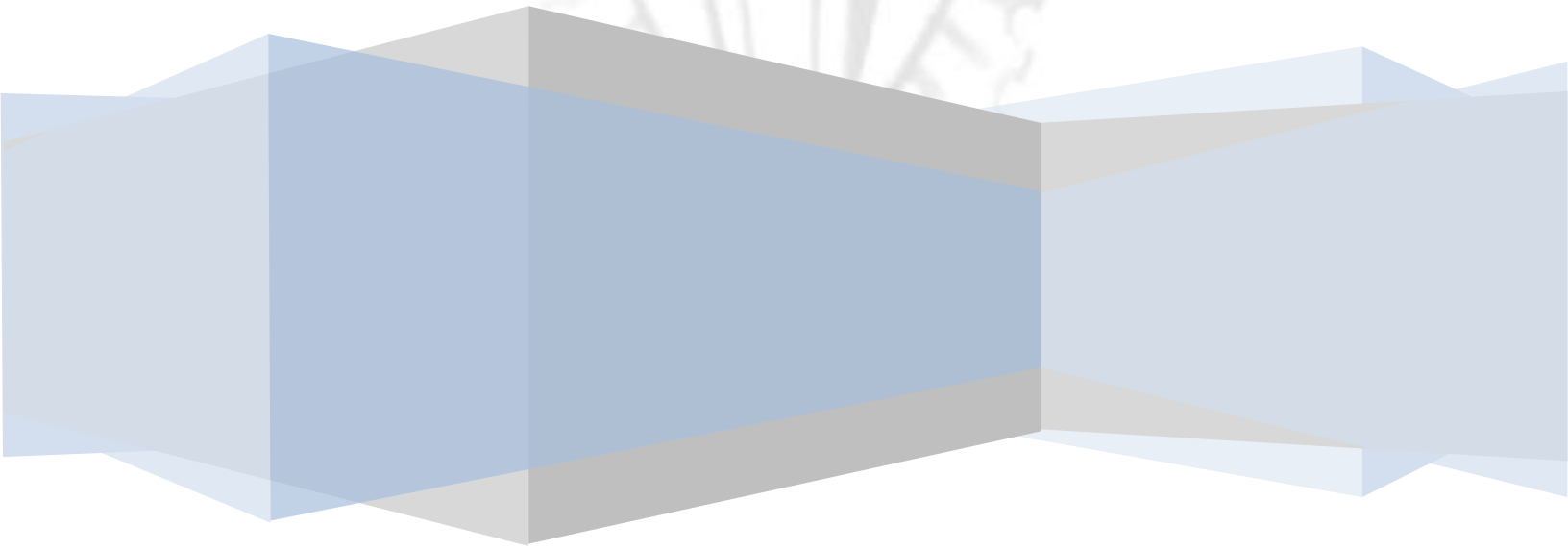
Universidad de las Ciencias Informáticas

Análisis, diseño e implementación del módulo Aprehensión del SIIPOL

Trabajo de diploma correspondiente al rol Diseñador-Programador del proyecto CICPC.

Autor: Humberto Rivero Guevara

Tutor(a): Ing. Diana García Vicente



Ciudad de La Habana, Junio de 2008

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de la presente tesis y reconozco 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 Junio del año 2008.

_____	_____
Firma del Autor	Firma del Tutor
Humberto Rivero Guevara	Ing. Diana García Vicente

Agradecimientos

Creo que agradecer es la parte más difícil, no obstante voy a arriesgarme:

Quisiera darles las gracias a la Revolución por la oportunidad de haber entrado a esta escuela y de haberme permitido superarme a lo largo de los años.

Aunque nada de esto hubiera sido posible sin la ayuda de mis padres que los dos se convirtieron en mi ejemplo de toda la vida, por eso creo que, incluso agradecerles este trabajo de diploma es poco para todo lo que les debo.

Agradecerle a mi familia por siempre estar ahí cuando las necesité, a mis abuelos que aunque ya algunos no estén los sigo queriendo, a mis tíos, a mis primos, en fin a todos.

Y por supuesto que no podía dejar de agradecerle a la persona que más me ha ayudado en mis últimos años de vida, a ti Sandri, a ti tampoco tengo como agradecerte, y gracias por existir, no sé que hubiera sido de mí. .

Antes dije mi familia, no obstante no sé ni cómo decirles gracias a mí segunda familia a Marta, Aida, Teresa, Andros, Yensi, y Floco.

Gracias a mi tutora Diana y a Yander.

Y no por último menos importante a mis amigos, a mis profesores de todos estos años, creo que todos han puesto un poco de su parte para lograr que este ser ya sea un Ingeniero.

Dedicatoria

A las personas más especiales que he tenido en toda mi vida, mis padres: Vivian y Humberto. Ellos me han enseñado mucho de lo que sé y siempre han estado ahí para mí. Por eso, mis ángeles guardianes, a ustedes va dedicado este trabajo que con tanto empeño he desarrollado. Los quiero con toda mi alma, a los dos; y los extraño.

Resumen

La documentación que se presenta recoge el proceso de desarrollo del subsistema Aprehensión del SIIPOL, el cual forma parte de la aplicación que actualmente se desarrolla en la facultad 8 de la Universidad de las Ciencias Informáticas incluida como parte de un contrato con el Cuerpo de Investigaciones Científicas, Penales y Criminalísticas de la hermana República Bolivariana de Venezuela. El Departamento de Aprehensión ha sido una de las áreas dentro del CICPC que fueron evaluadas como posibles a informatizar, en el tienen lugar diferentes procesos que incluyen desde que la persona es detenida hasta que es liberada o enviada a algún retén. Actualmente no existe ningún sistema informático que apoye dichos procesos, por lo que todo se realiza de forma manual y el manejo de los grandes volúmenes de información es lento y laborioso, lo que trae consigo otras deficiencias. Como solución al problema planteado se desarrolló una aplicación web, basada en el lenguaje Java, haciéndose uso de algunos de los frameworks que son soportados por dicho lenguaje, como son los casos de: JSF, Spring, Hibernate, entre otros. Gran parte de las funcionalidades del producto se han desarrollado teniendo en cuenta las actuales tendencias en materia de aplicaciones web, soportadas bajo el conjunto de tecnologías Ajax. Usando como guía la metodología de desarrollo de software RUP, se realiza el análisis, diseño e implementación, obteniéndose una aplicación que cumple con las condiciones generadas durante el proceso de Ingeniería de Requerimientos; lista para entrar en el flujo de trabajo de pruebas.

Índice

INTRODUCCIÓN 1

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 6

1.1 Introducción 6

1.2 Sistemas similares 6

Sistema Integrado de Información Policial (SIIPOL) 6

STEGPOL- Sistema Territorial de Emergencias y Gestión Policial. 7

Sistema de Gestión Penitenciaria (SIGEP) 8

1.3 Análisis de los principales modelos, metodologías y estándares para el desarrollo de software. Selección de la metodología que apoya la solución del problema. 9

Rational Unified Process (RUP) 10

Extreme Programing (XP) 11

SCRUM 13

Lenguaje de modelado UML 15

1.4 Tendencias y tecnologías actuales. Selección de las herramientas de desarrollo. 16

Principales lenguajes de programación investigados 17

Entorno de Desarrollo Integrado (IDE) 20

Herramientas CASE de Modelado con UML 22

Frameworks soportados por Java 24

1.5 Conclusiones 32

CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA PROPUESTA DE SISTEMA 33

2.1 Introducción 33

2.2 ¿Qué son el análisis y el diseño? 33

2.3 Previo al análisis. 35

Procesos elementales del negocio 36

Requerimientos funcionales 36

Descripción de los Casos de Uso 38
Diseño de las interfaces de usuario 42
2.4 Análisis 46
2.5 Diseño 51
Diagramas de clases de Diseño Web 54
Descripciones textuales de algunas de las clases: 68
Entidades persistentes y diagrama entidad relación 78
2.6 Conclusiones 81
CAPÍTULO 3: IMPLEMENTACIÓN 82
3.1 Introducción 82
3.2 Reseña 82
3.3 Diagrama de paquetes y componentes. 86
3.4 Conclusiones 93
CONCLUSIONES 94
RECOMENDACIONES 95
REFERENCIAS BIBLIOGRÁFICAS 96
BIBLIOGRAFÍA 99
GLOSARIO 101
ANEXOS 104

INTRODUCCIÓN

Al asumir la primera magistratura de la República Bolivariana de Venezuela el Teniente Coronel Hugo Rafael Chávez Frías, en febrero del año 1999; se inicia el proceso constituyente para la elaboración de una nueva constitución. En vista de la discusión del artículo 332 referente a los Órganos de Seguridad Ciudadana, un grupo de funcionarios del *Cuerpo Técnico de Policía Judicial (*CPTJ), se acerca a la Asamblea Nacional para participar como correlatores del artículo en discusión. El presidente Hugo Chávez, dando fiel cumplimiento a lo establecido en la Constitución Bolivariana de Venezuela, promulga el Decreto con fuerza de Ley de los "Órganos de Investigaciones Científicas, Penales y Criminalísticas". La ley estableció, la organización administrativa del Cuerpo de Investigaciones Científicas, Penales y Criminalísticas (CICPC), haciéndose efectiva el 15 de mayo de 2003.

El CICPC es una Institución que garantiza la eficiencia en la investigación del delito, mediante su determinación científica; asegurando que el ejercicio de la acción penal conduzca a una administración de justicia. Con la visión de ser la institución indispensable, con la finalidad de alcanzar el más alto nivel de credibilidad nacional e internacional en la investigación del fenómeno delictivo y la criminalidad violenta; además de cumplir el decreto que instituye las competencias del CICPC como órgano principal de investigaciones penales al servicio del Estado; el gobierno bolivariano se empeña en superar las deficiencias que presenta en la actualidad.

Al realizar el estudio de la situación general que presenta la Institución, se identificaron un conjunto de problemáticas en los procesos que desarrolla: existe un gran volumen de información, se requiere de mucho tiempo para obtener las estadísticas relacionadas con un caso en particular, la información no posee alto grado de fiabilidad, ya que algunas áreas, no cuentan con sistema automatizados para el control del proceso, por lo que se realiza de forma manual. Estas deficiencias involucran a todas las áreas del CICPC.

Con el objetivo de minimizar tales limitaciones, se pretende: Desarrollar un nuevo Sistema Integrado de Información Policial, tomando como referencia el sistema existente SIIPOL, que incremente las posibilidades de uso de la información, al agregar nuevas funciones al mismo, mejorar las ya existentes, y brindar una interfaz de fácil manejo para usuarios inexpertos que necesitan de tiempo para la solución de cada caso.

Tras un previo análisis de los procesos en la institución se han detectado una serie de requisitos funcionales y no funcionales con los cuales el nuevo sistema debe cumplir para mejorar el nivel de respuesta a las necesidades de seguridad del ciudadano venezolano y facilitar la sistematización del trabajo en sus dependencias.

