes por cada CU, además se describieron las clases de diseño más significativas. Una vez concluida la implementación e integración de todos los CU, se recoge en este capítulo, los artefactos generados en ese flujo de trabajo.

CONCLUSIONES GENERALES

Durante el desarrollo del presente trabajo, se pude concluir que es importante integrar a los sistemas de gestión de información policial, funcionalidades para el manejo y consulta de los temas referentes a servicios forenses.

Con el uso de la metodología DDWUI se puede lograr un buen diseño y desarrollo de las interfaces de usuario cumpliendo con las pautas establecidas de diseño gráfico y arquitectura de la información. La aplicación de la tecnología JSF, facilitó el trabajo de desarrollo, a pesar de ser un framework nuevo y de la complejidad del sistema.

Con el análisis, diseño e implementación de la capa UI, del sistema modelado, se ha dado cumplimineto al objetivo general que se planteó. El Módulo de Investigación en Ciencias Forenses integrado al SIIPOL aportará grandes beneficios a los funcionarios de la CNCF, del CICPC y a los órganos judiciales.

RECOMENDACIONES

Analizando los resultados del presente trabajo se recomienda:

Continuar las pruebas de calidad e integración, para certificar la eficiencia de la parte del subsistema desarrollado, con el fin de entregar al cliente un producto de excelencia.

Refactorizar la implementación de los CU, en la capa de presentación del subsistema, para optimizar el rendimiento de la aplicación y la reutilización de código.

Transmitir experiencias a otros proyectos que utilicen metodología, herramientas y tecnologías similares, además de proponer el uso de la metodología DDWUI para el diseño y desarrollo de las interfaces UI.

Usar el presente trabajo como material de consulta en próximas etapas del desarrollo del proyecto, utilizándolo como apoyo para la capacitación de nuevos programadores, diseñadores y montadores UI.

Analizar la posibilidad de desarrollar un proyecto similar para el Instituto de Medicina Legal de Cuba.

REFERENCIAS BIBLIOGRÁFICAS

1. **González, Robin José Rodríguez.** Análisis sobre la delincuencia en Venezuela. 2006. [Disponible en:] www.soberania.org/Articulos/articulo_2082.htm.

2. **Velásquez, Hugo Acero.** SITUACIÓN DE VIOLENCIA Y DELINCUENCIA DE VENEZUELA Y CONCENTRACIÓN DELINCUENCIA EN CARACAS. 2006. [Disponible en:] http://www.comunidadesegura.org/files/active/0/diagnostico_violencia_y_delincuencia_Venezuela_y_Caracas.pdf.

3. **Núñez, Reynaldo Roselló.** Ingeniería de Requerimientos del Proceso. Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2007.

4. **Ediciones, Tibidabo.** Helios.Gestión Policial Informatizada. 2008. [Disponible en:] http://www.tibidaboediciones.com/.

5. **Digitales, Mapas.** SEGURIDAD CIUDADANA Y GESTIÓN POLICIAL. [Disponible en:] http://www.dmapas.cl/productos_stegpol.htm.

6. **eLABORO.** ePatrol. [Disponible en:] http://www.epatrol.es/.

7. **Padrino, Iris Armas.** Reconocen desarrollo de la medicina legal cubana. 2006. [Disponible en:] http://www.cnctv.cubasi.cu/noticia.php?idn=4484.

8. **Sanchez, María A. Mendoza.** Metodologías De Desarrollo De Software. 2004. [Disponible en:] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

9. **Álvarez, Camilo Javier Solis y Figueroa Díaz, Roberth Gustavo.** Metodologías Tradicionales vs. Metodologías Ágiles. 2007. [Disponible en:] http://www.mygnet.net/manuales/software/metodologias_tradicionales_vs_dot_metodologias_agiles.1515.

10. **UCI, Ingeniería de Software I.** Conferencia 1 Introducción a la Ingeniería de Software. 2007-2008. [Disponible en:] http://teleformacion.uci.cu/mod/resource/view.php?id=6655.

11. **Microsoft.** Microsoft Solutions Framework. [Disponible en:] http://www.microsoft.com/spanish/MSDN/estudiantes/ingsoft/planificacion/msf.mspx.

12. **Canós, José H., Letelier, Patricio y Penadés, Mª Carmen.** Métodologías Ágiles en el Desarrollo de Software. [Disponible en:] http://www.willydev.net/descargas/prev/TodoAgil.Pdf.