Dada esta situación se realizó un levantamiento de requisitos de CICPC y se modeló un sistema informático de 14 módulos, uno de los cuales es el módulo de Aprehensión.

Al Departamento de *Aprehensión* son conducidas aquellas personas que son detenidas por haber cometido algún delito, ya sea porque existiera alguna orden de aprehensión previa o por haber sido atrapado infraganti. Cuando la persona llega a dicho departamento le son tomados un conjunto de datos los cuales pasan a formar parte de la planilla de ingreso del detenido, tales como sus datos personales, el delito o la orden de aprehensión por la que fue detenido, entre otros. Durante la estancia del detenido en el departamento, se realizan una serie de acciones por parte de los funcionarios; las cuales son recogidas en las actas del detenido, y son asociadas a la carpeta del detenido que contiene la planilla de ingreso mencionada anteriormente. Igualmente, cuando el detenido es puesto en libertad o enviado a algún retén a cumplir sentencia, son almacenados estos datos en la boleta de excarcelación y la boleta de traslado al retén respectivamente.

Todos estos procesos marchan a la par de la investigación por la cual fue detenida la persona, por tanto tributan directamente al caso. De ahí la necesidad del procesamiento de dicha información de manera rápida, eficiente, confiable y segura.

Actualmente en dicho departamento no existe ningún sistema informático que sustente dicho negocio, lo explicado anteriormente provoca que se tornen lentas todas las investigaciones asociadas.

En función de desarrollar un sistema informático que tribute a dicha institución, se propone como **problema científico:** ¿Cómo garantizar el cumplimiento de los requisitos funcionales y no funcionales asociados al módulo Aprehensión del SIIPOL?, para lo que se define como **objetivo general de la investigación:** Desarrollar una herramienta informática que cumpla con los requisitos funcionales y no funcionales obtenidos como resultado de aplicarle Ingeniería de Requerimientos a los procesos de gestión de la investigación que se llevan a cabo en el Departamento de Aprehensión.

De lo planteado anteriormente se deriva como **objeto de estudio:** *Proceso de desarrollo del software de gestión SIIPOL.* Enfocando la investigación hacia el **campo de acción**: Análisis, Diseño e Implementación del módulo Aprehensión del sistema SIIPOL.

Por tal razón, se plantea la **idea a defender:** La aplicación correcta de la metodología de desarrollo de software para el análisis, diseño e implementación del módulo de Aprehensión, posibilita la elaboración de un sistema informático que cumple con los requisitos funcionales y no funcionales definidos para el mismo.

Los **aportes prácticos** esperados son:

I. Contribuir al procesamiento y gestión de la información de las acciones que se realizan con el ciudadano en el departamento de Aprehensión.

II. Facilitar la sistematización del trabajo de las delegaciones, subdelegaciones y dependencias en general.

III. Mejorar el nivel de respuesta a las necesidades de seguridad del ciudadano venezolano.

La **novedad científica** de la misma, radica en el desarrollo de una aplicación para múltiples plataformas haciendo uso del lenguaje Java y de las tecnologías que dicho lenguaje soporta como son sus frameworks de desarrollo; los cuales propician la utilización de novedosos paradigmas de programación, como es: la Programación Orientada a Aspectos (AOP), que complementa la Programación Orientada a Objetos (POO) proponiendo otra manera de pensar sobre la estructura de un programa; soporta también el uso de

tecnologías como AJAX, lo cual constituye un nuevo modo de concepción para las aplicaciones web, permitiendo desarrollar sistemas con una interfaz web mucho más interactiva y amigable.

Para cumplir con lo expuesto anteriormente se citan un conjunto de **tareas investigativas**:

1. Realizar el diseño teórico de la investigación.
2. Investigar acerca de otras aplicaciones o soluciones similares.
3. Investigar acerca de los lenguajes y metodologías de desarrollo de software existentes, así como las facilidades que brindan los mismos.
4. Analizar el módulo Aprehensión del sistema SIIPOL.
5. Diseñar el módulo Aprehensión del sistema SIIPOL.
6. Implementar el módulo Aprehensión del sistema SIIPOL.
7. Elaborar la documentación con los artefactos de los flujos de trabajo.

El presente trabajo está estructurado en tres capítulos principales:

Capítulo 1. Fundamentación Teórica: Este capítulo trata acerca de algunos elementos teóricos que soportan la investigación, tales como: sistemas similares existentes vinculados al campo de acción y su descripción, selección de herramientas y tecnologías que se ajusten a lo requerido, entre otros.

Capítulo 2. Análisis y diseño de la propuesta de sistema: Este capítulo inicia con una breve conceptualización sobre el análisis y el diseño, además de presentar una breve descripción de los artefactos utilizados como entrada para el análisis. Se presentan los diagramas de clases del análisis, que constituyen un aporte significativo para el diseño del sistema, además de los diagramas de clases Web, que reflejan de una forma clara el funcionamiento del sistema, las clases del mismo y la descripción de la algunas de ellas.

Capítulo 3. Implementación: En este capítulo se describe como está implementado el sistema, a través de los diagramas de componentes y el diagrama de despliegue. Se exponen y detallan los diferentes aspectos que se tuvieron en cuenta durante su desarrollo.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 *Introducción*

Al iniciarse el desarrollo de un producto de software los primeros pasos que se dan son en función de definir el tipo de aplicación y su ambiente de desarrollo. El presente capítulo aborda la fundamentación teórica del desarrollo del módulo Aprehensión del SIIPOL, define el marco teórico del tema, el estudio de sistemas similares existentes vinculados al campo de acción; así como las tendencias, tecnologías, metodologías y herramientas actuales, realizando una selección de aquellas que serán utilizadas durante el desarrollo del proyecto.

1.2 *Sistemas similares*

El CICPC es el órgano principal de Venezuela responsable de la investigación de los hechos delictivos que ocurren en cada uno de los estados venezolanos, por lo que cuenta con dependencias desplegadas por todo el país.

En aras de brindar una mayor protección y seguridad a cada uno de los venezolanos de un modo rápido y eficiente, y en este caso en particular lo que concierne al Departamento de Aprehensión, surge la necesidad de investigar acerca de otros sistemas en el mundo o en el marco nacional que pudiesen apoyar todo este proceso. Como resultado se obtuvo lo siguiente:

Sistema Integrado de Información Policial (SIIPOL)

El Sistema Integrado de Información Policial es una aplicación informática que el Cuerpo de Investigaciones Científicas, Penales y Criminalísticas de la República Bolivariana de Venezuela utiliza

para llevar a cabo las tareas fundamentales que realizan. El SIIPOL está desarrollado sobre una tecnología actualmente obsoleta y sobre el lenguaje de programación Adabas-Natural y servidores de base datos SUN 6500.

Este sistema no abarca todos los procesos que se realizan en esta institución, lo cual repercute de manera negativa en el buen desempeño de los funcionarios a la hora de esclarecer los hechos delictivos en el tiempo necesario para ello. Además, dentro de los procesos para los cuales no brinda soporte, se encuentran aquellos que se llevan a cabo dentro del Departamento de Aprehensión. (1)

STEGPOL- Sistema Territorial de Emergencias y Gestión Policial.

Es un Sistema de Información Geográfica sobre una Plataforma Nacional Común de Información aplicada al "Sistema de Emergencias Nacionales" y al "Sistema Territorial de Gestión Policial" que actualmente operan en Chile.

Se caracteriza por ser un sistema con tecnología de punta aplicado al "control territorial de la gestión policial", el cual identifica en forma veraz y efectiva dónde geográficamente se están cometiendo o se han cometido actos delictuales a nivel territorial, apoyado por sistemas de información en-línea desde el lugar de los hechos; lo cual permite la acción rápida y coordinada entre los diferentes actores encargados de la seguridad ciudadana a nivel nacional.

Integra a las diferentes entidades como Carabineros, Investigaciones, Ministerio del Interior y Municipios, entre otros, en una Plataforma Nacional Común de Información que permite el intercambio de datos y que sirve de apoyo a la gestión operacional regional o comunal, donde dichas instituciones están interconectadas entre sí, en especial con diferentes Divisiones o Departamentos de Carabineros, tales como Jefaturas de Zona, Prefecturas, Comisarías, Sub-Comisarías, Tenencias y por último los Retenes, más el apoyo directo y coordinado entre Carabineros y el área de Seguridad de los Municipios.

Vía la Intranet del Estado o vía comunicación en Banda Ancha - Internet se puede llegar a acceder a un Sistema Territorial de Emergencias y Gestión Policial (STEGPOL) montado en la Web con diferentes

contraseñas de acceso restringido tanto para los Usuarios Operadores encargados de ingresar o modificar datos en línea, como para aquellos que solo tengan acceso a consultar. (2)

Sistema de Gestión Penitenciaria (SIGEP)

El SIGEP es uno de los proyectos surgidos a raíz del ALBA, el cual es desarrollado por un grupo de estudiantes y profesores de la Facultad 4 de la Universidad de las Ciencias Informáticas. Sus funcionalidades están directamente vinculadas a los procesos que se llevan actualmente dentro de las prisiones venezolanas. No abarca el conjunto de procesos que se realizan dentro del CICPC, pero, aunque está destinado a otra área, su arquitectura sirvió como punto de partida para la investigación previa a definir la del proyecto CICPC, pues incluso en los términos de contrato de ambos proyectos, las herramientas y tecnologías solicitadas por el cliente eran similares.

Es una aplicación web desarrollada en java con los frameworks: Hibernate, Spring y para la parte de presentación se utiliza el módulo de Spring: Spring MVC; Acegi (que actualmente ha pasado a llamarse SpringSecurity) para manejar la seguridad del sistema.

Los sistemas descritos anteriormente a pesar de que sirven como apoyo a distintas instituciones relacionadas con los temas policiales, sus alcances se quedan muy cortos en comparación con las necesidades actuales del CICPC. Por cuestiones propias de seguridad, encontrar bibliografía o algunas fuentes que brinden información acerca de sistemas como estos, es un poco difícil; ya que las compañías que lo desarrollan o las mismas instituciones que los utilizan, se reservan sus características para evitar algún tipo de ataque al sistema o a la institución. Producto a esto se llegó a la necesidad de desarrollar un sistema nuevo, permitiendo además que en determinadas áreas de la institución se realizara algún tipo de restructuración de los procesos, si es que así lo requiriese.

1.3 *Análisis de los principales modelos, metodologías y estándares para el desarrollo de software. Selección de la metodología que apoya la solución del problema.*

A nivel internacional, las instituciones y empresas dedicadas a la industria de software, emplean en su actividad de desarrollo, modelos, metodologías o procedimientos estándares para desarrollar, instalar y mantener un producto de este tipo.

El objetivo de un proceso de desarrollo de este tipo, es elevar la calidad del software (en todas las fases por las que pasa) a través de una mayor transparencia y control sobre el proceso. Da igual el alcance que se desee, hay que producir lo esperado en el tiempo y con el costo esperado.

Es labor del proceso de desarrollo hacer que esas medidas para aumentar la calidad sean reproducibles en cada desarrollo. Estas metodologías establecen un conjunto de actividades que definen cómo se debe hacer el software, quién debe hacer cada actividad, cuándo hacerla y qué se debe hacer.

La implantación de un proceso de desarrollo es una labor más a medio-largo plazo que una labor de resultados inmediatos. Cuesta tiempo que los trabajadores se adapten al proceso, pero una vez superado, la inversión se recupera con creces. Es por ello que no tiene sentido ajustarse a un proceso al pie de la letra, sino que hay que adaptarlo a las necesidades y características de cada empresa, equipo de trabajo o casi a cada proyecto. (3)

En los últimos tiempos la cantidad y variedad de los procesos de desarrollo ha aumentado de forma impresionante, sobre todo teniendo en cuenta el tiempo que estuvo en vigor como ley única el famoso desarrollo en cascada. Se podría decir que en estos últimos años se han desarrollado dos corrientes en lo referente a los procesos de desarrollo, los llamados métodos pesados y los métodos ligeros. La diferencia fundamental entre ambos es que mientras los métodos pesados intentan conseguir el objetivo común por

medio de orden y documentación, los métodos ligeros (también denominados métodos ágiles), tratan de mejorar la calidad del software por medio de una comunicación directa e inmediata entre las personas que intervienen en el proceso.

Rational Unified Process (RUP)

RUP es uno de los procesos más generales de los existentes actualmente, ya que en realidad está pensado para adaptarse a cualquier proyecto, y no tan solo de software. Unifica completamente a un equipo de desarrollo de software y optimiza la productividad de cada uno de los miembros del equipo brindándoles la experiencia de los líderes de la industria y las lecciones aprendidas a través de miles de proyectos. Está fundamentada en un enfoque orientado a modelos de desarrollo basado en componentes, utilizando para ello el Lenguaje de Modelado Unificado (UML, Unified Modeling Language) el que define técnicas de análisis y diseño que ayudan a la confección de una solución sólida de software.

RUP se caracteriza por ser ***dirigido por casos de uso*** donde los casos de uso definen lo que el usuario desea a partir de la captura de requisitos y la modelación del negocio. Es ***centrado en la arquitectura***, característica que brinda una visión completa del sistema, se describen los procesos del negocio que son más importantes, para comprenderlo, desarrollarlo y producirlo de una forma eficaz. ***Iterativo e Incremental*** donde cada fase se desarrolla en iteraciones, de forma tal que se pueda dividir en pequeños proyectos mejorando su comprensión y desarrollo

Un proyecto realizado siguiendo RUP se divide en cuatro fases:

1. Inicio.
2. Elaboración.
3. Construcción.
4. Transición.

En cada fase se ejecutarán una o varias iteraciones (de tamaño variable según el proyecto), y dentro de cada una de ellas seguirá un modelo de cascada para los flujos de trabajo. RUP define nueve disciplinas a

realizar en cada ciclo del proyecto, donde 6 de ellas son flujos de trabajo básicos y las otras disciplinas de soporte. (4)

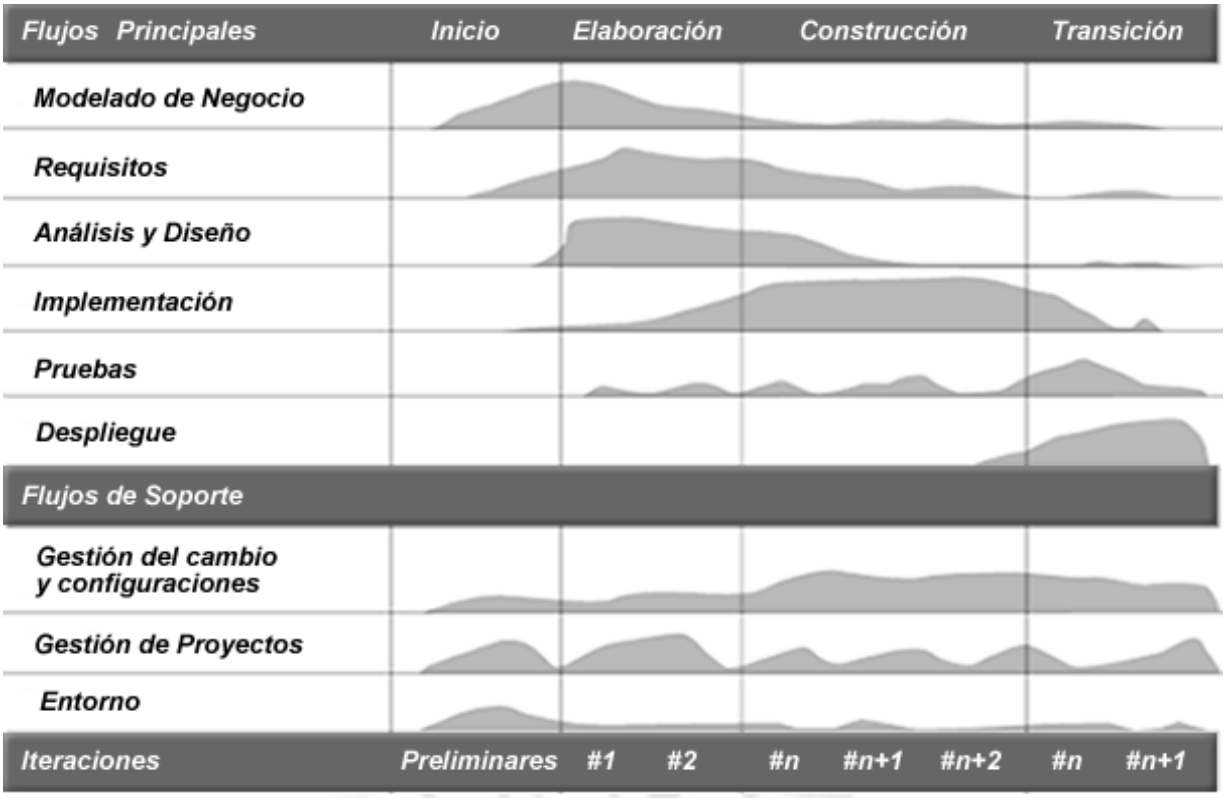


Fig. 1-1 RUP en dos dimensiones

Extreme Programing (XP)

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo.

Se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios.

XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. (5)

Intenta reducir la complejidad del software por medio de un trabajo orientado directamente al objetivo, basado en las relaciones interpersonales y la velocidad de reacción.

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. Este 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.

Características de XP:

• **Pruebas Unitarias**: se basa en las pruebas realizadas a los principales procesos, de tal manera que el adelantarse en algo hacia el futuro, se puedan hacer pruebas de las fallas que pudieran ocurrir. Es como si pudieran obtenerse los posibles errores.

• **Refabricación**: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

• **Programación en pares:** una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa. (6)

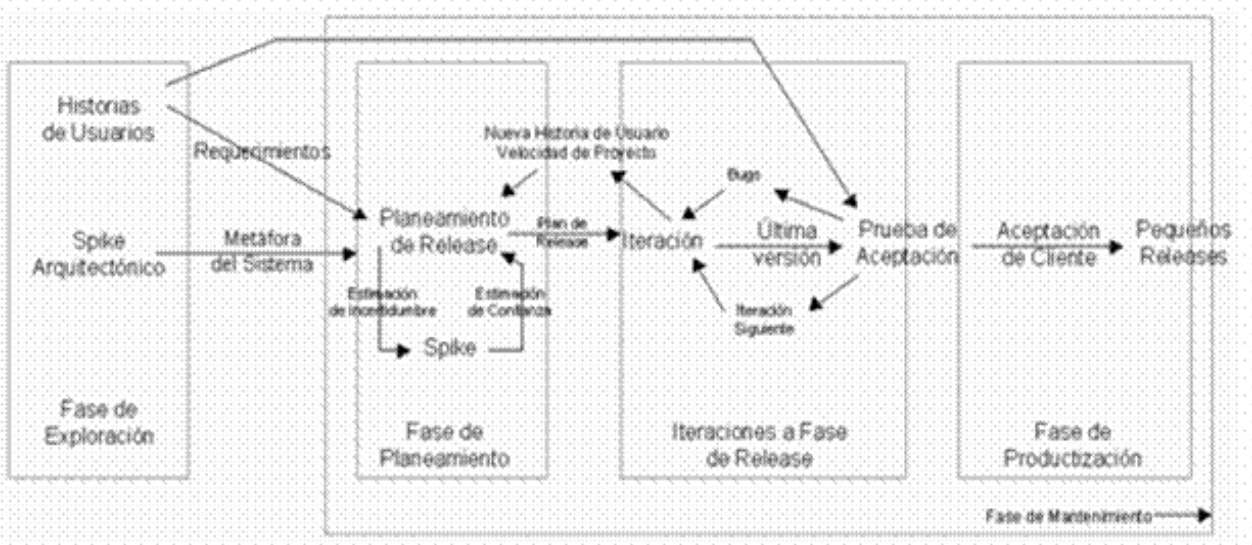


Fig. 1-2 XP ciclo de desarrollo

SCRUM

Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas ***sprints***, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las ***reuniones (scrum)*** a lo largo proyecto. Éstas son las verdaderas protagonistas, especialmente la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración. (5)

SCRUM es un proceso ágil que permite focalizar en la entrega del mayor valor de negocio en el menor plazo. Permite inspeccionar software listo para ser liberado en forma rápida y continua (cada dos a cuatro semanas). El negocio fija las prioridades. Los grupos se **auto-gestionan** para determinar la mejor manera

de entregar las funcionalidades de mayor prioridad. Cada dos a cuatro semanas cualquiera puede ver el software funcionando y decidir liberarlo como está o continuar mejorándolo durante otra iteración.