13. **Molpeceres, Alberto.** Procesos de desarrollo: RUP, XP y FDD. 2002. [Disponible en:] http://www.javahispano.org/contenidos/es/procesos_de_desarrollo/.

14. **Rumbaugh, James, Jacobson, Ivar y Booch, Grady.** *El Lenguaje Unificado de Modelado.Manual de Referencia.* 2000.

15. **Wikipedia.** Lenguaje de programación Java. 2008. [Disponible en:] http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java.

16. **Molpeceres Touris, Alberto y Pérez Mariñán, Martín.** Un sistema software que pudiera ser reutilizado.2002. [Disponible en:] http://www.willydev.net/InsiteCreation/v1.0/willycrawler/2008.05.02.articulo.arquitectura%20empresarial%20y%20software%20libre,%20j2ee.pdf.

17. **Extremo Baigorri, Unai y Sotomayor Basilio, Borja.** La Plataforma .NET: ¿El Futuro de la Web? [Disponible en:] http://www.telecentros.info/articulosd.asp?id=5.

18. **Pereyra, Martín Tuesta.** Sistemas de Información. [Disponible en:] http://www.elprisma.com/apuntes/curso.asp?id=13324.

19. **Hall, Marty.** Servlets y JSP. [Disponible en:] http://www.programacion.net/java/tutorial/servlets_jsp/.

20. **Cavaness, Chuck.** *Programming Jakarta Struts.* s.l. : O'Reilly, 2004.

21. **García, Alejandro Pérez.** Introducción a JSF (Java Server Faces). [Disponible en:] http://www.desarrolloweb.com/articulos/2380.php.

22. **González, Andrés.** Struts vs. JSF. [Disponible en:] 2007. http://coyotevil.blogspot.com/2007/05/struts-vs-jsf.html.

23. **Musciano, Chuck y Kennedy, Billy.** *HTML La guia completa.* s.l. : O' Reilly, 1999.

24. **Pérez, Javier Eguíluz.** Introducción a CSS.2007. [Disponible en:] http://www.librosweb.es/css.

25. **Pérez, Javier Eguíluz.**. Introdución a JavaScript. 2008. [Disponible en:] http://www.librosweb.es/javascript.

26. **Pérez, Javier Eguíluz.**. Introducción a Ajax. 2008. [Disponible en:] http://www.librosweb.es/ajax.

27. **UCI, Ingeniería de Software I.** Conferencia 6 Fase de Inicio. Flujo de Análisis y Diseño. Modelo de Análisis. 2007-2008. [Disponible en:] http://teleformacion.uci.cu/mod/resource/view.php?id=10349.

28. **UCI, Ingeniería de Software II.** Material de apoyo.Conferencia de Diseño. 2008. [Disponible en:] http://teleformacion.uci.cu/mod/resource/view.php?id=21363.

29. **UCI, Ingeniería de Software II.** Conferencia 4 Flujo de Implementación. 2007. [Disponible en:] http://teleformacion.uci.cu/mod/resource/view.php?id=22199.

BIBLIOGRAFÍA

Ajax 4 JSF Developer Guide. 2007.

Avellón, Iván Zaera. Guía de referencia de JSF. [Disponible en:] http://www.adictosaltrabajo.com/tutoriales.php?id=32.

Bonilla, Beatríz. validaciones y Conversiones en JSF. [Disponible en:] http://www.adictosaltrabajo.com/tutoriales.php?id=32.

Crane, Dave, Pascarello, Eric and James, Darren. *Ajax in Action.* Greenwich : Manning, 2006.

Flanagan, David. *Java en pocas palabras.* s.l. : O' Reilly, 1998.

Gallardo, David, Burnette, Ed y McGovern, Robert. Eclipse in Action. Manning, 2003.**García de Jalón, Javier and autores, grupo de.** *Aprenda Java como si estuviera en primero.* San Sebastián : s.n., 2000.

Mann, Kito D. *Java Server Faces in Action.* Greenwich : Manning, 2005.

Patzer, Andrew. *JSP Ejemplos Prácticos.* s.l. : Grupo Anaya, 2003.

Ramos, Juan Alonso. Introducción a Ajax 4 Jsf. [Disponible en:] http://www.adictosaltrabajo.com/tutoriales.php?id=32.

RichFaces Developer Guide. 2007.

Rumbaugh, James, Jacobson, Ivar and Booch, Grady. *Proceso Unificado de Desarrollo de Sofware.* Madrid : Addison Wesley, 2000.