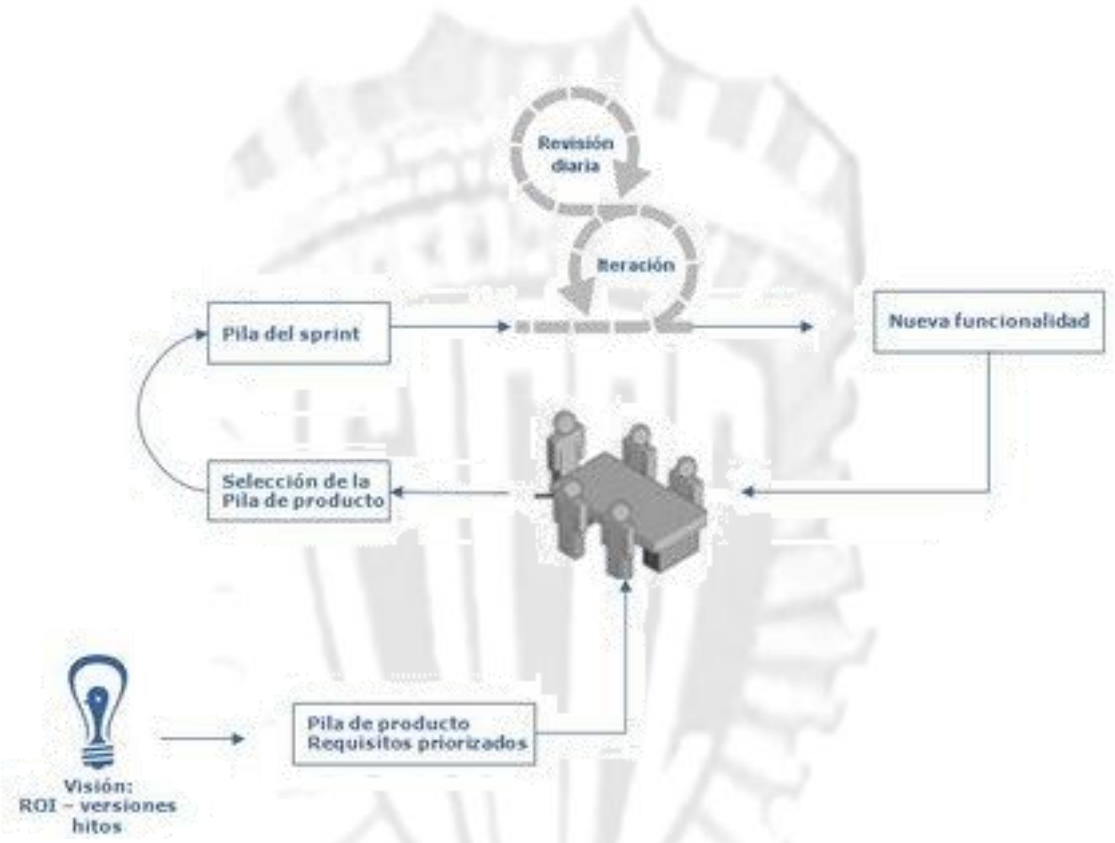


Fig. 1-3 SCRUM ciclo de desarrollo 1

Selección de la metodología a utilizar

Teniendo en cuenta las características de algunas de las metodologías de desarrollo, se pudo constatar de que a pesar de que se centran en la producción de software (orientado a objetos mayormente) y que el proceso se implanta para aumentar la calidad del software producido y la eficiencia de los desarrolladores, existen ciertas diferencias.

La metodología **XP** está destinada a proyectos de corto plazo, se basa en iteraciones pequeñas y está dirigida fundamentalmente a los clientes. **SCRUM** también se basa en iteraciones cortas y es preferible para proyectos pequeños, además de definir un grado de independencia entre los grupos que define. SCRUM únicamente indica cómo conseguir que todos trabajen con el mismo objetivo, a corto plazo y deja bastante visible como avanza el proyecto día a día. **RUP** está pensado para proyectos y equipos grandes, en cuanto a tamaño y duración.

Por las características expuestas anteriormente y por las peculiaridades presentadas por el proyecto, como son, que el cliente se encuentra lejos del lugar de desarrollo y hay una gran dependencia entre sus grupos de trabajo, se ha decidido que la metodología sea **RUP,** la cual basa su trabajo fundamentalmente en la documentación del software y expone un conjunto de actividades que están orientadas a visualizar, especificar, construir, documentar y comunicar los artefactos necesarios para el desarrollo de un software de calidad, presentando una exhaustiva definición de artefactos para ello. Además utiliza el lenguaje unificado de modelado (UML), cuya utilización de diagramas y gráficos brindan una mejor perspectiva de lo que se quiere.

Lenguaje de modelado UML

El Lenguaje Unificado de Modelado (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. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos.

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Un sistema se modela como una colección de objetos discretos que interactúan para realizar un trabajo que finalmente beneficia a un usuario externo. La estructura estática define los tipos de objetos importantes para un sistema y para su implementación, así como las relaciones entre ellos. El comportamiento dinámico define la historia de los objetos en el tiempo y la comunicación entre objetos para cumplir sus objetivos. El modelar un sistema desde varios puntos de vista, separados pero relacionados, permite entenderlo para diferentes propósitos. (7)

1.4 *Tendencias y tecnologías actuales. Selección de las herramientas de desarrollo.*

Para dar solución a la problemática planteada se propone un sistema sobre tecnología Web por las características de la propia organización del CICPC. A continuación se exponen algunas de las ventajas de esta tecnología, que avalan la propuesta:

- Permite tener clientes independientes de tecnología y las estaciones de trabajo.
- Sólo requieren tener un navegador Web con el cual se accede al sistema.
- Se reducen las necesidades de recursos de las estaciones de trabajo.
- El capital humano para el mantenimiento del sistema se reduce pues no es necesaria la instalación ni configuración individualizada.
- Facilita implementar mecanismos de seguridad como la autentificación IP y el monitoreo de los accesos.
- La administración es de forma centralizada.

- Permite tener la información centralizada, lo cual elimina la necesidad de hacer replicas periódicas y mejor resguardo de la información con copias de respaldo de la información.
- Mediante sistemas informáticos sobre la Web se obtienen resultados superiores en cuanto al trabajo con información, las intranets son una vía barata y efectiva de proveer información a toda la organización para la cual se implementa.

Principales lenguajes de programación investigados

Teniendo en cuenta las características del sistema que se desea implementar, que tiene que ser un sistema Web, se ha decidido adoptar una arquitectura cliente-servidor.

Una arquitectura cliente-servidor se caracteriza por: El cliente y el servidor pueden actuar como una sola entidad y también como entidades separadas, realizando actividades o tareas independientes. El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa. El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo. Los cambios en el servidor implican pocos o ningún cambio en el cliente. Las funciones de cliente y servidor pueden estar en plataformas separadas o en la misma plataforma. Un servidor da servicios múltiples de forma concurrente. (8)

Un lenguaje de programación es una técnica estándar de comunicación que permite expresar las instrucciones que han de ser ejecutadas en una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen un lenguaje informático. (9)

Hoy en día existen un gran número de lenguajes de programación que permiten desarrollar una aplicación con las características planteadas anteriormente, los mismos pueden dividirse en dos grandes grupos: lenguajes del lado del cliente y lenguajes del lado del servidor, dentro de este último grupo pueden mencionarse: PHP Hypertext Pre-processor (PHP), Active Server Page (ASP), Java, entre otros.

***PHP**:* Es un lenguaje de programación usado generalmente para la creación de contenido para sitios Web. PHP es un acrónimo recurrente que significa "PHP Hypertext Pre-processor" (inicialmente PHP Tools, o,

Personal Home Page Tools), y se trata de un lenguaje interpretado usado para la creación de aplicaciones para servidores, o creación de contenido dinámico para sitios Web.

El fácil uso y la similitud con los lenguajes más comunes de programación, como C++, permiten, a la mayoría de los programadores experimentados, crear aplicaciones complejas con una curva de aprendizaje muy suave. También les permite involucrarse con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones y prácticas.

Su interpretación y ejecución se da en el servidor Web, en el cual se encuentra almacenado el script, y el cliente sólo recibe el resultado de la ejecución. Cuando el cliente hace una petición al servidor para que le envíe una página Web, generada por un script PHP, el servidor ejecuta el intérprete de PHP, el cual procesa el script solicitado que generará el contenido de manera dinámica, pudiendo modificar el contenido a enviar, y regresa el resultado al servidor, el cual se encarga de regresárselo al cliente.

Desventajas de PHP

- No posee adecuado manejo de internacionalización, Unicode.
- Por su diseño dinámico no puede ser compilado y es muy difícil de optimizar.
- Por sus características, promueve la creación de código desordenado y complejo de mantener.
- Está diseñado especialmente para un modo de hacer aplicaciones web que es ampliamente considerado problemático y obsoleto (mezclar el código con la creación de la página web).

Las dos últimas desventajas aquí mencionadas, pueden no serlo, si el programador es disciplinado y se preocupa de elaborar un diseño previo de lo que quiere hacer antes de ponerse a teclear código**.** (10)

Java*:* es un lenguaje de programación orientado a objetos 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.

Cualquier aplicación que se desarrolle se apoya en un gran número de ***clases*** preexistentes. Algunas de ellas las ha podido hacer el propio usuario, otras pueden ser comerciales, pero siempre hay un número muy importante de clases que forman parte del propio lenguaje (el ***API*** o ***Application Programming Interface*** de ***Java***). ***Java*** incorpora en el propio lenguaje muchos aspectos que en cualquier otro lenguaje son extensiones propiedad de empresas de software o fabricantes de ordenadores (threads (hilos de procesamiento), ejecución remota, componentes, seguridad, acceso a bases de datos, etc.). Por eso muchos expertos opinan que ***Java*** es el lenguaje ideal para aprender la informática moderna, porque incorpora todos estos conceptos de un modo estándar, mucho más sencillo y claro que con las citadas extensiones de otros lenguajes. Esto es consecuencia de haber sido diseñado más recientemente y por un único equipo. (11)

Java se caracteriza principalmente por ser un **lenguaje puro Orientado a Objetos**, lo que le propina una gran reusabilidad; **independencia de la plataforma**, significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware. Es lo que significa ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo, tal como reza el axioma de Java, ‘’write once, run everywhere’’, lo que traducido podríamos decirle: “Escríbelo una vez, ejecútalo en cualquier parte”.

Además posee un **recolector automático de basura (*garbage collector*)** lo cual independiza al programador de tener que administrar memoria solicitada dinámicamente de forma manual.

Selección del lenguaje a emplear en la implementación de la solución

Después de haberse realizado un estudio previo se arriba a la siguiente conclusión: PHP es adecuado para pequeñas aplicaciones web, no posee una abstracción de base de datos estándar, sino bibliotecas especializadas para cada motor (a veces más de una para el mismo motor), además de poseer variables globales que dificultan el mantenimiento de la aplicación. Requiere de un carácter especial para el nombre de las variables, posee menos tipos de datos, además de no ser un lenguaje fuertemente tipado y no poseer el uso de interfaces. En cambio Java es más general y es más adecuado para aplicaciones grandes, posee múltiples hilos de procesamiento.

Por las características anteriormente planteadas puede apreciarse que Java es un lenguaje más robusto que PHP, además que por el tamaño tan grande y la alta seguridad que tiene el sistema en desarrollo se ha decidido el uso de Java como lenguaje de programación.

Entorno de Desarrollo Integrado (IDE)

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Los IDEs pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes.

Eclipse

Eclipse comenzó como un proyecto de IBM Canadá. Fue desarrollado por OTI (Object Technology International) como reemplazo de VisualAge también desarrollado por OTI. En noviembre del 2001, se formó un consorcio para el desarrollo futuro de Eclipse como código abierto. En 2003, la fundación independiente de IBM fue creada. Eclipse 3.0 (2003) seleccionó las especificaciones de la plataforma Open Services Gateway Initiative (OSGI) como la arquitectura de tiempo de ejecución.

Callisto: En 2006 la fundación Eclipse coordinó sus 10 proyectos de código abierto, incluyendo la Plataforma 3.2, para que fueran liberados el mismo día. Esta liberación simultánea fue conocida como la liberación Callisto.

Europa: La versión consecutiva a Callisto es Europa, que corresponde a la versión 3.3 de Eclipse.

En la web oficial de Eclipse (www.eclipse.org), se define como “An IDE for everything and nothing in particular” (un IDE para todo y para nada en particular). Eclipse es, en el fondo, únicamente un armazón (workbench) sobre el que se pueden montar herramientas de desarrollo para cualquier lenguaje, mediante la implementación de los plugins adecuados.

La arquitectura de plugins de Eclipse permite, además de integrar diversos, lenguajes sobre un mismo IDE, introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces, ayuda en línea para librerías, entre otros.

Eclipse es una aplicación multiplataforma que dispone de: un editor de texto, resaltado de sintaxis, compilación en tiempo real, pruebas unitarias con JUnit, control de versiones con CVS, integración con Ant, wizards para la creación de proyectos, clases, test, componentes, refactorización. Asimismo, a través de "plugins" libremente disponibles es posible añadir: Control de versiones con Subversion, vía Subeclipse; Integración con Hibernate, vía Hibernate Tools, con Aptana, para desarrollar con java scripts. (12)

NetBeans

NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. Es un producto libre y gratuito sin restricciones de uso.

El NetBeans es un IDE de código abierto escrito completamente en Java. El NetBeans IDE soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en Ant, control de versiones y refactorización.

Modularidad: todas las funciones del IDE son provistas por módulos. Cada módulo provee una función bien definida, tales como el soporte de Java, edición, o soporte para el sistema de control de versiones. NetBeans contiene todos los módulos necesarios para el desarrollo de aplicaciones Java en una sola descarga, permitiéndole al usuario comenzar a trabajar inmediatamente.

Desde Julio de 2006, NetBeans IDE es licenciado bajo la Common Development and Distribution License (CDDL), una licencia basada en la Mozila Public License (MPL).

En la versión 5.0 ya se cuenta con un debugger donde el establecimiento de los breakpoints (puntos de ruptura) y sus propiedades son accesibles desde el propio editor. Se permite la evaluación de expresiones sobre la marcha (on the fly), así como la invocación de métodos. Se ha incorporado un debugger para Ant. Reconoce los directorios de trabajo de CVS y lista en la ventana de “Versioning” los ficheros que han cambiado (NetBeans). Se ha mejorado sustancialmente la facilidad de incorporación y manejo de

JavaServer Faces (JSF) y Struts; que puede activarse desde el nuevo “New Project Wizard” indicando que se desea crear una aplicación web.

La versión NetBeans IDE 5.5.1, la cual fue lanzada en Mayo de 2007. NetBeans IDE 5.5 extiende las características existentes del Java EE (incluyendo Soporte a Persistencia, EJB 3 y JAX-WS). Adicionalmente, el NetBeans Enterprise Pack soporta el desarrollo de aplicaciones empresariales con Java EE 5, incluyendo herramientas de desarrollo visuales de SOA, herramientas de esquemas XML, orientación a web servicies (for BPEL), y modelado UML. El NetBeans C/C++ Pack soporta proyectos de C/C++.

Ya la versión 6.0 soporta las tecnologías Enterprise JavaBeans (EJB) 3.0, JAX-WS 2.1, Java Server Faces 1.2, Java Server Page 2.1, JavaServer Pages Standard Tag Library (JSTL) 1.1, el editor es más rápido y más inteligente. (13)

Selección del IDE

A pesar de que Netbeans tiene gran parte de su funcionalidad similar con eclipse, le faltan aspectos como: carece de cierta funcionalidad de alto nivel, como las vistas y perspectivas. Además al inicio del proyecto CICPC aun la versión 6.0 no había sido liberada (fue liberada en diciembre del 2007) y la versión 5.0 a pesar de que ya había sido liberada, existía muy poca experiencia sobre dicha herramienta. Por tales motivos fue que la decisión estuvo a favor de eclipse como IDE de desarrollo.

Herramientas CASE de Modelado con UML

Desde la publicación oficial del UML a fines de 1997, la cantidad de las herramientas comerciales para el modelado con UML se incrementó dramáticamente. Esto provee de un mayor número de alternativas y exige realizar una investigación más profunda para seleccionar la herramienta de modelado UML que responda mejor a los requisitos del negocio y desarrollo de software de aplicación que permite lograr el mejor retorno de la inversión.

CASE es una sigla, que corresponde a las iniciales de: **C**omputer **A**ided **S**oftware **E**ngineering; y en su traducción al Español significa Ingeniería de Software Asistida por Computación.

A medida que los sistemas que hoy se construyen se tornan más complejos, las herramientas de modelado con UML ofrecen muchos beneficios para todos los involucrados en un proyecto, por ejemplo, administrador del proyecto, analistas, arquitectos, desarrolladores y otros. Las herramientas CASE de modelado con UML permiten aplicar la metodología de análisis y diseño orientados a objetos y abstraerse del código fuente, en un nivel donde la arquitectura y el diseño se tornan más obvios y más fáciles de entender y modificar. Cuanto más grande es un proyecto, es más importante utilizar una herramienta CASE. Por otro lado, al usar las herramientas CASE:

- Los Analistas de Negocio/ Sistemas pueden capturar los requisitos del negocio/sistema con un modelo de casos de uso.
- Los Diseñadores/Arquitectos pueden producir el modelo de diseño para articular la interacción entre los objetos o los subsistemas de la misma o de diferentes capas (los diagramas UML típicos que se crean son los de clases y los de interacción).
- Los Desarrolladores pueden transformar rápidamente los modelos en una aplicación funcionando, y buscar un subconjunto de clases y métodos y asimilar el entendimiento de cómo lograr interfaces con ellos.

Visual Paradigm: es una herramienta CASE que usa UML como lenguaje de modelado, se integra con herramientas como Eclipse, Hibernate, Subversión, todas aplicables para el desarrollo, permite la generación automática de reportes en formato pdf y html, el reconocimiento de artefactos de ingeniería a partir de reconocimiento de textos formales o informales, implementa una actualización automática del modelo de diseño y código permitiendo mantener la documentación de ambos modelos actualizadas con los cambios que ocurran en ambos sentidos, optimizando la descripción textual de elementos de código a partir de la descripción visual. Es capaz de importar y exportar elementos de otras herramientas CASE como Rational Rose y presenta una interfaz de uso intuitiva y con increíbles facilidades a la hora de modelar los diagramas que soportan la Ingeniería de Requerimientos. (14)

Rational Rose: es la herramienta CASE desarrollada por los creadores de UML (Booch, Rumbaugh y Jacobson), que cubre todo el ciclo de vida de un proyecto: Concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables. El navegador UML de Rational Rose permite establecer una trazabilidad real entre el modelo (análisis y diseño) y el código ejecutable. Facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (vista de Casos de Uso, vista Lógica, vista de Componentes y vista de Despliegue), pero utilizan un lenguaje común para comprender y comunicar la estructura y la funcionalidad del sistema en construcción. (15)

Por las características que posee, su mejor generación de código, además de su integración con Eclipse se ha escogido Visual Paradigm como herramienta CASE de desarrollo.

Frameworks soportados por Java

El concepto framework se emplea en muchos ámbitos del desarrollo de sistemas software, no solo en el ámbito de aplicaciones Web. Se pueden encontrar frameworks para el desarrollo de aplicaciones médicas, de visión por computador, para el desarrollo de juegos, y para cualquier otro ámbito.

En general, el término framework, se refiere a una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework se puede considerar como una aplicación genérica incompleta y configurable a la que pueden añadírsele las últimas piezas para construir una aplicación concreta. (16)

Los objetivos principales que persigue un framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones.

Java soporta gran números de frameworks especializándose cada uno de ellos en una parte específica del sistema, ya sea la capa de presentación, la capa de la lógica de negocio o la capa de acceso a datos.

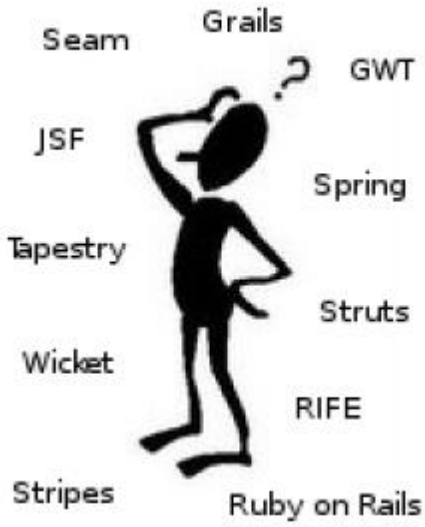


Fig. 1-4 La gran decisión de escoger los frameworks para desarrollar

Capa de Presentación

Se pueden mencionar varias tecnologías que facilitan el desarrollo de la capa de presentación de un sistema: servlet, JSP, Spring MVC, JSF o Struts. Se ha decidido enfocar el estudio hacia las dos últimas.

Struts: es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC bajo la plataforma J2EE (Java 2, Enterprise Edition). Struts se desarrollaba como parte del proyecto Jakarta de la Apache Software Foundation, pero actualmente es un proyecto independiente conocido como Apache Struts.

Cuando se programan aplicaciones Web con el patrón MVC, siempre surge la duda de usar un solo controlador o varios controladores, pues si se considera mejor usar un solo controlador para tener toda la lógica en un mismo lugar, surge un inconveniente, ya que el controlador se convierte en lo que se conoce como "fat controller", es decir un controlador de peticiones, Struts surge como la solución a este problema, ya que implementa un solo controlador (ActionServlet) que evalúa las peticiones del usuario mediante un archivo configurable (struts-config.xml).

***JSF Framework**:* estándar para aplicaciones web basadas en Java, facilita la construcción de aplicaciones siguiendo el patrón MVC, modelo de componentes orientado a objetos, conversión de tipos, separación de responsabilidades, desarrolladores de componentes, desarrolladores de lógica de aplicación, montadores de páginas, poderoso sistema de navegación declarativa, uso de simples clases java como controladores, fácil incorporación de potencialidades Ajax, posee un conjunto prefabricado de componentes de interfaz de usuario, modelo de programación orientado a eventos. (17)

Patrón Modelo- Vista-Controlador (MVC): es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres tipos de componentes distintos. (18)

Framework Ajax4JSF: Ajax4jsf es una librería open source que se integra totalmente en la arquitectura de JSF y extiende la funcionalidad de sus etiquetas dotándolas con tecnología Ajax de forma limpia y sin añadir código Java Script. Mediante este framework podemos variar el ciclo de vida de una petición JSF, recargar determinados componentes de la página sin necesidad de recargarla por completo, realizar peticiones al servidor automáticas, control de cualquier evento de usuario, etc. En definitiva Ajax4jsf permite dotar a nuestra aplicación JSF de contenido mucho más profesional con muy poco esfuerzo. (19)

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, mejorando algunas de sus deficiencias. Además, algunos Entornos de Desarrollo Integrado como Eclipse y Netbeans brindan facilidades que soportan el trabajo con el mismo. A diferencia de Struts, JSF es parte de Java EE, todos los servidores de aplicaciones, por tanto, incluyen JSF. (20)

Capa de lógica de negocio

Spring: Framework de código abierto, provee servicios empresariales en POJOs, es el más popular y el más ambicioso de todos los framework de peso ligero, interviene en todas las capas arquitectónicas de una aplicación J2EE, brinda soporte a varios frameworks de presentación, entre ellos Java Server Faces (JSF), brinda soporte a Hibernate.