GLOSARIO DE TÉRMINOS

ALBA: Siglas de la Alternativa Bolivariana para las Américas.

Applet: Componente de software que corre en el contexto de otro programa, por ejemplo un navegador web. El Applet es una aplicación Java, por lo cual debe ser ejecutado sobre una máquina virtual.

Artefacto: Una parte de la información que es producida, modificada, o usada por un proceso, define un área de responsabilidad, y está sujeta al control de versión. Un artefacto puede ser un modelo, un elemento del modelo, o un documento. Un documento puede adjuntar otros documentos.

Beans: Es una clase de Java.

Bytecode: Es un código intermedio más abstracto que el código máquina. Un fichero binario producido por el compilador.

Caso 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.

Cliente: Una persona u organización, interna o externa a la organización productora que toma responsabilidad financiera por el sistema. El cliente es el último destinatario del producto desarrollado y sus artefactos.

Dispatcher Servlet: Controlador frontal adonde se dirige inicialmente la petición en Spring MVC y que posteriormente es enviada hacia otros componentes para su ejecución.

El Object Management Group (OMG): En Español Grupo de Gestión de Objetos, es un consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos.

Ente: Funcionario que solicita la solicitud de experticia forense**.**

Etiqueta:(*tag*) es una marca con tipo que delimita una región en los lenguajes basados en XML.

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

Exhumación: Es la acción de extraer un cadáver previamente inhumado.

Experticia: Es un informe donde se plasman los resultados obtenidos al realizar un análisis pericial.

Formulario: Plantilla o página con espacios vacíos que han de ser rellenados con alguna finalidad

Framework: Marco de trabajo, solución reutilizable y extensible.

Hibernate: Framework libre y de código abierto objeto relacional para el acceso a datos.

HTTP: Siglas en inglés de HyperText Transfer Protocol (protocolo de transmisión del hipertexto).

Interfaz de usuario: Conjunto de componentes empleados por los usuarios para comunicarse con las computadoras.

Interfaz gráfica: Tipo de interfaz que permite a los usuarios comunicarse con un programa mediante funcionalidades gráficas. Normalmente incluyen una combinación de gráficos, barras de menús e iconos.

Interfaz: Colección de operaciones que son usadas para especificar un servicio de una clase o un componente.

Iteración: Repetición de una serie de instrucciones en un programa de computadora.

Librerías: Conjunto de subprogramas utilizados para desarrollar software.

Máquina virtual: Ambiente sobre el cual se desarrollan y ejecutan aplicaciones Java.

Metodología de desarrollo: Se refiere a los métodos de investigación en una ciencia

Modelo: Cosa que ha de servir de objeto de imitación. Objeto, construcción u otra cosa con un diseño del que se reproduce más iguales. Esquema teórico de un sistema o de una realidad compleja que se elabora para facilitar su comprensión y estudio.

Módulo: Término que denota una unidad para el almacenamiento y manipulación del software. La palabra no corresponde a una única estructura de UML, sino que incluye varias estructuras.

Nomencladores: Clases que sus atributos guardan valores fijos.

Open Source: Código abierto (del inglés *open source*) es el término con el que se conoce al software distribuido y desarrollado libremente.

Pauta: Lo que sirve como norma o modelo para realizar algo.

Proyecto: Es la entidad que contiene el grupo de tareas necesarias para desarrollar un determinado producto.

Refactorización: Es una técnica de la ingeniería de software para reestructurar un código fuente, alterando su estructura interna sin cambiar su comportamiento externo.

Release: En español, "revisión" o "versión". Versiones de una aplicación de software en proceso de construcción. El número de versión suele indicar el avance de los cambios.

Requisitos: Conjunto de características que debe tener un producto o servicio para satisfacer las necesidades y expectativas del cliente.

Rol: Papel, cometido o función que tiene o desempeña que interpreta un actor.

Servidor: Un servidor es un ordenador de gran potencia, que se encarga de "prestar un servicio" a otros ordenadores que se conectan a él.

Sistema de gestión de información: Conjunto de elementos que interactúan entre sí con el fin de apoyar las actividades que se realizan en una organización o para automatizar los procesos de trabajo que se efectúan dentro de la misma.

Sistema: Conjunto de procedimientos, normas o métodos integrados para la construcción de un fin.

Software: Término genérico utilizado en informática para designar programas o fragmentos de programas.

Web: Red de documentos HTML intercomunicados y distribuidos entre servidores web.