La Programación Orientada a Aspectos (AOP) complementa la Programación Orientada Objetos (POO) proponiendo otra manera de pensar sobre la estructura de un programa. Mientras que la POO

descompone las aplicaciones en una jerarquía de objetos, la AOP descompone los programas en aspectos (aspects) o preocupaciones (concerns). Esto permite la modularización de preocupaciones justo como el manejo de transacciones que pueden ser aplicados a múltiples objetos. (Tales preocupaciones a menudo se nombran preocupaciones tranversales y en inglés crosscutting concerns.)

Uno de los componentes clave de Spring es el framework AOP. Los contenedores IoC (Inversion of Control) de Spring (BeanFactory y ApplicationContext) no dependen de AOP, lo que implica que no necesitas usar AOP si no quieres, AOP complementa el IoC de Spring para proporcionar una solución muy capaz de *middleware*.

AOP se usa en Spring para proporcionar servicios empresariales de manera declarativa, especialmente como reemplazo para los servicios declarativos de EJB. El más importante de dichos servicios es el manejo declarativo de transacciones, que está construido sobre la abstracción de transacciones de Spring, para permitir que los usuarios implementen aspectos personalizados, complementando el uso de POO con AOP.

Framework Acegi Security System: framework que cubre todas las facetas de la seguridad de una aplicación empresarial, concebido para integrarse con Spring, fácil de extender, no necesita de contenedores para su uso, es Open Source, suficiente bibliografía. (21)

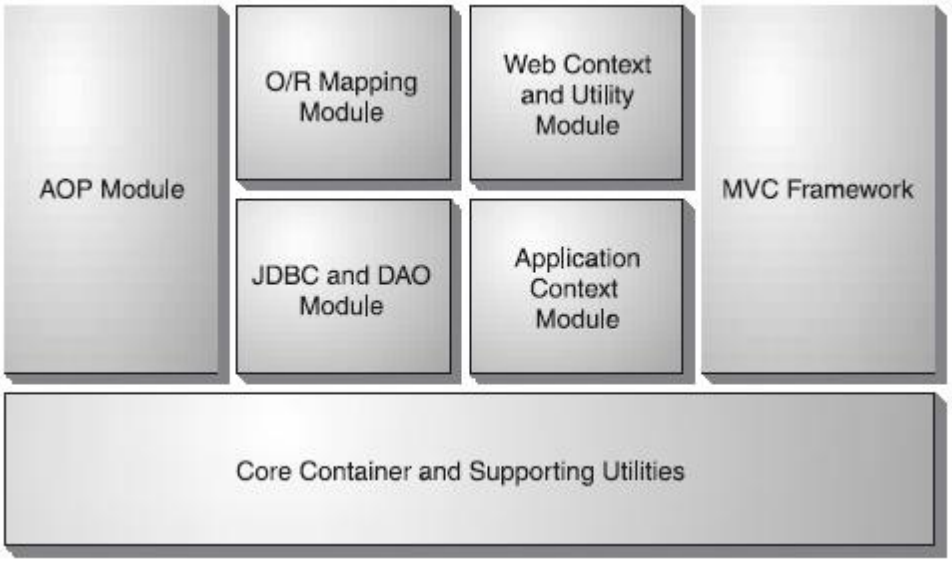


Fig. 1-5 Módulos de Spring

JBoss Seam Framework: Seam es un potente framework de Java que surgió el 11de junio del 2006, unifica la integración de diversas tecnologías, tales como: Java Script asíncrono y XML (AJAX), JSF, Enterprise Java Beans (EJB 3.0), Web Services y administración de procesos de negocio (BPM). Surgió con el objetivo de eliminar la complejidad de la arquitectura y del API, habilitando a los desarrolladores para que puedan desarrollar complejos sistemas con simples anotaciones en los POJOs y poco código XML. Soporta Rich-Faces y ICE–Faces, dos soluciones open source de JSF basado en AJAX. Se inclina más por las anotaciones que por el XML; con las anotaciones, EJB 3.0 se encarga de brindarle la información necesaria al contexto, no tanto JSF que aún depende del XML para definir sus reglas de navegación. (22)

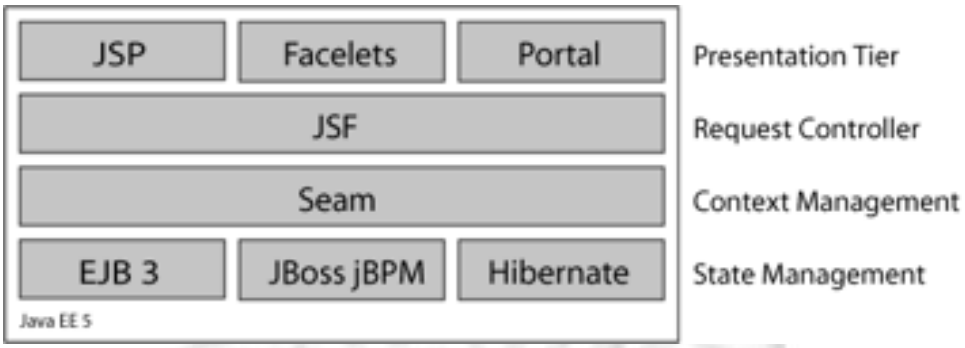


Fig.1-6 Estructura de Seam

Capa de Acceso a datos

La forma que existe para la persistencia en una aplicación en Java es: abrir una conexión JDBC, crear una sentencia SQL donde se copian los datos del objeto y ejecutarla, esto se convierte un proceso muy tedioso y repetitivo, además de muy propenso a errores.

La persistencia en una aplicación orientada a objetos, es un tema de sumo cuidado ya que se refiere a aquellos objetos que van a persistir en una base de datos o en disco duro a través del tiempo. Una aplicación con un modelo de dominio no interactúa directamente con las filas y las columnas, en las que está estructurada la información en la base de datos, en su lugar lo hará con las entidades definidas en su modelo de dominio, aquí es donde entran a jugar un papel muy importante los Object/Relational Mapping (ORM).

La idea de los distintos ORM, es definir un mapeo de los objetos con los que se trabaja en memoria a una base de datos relacional en la que se almacena toda la información del negocio.

Hibernate: Es un ORM no intrusivo, un entorno de trabajo que tiene como objetivo facilitar la persistencia de objetos Java en bases de datos relacionales y al mismo tiempo la consulta de estas bases de datos para obtener objetos.Es una capa de persistencia objeto/relacional y un generador de sentencias SQL. Permite diseñar objetos persistentes que podrán incluir polimorfismos, relaciones, colecciones, y un gran número de tipos de datos.

De una manera muy rápida y optimizada se puede generar base de datos en cualquiera de los entornos soportados: Oracle, DB2, MySql, y otros. Lo más importante de todo, es ***open source***. Uno de los posibles procesos de desarrollo consiste en, una vez que se tenga el diseño de datos realizado, mapear este a ficheros XML siguiendo la DTD de mapeo de Hibernate. Desde estos se puede generar el código de los objetos persistentes en clases Java y también crear base de datos independientemente del entorno escogido. (23)

Hibernate es un ambicioso proyecto que apunta a ser una solución completa al problema de la persistencia. Es una solución no intrusiva, que no te obliga a seguir determinadas reglas o patrones de diseño cuando se desarrolla la capa de lógica de negocio o las clases del dominio. Las clases persistentes no requieren implementar la interfaz serializable, aunque si de un constructor sin parámetros. (24)

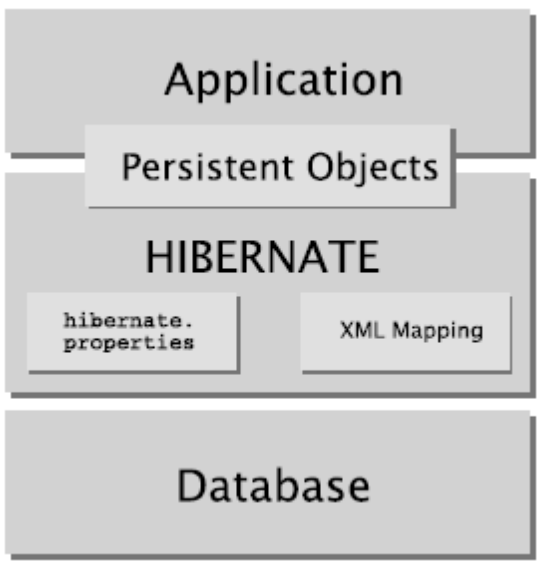


Fig. 1-7 Vista de alto nivel de la arquitectura de Hibernate.

***iBATIS*:** es un framework de código abierto basado en capas, desarrollado por Apache Software Foundation, que se ocupa de la capa de Persistencia (se sitúa entre la capa de Negocio y la capa de la Base de Datos). Puede ser implementado en Java y .NET.

iBATIS SQL MAPS asocia objetos de modelo (JavaBeans) con sentencias SQL o procedimientos almacenados mediante ficheros descriptores XML, simplificando la utilización de bases de datos, o sea, basa su funcionamiento en el mapeo de sentencias SQL que se incluyen en ficheros XML. Además cada objeto de modelo, que representa al objeto en la aplicación, se relaciona con un fichero del tipo sqlMap.xml, que contiene sus sentencias SQL. Por ejemplo, un objeto Java Usuario con un objeto XML usuario.xml.

Por tal motivo para desarrollar en este framework el desarrollador necesita mucho conocimiento de SQL. Por otra parte, permite la optimización de las consultas, ya sea con lenguaje estándar o con SQL propietario del motor de base de datos utilizado, siempre se sabe lo que se está ejecutando en la base de datos (25)

Para cada sentencia básica Insert, select, update o delete se necesita escribir la sentencia SQL en el fichero de configuración y después en el código llamar a dicha sentencia, por el nombre que tenia especificado. [1]

Tras el análisis anteriormente descrito se ha decidido desarrollar una aplicación web implementada en Java y haciendo uso de sus frameworks de desarrollo tales como JSF, Spring y Hibernate. El uso de eclipse como IDE de desarrollo y Visual Paradigm como herramienta CASE. Todo esto con RUP como metodología de desarrollo. Haciendo uso no solo del patrón MVC sino también del patrón N capas de 3 niveles, que se muestra a continuación.

[1] Ver ejemplo en los anexos

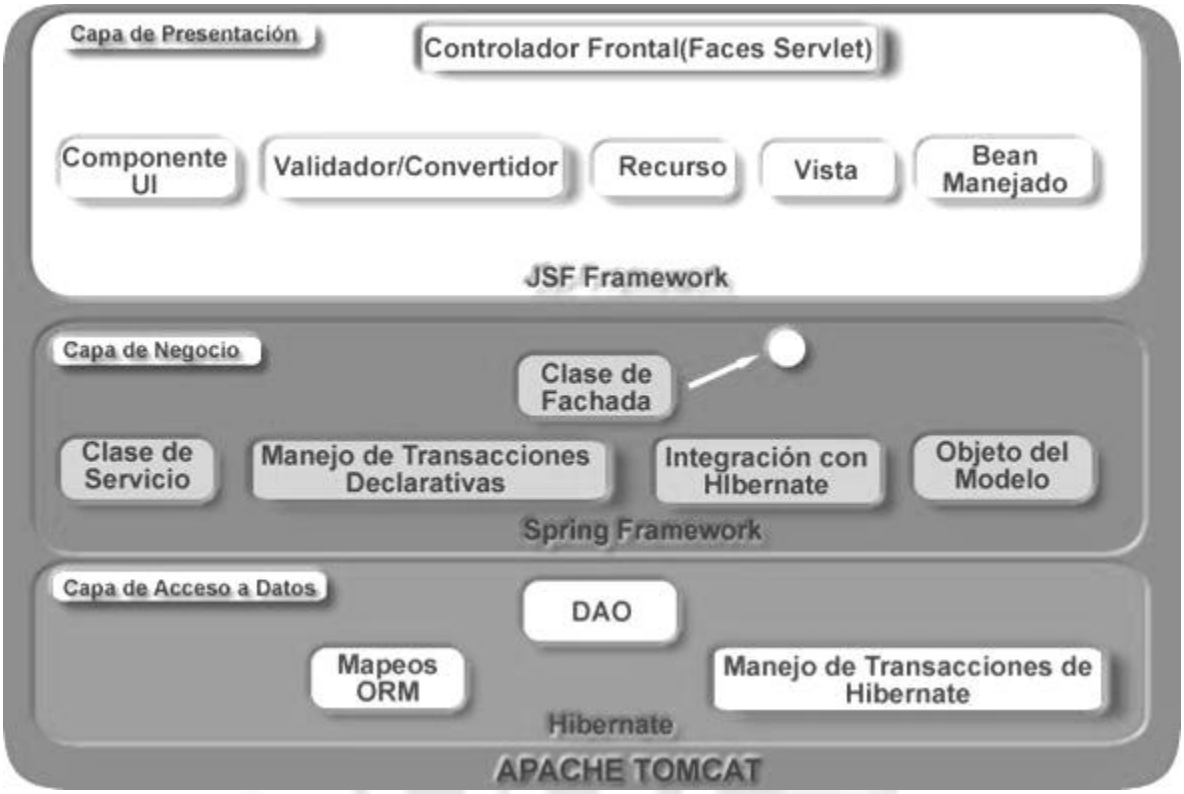


Fig. 1-8 Patrón n capas de 3 niveles de la aplicación

1.5 *Conclusiones*

Teniendo en cuenta el estudio realizado previamente sobre las principales tendencias que existen actualmente acerca de las tecnologías y herramientas más usadas en el campo de la informática, se toma como decisión desarrollar una aplicación web, sobre el lenguaje Java, con la integración de los frameworks Java Server Faces (JSF), Spring y Hibernate, ganando en velocidad de desarrollo y aplicando el patrón n capas con tres niveles, lo que facilita el desarrollo del producto disminuyendo el acoplamiento. Para lograr tal resultado se propone el uso de las herramientas Eclipse y Visual Paradigm por las facilidades que estas brindan. Todo este proceso será controlado y orientado por la metodología RUP, la cual más que una dogma constituye una guía de cómo se debe desarrollar.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA PROPUESTA DE SISTEMA

2.1 Introducción

En el presente capítulo se tratan los temas relacionados con el análisis y diseño de la propuesta del sistema, para ello se comenzará con una breve referencia de los principales procesos que ocurren dentro del departamento de Aprehensión, así como también la descripción de los casos de uso del módulo, los cuales fueron el resultado de la Ingeniería de Requerimientos aplicada a dicho departamento que se tuvo como entrada para el desarrollo de la propuesta. Además se explicarán temas relacionados a cuestiones propias del sistema, que se adaptaron a la arquitectura definida por los arquitectos del proyecto.

2.2 ¿Qué son el análisis y el diseño?

En la fase inicial, el análisis y el diseño se centran en establecer si el sistema que se ha concebido es factible, y en evaluar las tecnologías potenciales para la solución. Si se percibe que supone algún riesgo para el desarrollo (a causa, por ejemplo, de que el dominio se entiende bien, el sistema no es nuevo), entonces esta actividad puede omitirse.

La fase inicial de elaboración se centra en la creación de una arquitectura inicial para el sistema (Actividad: Definir una arquitectura candidata) para proporcionar un punto de inicio para el trabajo de análisis principal. Si la arquitectura ya existe (porque se ha producido en anteriores iteraciones, en anteriores proyectos o se ha obtenido de una infraestructura de aplicación, la atención del trabajo cambia y pasa a ser el perfeccionamiento de la arquitectura (la Actividad: Perfeccionar la arquitectura. Se crea un

conjunto inicial de elementos que proporcionan el comportamiento adecuado (la Actividad: Analizar el comportamiento).

Una vez que se han identificado los elementos iniciales, se perfeccionan aún más. La Actividad: Diseñar componentes produce un conjunto de componentes que proporcionan el comportamiento adecuado para satisfacer los requisitos del sistema. Si el sistema incluye una base de datos, la Actividad: Diseñar la base de datos se da en paralelo.

Análisis

El objetivo de este flujo de trabajo es traducir los requisitos a una especificación que describe cómo implementar el sistema. El análisis consiste en obtener una visión del sistema que se preocupa de ver QUÉ hace, de modo que sólo se interesa por los requisitos funcionales.

En el análisis se presentan los siguientes estereotipos de clases:

Clase de interfaz: Modela la interfaz del sistema, y manejan la comunicación entre el entorno y el interior del mismo. Durante el diseño, estas clases son refinadas para tomar en consideración los mecanismos de interfaz seleccionados o implementados, además de facilitar la comunicación con otros sistemas, entre otros.

Fig. 2-1 Clase interfaz

Clases de entidad: Representan la información manejada en el caso de uso, además de que modelan información y comportamiento asociado que generalmente es de larga duración. Reflejan entidades del mundo real, que resultan necesarias para realizar tareas internas del sistema.

Fig. 2-2 Clase entidad

Clases de controladora: Coordinan los eventos necesarios para la realización o especificación del caso de uso, con otras palabras, son las que ejecutan el caso de uso. Usualmente son dependientes de la aplicación, además de tener un control sobre todas las acciones a realizar.

Fig. 2-3 Clase controladora

Diseño

El diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva CÓMO cumple el sistema sus objetivos. El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades. De hecho, cuando la precisión del diseño es muy grande, la implementación puede ser hecha por un generador automático de código. En el diseño además se tienen en cuenta diferentes patrones de diseño que no son más que diferentes soluciones que ya existen a los principales problemas que han surgido en el desarrollo de software. (18)

2.3 Previo al análisis.

La metodología de desarrollo seleccionada para desarrollar el sistema (RUP), define 6 disciplinas básicas en los ciclos de desarrollo para producir software, el modelamiento del negocio y requerimientos son las disciplinas que anteceden al análisis.

Procesos elementales del negocio

Entre los principales procesos que ocurren en el Departamento de Aprehensión se describen: el proceso de aprehensión que es cuando una persona es detenida y llevada a dicho departamento, donde se le llena la planilla de ingreso, en la cual se toman los datos referentes al delito que estaba cometiendo o la orden de aprehensión que estaba emitida contra dicha persona. Sus datos personales y todo, es archivado en la carpeta del detenido asociada. Después que la persona se encuentra ubicada en el departamento se le realizan un conjunto de acciones por parte de los funcionarios, de cada una de estas, quedan como constancia las actas de detenido, las cuales son archivadas en la carpeta del detenido asociada. Si la persona fuese inocente es emitida por el tribunal una Boleta de Excarcelación, la cual es enviada al departamento de aprehensión donde se encuentra detenida la persona. En caso contrario es emitida una Boleta de Traslado al Retén, en la cual se encuentra constancia del Retén al cual será trasladada la persona y bajo qué Centro de Atención al Recluso estará.

Requerimientos funcionales

RF 1 Gestionar Detenido

1.1 Ver datos de un detenido.

1.2 Ingresar un detenido.

1.3 Modificar datos de un detenido.

RF 2 Crear una nueva Carpeta de Detenido.

RF 3 Actualizar la entidad Persona.

3.1 Cambiar el estado de la entidad Persona.

RF 4 Imprimir o exportar a PDF los datos de la Planilla de Detención.

RF 5 Mantener informado al usuario del resultado de las operaciones.

RF 6 Gestionar Acta de Detenido

6.1 Incluir un Acta de Detenido.

6.2 Ver los datos de un Acta de Detenido.

RF 7 Asociar un Acta de Detenido a una Carpeta de Detenido.

RF 8 Imprimir o exportar a PDF un Acta de Detenido.

RF 9 Gestionar Boleta de Excarcelación

9.1 Ver datos de la Boleta de Excarcelación

9.2 Incluir Boleta de Excarcelación.

RF 10 Imprimir o exportar a PDF Boleta de Excarcelación.

RF 11 Asociar una boleta de Excarcelación a una Carpeta de Detenido

RF 12 Gestionar Boleta de Traslado al Retén.

12.1 Incluir una Boleta de Traslado al Retén.

12.2 Ver los datos de una Boleta de Traslado al Retén.

RF 13 Asociar una Boleta de Traslado al Retén a una Carpeta de Detenido.

RF 14 Imprimir o exportar a PDF Boleta de Traslado al Retén.

RF 15 Consultar un listado de Carpetas de Detenidos.

15.1 Buscar Carpetas de Detenidos dados criterios.

15.2 Mostrar un listado de Carpetas de Detenidos ordenadas por un criterio.

15.3 Ordenar un listado de Carpetas de Detenidos dado un criterio.

RF 16 Imprimir o Exportar a PDF un listado de Carpetas de Detenido.

RF 17 Consultar diligencias sobre detenido.

17.1 Buscar las diligencias realizadas sobre un detenido dado criterios.

17.2 Mostrar un listado de las diligencias sobre un detenido ordenada por un criterio.

RF 18 Imprimir o Exportar a PDF las diligencias sobre un detenido.

RF 19 Consultar Órdenes de Aprehensión.

19. 1 Buscar Órdenes de Aprehensión dado criterios

19.2 Mostrar un listado de Órdenes de Aprehensión ordenadas por un criterio

19.3 Ordenar un listado de Órdenes de Aprehensión dado un criterio

RF 20 Imprimir o Exportar a PDF Órdenes de Aprehensión.

RF 21 Ingresar el detenido asociado a la Orden de Aprehensión.

Descripción de los Casos de Uso

La información que se presenta a continuación representa una breve síntesis de los casos de uso que se tomaron como entrada para la realización del presente trabajo de diploma, las descripciones expandidas se podrán encontrar en los anexos de este documento.[2]

Caso de uso	
	Gestionar Detenido
Propósito	Incluir, ver o modificar los datos de un Detenido.

[2] Ver la descripción detallada en los Anexos

Actores: Gestor de Aprehensión
Resumen: El caso de uso se inicia cuando el actor selecciona la opción que le permite realizar una acción sobre un Detenido en el sistema. El actor puede incluir, ver o modificar los datos de un Detenido. En caso de que seleccione la opción de incluir un nuevo Detenido, el sistema dará la posibilidad de introducir los datos de la Planilla de Ingreso del Detenido, y se crea la Carpeta de Detenido correspondiente, la persona puede o no tener alguna orden de aprehensión. Si el actor elige la opción de ver los datos de un Detenido, el sistema mostrará los datos del Detenido en cuestión. Si el actor elige la opción de modificar los datos de un Detenido el sistema mostrará los datos que pueden ser editables, y una vez realizados los cambios, guardará las modificaciones. El sistema permite imprimir y exportar a PDF los datos del Detenido, dando fin al caso de uso.

Caso de uso	
	Gestionar constancia de acción realizada sobre detenido
Propósito	Incluir, ver o modificar un Acta de Detenido en la Carpeta de Detenido.
Actores: Gestor de Aprehensión	
Resumen: El caso de uso inicia cuando el actor desea realizar alguna acción sobre un Acta de Detenido, si desea incluir una nueva Acta de Detenido, el sistema muestra opciones para la inclusión de la misma, el actor incluye los datos y el sistema genera automáticamente la vista del Acta de Detenido. En caso que desee ver un Acta de Detenido, el sistema muestra todos los campos del Acta de Detenido seleccionada. En caso que se desee modificar un Acta de Detenido, se muestra la información actual y se permite realizar modificaciones sobre sus datos. El caso de uso termina. El sistema permite imprimir y exportar a PDF los datos del Detenido, dando fin al caso de uso.	

Caso de uso	
	Gestionar Boleta de Excarcelación
Propósito	Incluir o ver Boleta de Excarcelación.
Actores: Gestor de Aprehensión	
Resumen: El caso de uso se inicia cuando el Investigador selecciona la opción que le permite realizar una acción sobre una Boleta de Excarcelación. El actor puede incluir y ver una Boleta de Excarcelación. En caso de que seleccione la opción de incluir una Boleta de Excarcelación, el sistema dará la posibilidad de insertar los datos que se necesitan para llenar esta plantilla. Si el actor elige la opción de ver Boleta de Excarcelación, el sistema mostrará el contenido de la Boleta de Excarcelación en cuestión. El sistema permite ver una vista previa de la Boleta de Excarcelación, con posibilidad de imprimir y exportar a PDF terminando así el caso de uso.	

Caso de uso	
	Gestionar Boleta de Traslado al Retén
Propósito	Incluir o ver Boleta de Excarcelación Traslado al Retén
Actores: Gestor de Aprehensión	
Resumen: El caso de uso inicia cuando el actor desea realizar alguna acción sobre una Boleta de Traslado al Retén, si desea incluir una nueva Boleta, el sistema muestra opciones para la inclusión de la misma, el actor incluye los datos y el sistema genera automáticamente la vista de la Boleta de Traslado al Retén. En caso que desee ver una Boleta de Traslado al Retén, el sistema muestra los campos de la Boleta seleccionada. El caso de uso termina.	

Caso de uso	
	Consultar Carpetas de Detenidos
Propósito	Buscar y listar de manera ordenada un resumen de los datos de las Carpetas de Detenidos coincidentes con uno o varios criterios de búsqueda.
Actores: Consultor de Aprehensión	
Resumen: El caso de uso se inicia cuando el actor accede a la opción que le permite consultar por diferentes criterios datos relacionados con las carpetas de detenido del Despacho al que pertenece. El sistema solicita criterios de búsqueda y brinda la información que se corresponda con los criterios introducidos por el actor. El sistema permite ordenar el listado mostrado según el criterio que seleccione el actor (fecha de creación, número de Carpeta de Detenido, entre otros.) El sistema permite seleccionar una Carpeta de Detenido para ver sus detalles. Si el actor elige la opción de consultar una Carpeta de Detenido, el sistema muestra los datos de esta. El sistema permita imprimir o exportar a PDF el listado seleccionado y el caso de uso termina.	

Caso de uso	
	Consultar Diligencias sobre Detenido
Propósito	Buscar y listar de manera ordenada un resumen de los datos de las diligencias sobre el detenido coincidente con uno o varios criterios de búsqueda.
Actores: Consultor de Aprehensión	

Resumen: El caso de uso se inicia cuando el actor selecciona la opción de consultar las diligencias sobre un detenido. El sistema brinda la posibilidad de introducir los datos de búsqueda y el período de tiempo. El actor introduce los datos, el sistema busca las diligencias realizadas sobre un detenido en un determinado período de tiempo y muestra los datos de la consulta. El sistema permite imprimir o exportar a PDF el resultado de la consulta y el caso de uso termina.

Caso de uso	
	Consultar Órdenes de Aprehensión
Propósito	Buscar y listar de manera ordenada un resumen de los datos de las Órdenes de Aprehensión coincidentes con uno o varios criterios de búsqueda.
Actores: Consultor de Aprehensión	
Resumen: El caso de uso se inicia cuando el actor selecciona la opción que le permite consultar las Órdenes de Aprehensión. El actor realiza una búsqueda de las Órdenes de Aprehensión del despacho y el sistema muestra un listado de estas. El sistema permite listar todas las Órdenes de Aprehensión sin realizar la búsqueda. El actor tiene la opción de ver los detalles de un elemento en específico de los mostrados en el listado, o puede asociar una Orden de Aprehensión a un detenido que se vaya a ingresar en el despacho, dando fin al caso de uso. El sistema permite imprimir o exportar a PDF el resultado de la consulta y el caso de uso termina.	

Diseño de las interfaces de usuario

Otros de los aspectos a tener en cuenta para el desarrollo de la aplicación son los diseños de las interfaces de usuario que han sido definidos y elaborados por los diseñadores de interfaz de usuario en correspondencia con el Arquitecto de Información del proyecto, a continuación se presentarán algunos de ellos, correspondientes a los casos de usos descritos en la sección anterior.

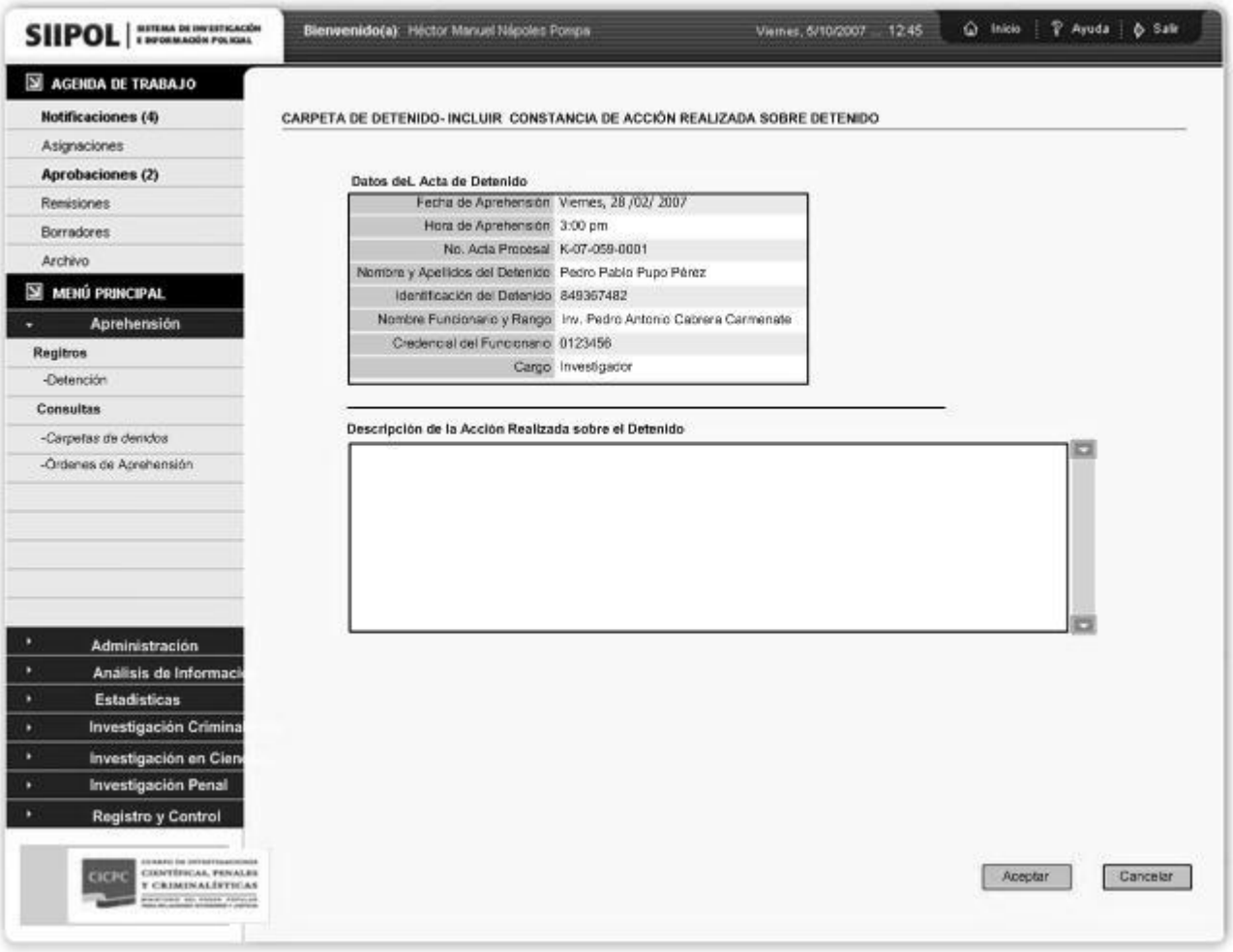


Fig. 2-4 Diseño de interfaz de usuario CU Gestionar constancia de acción realizada sobre detenido.

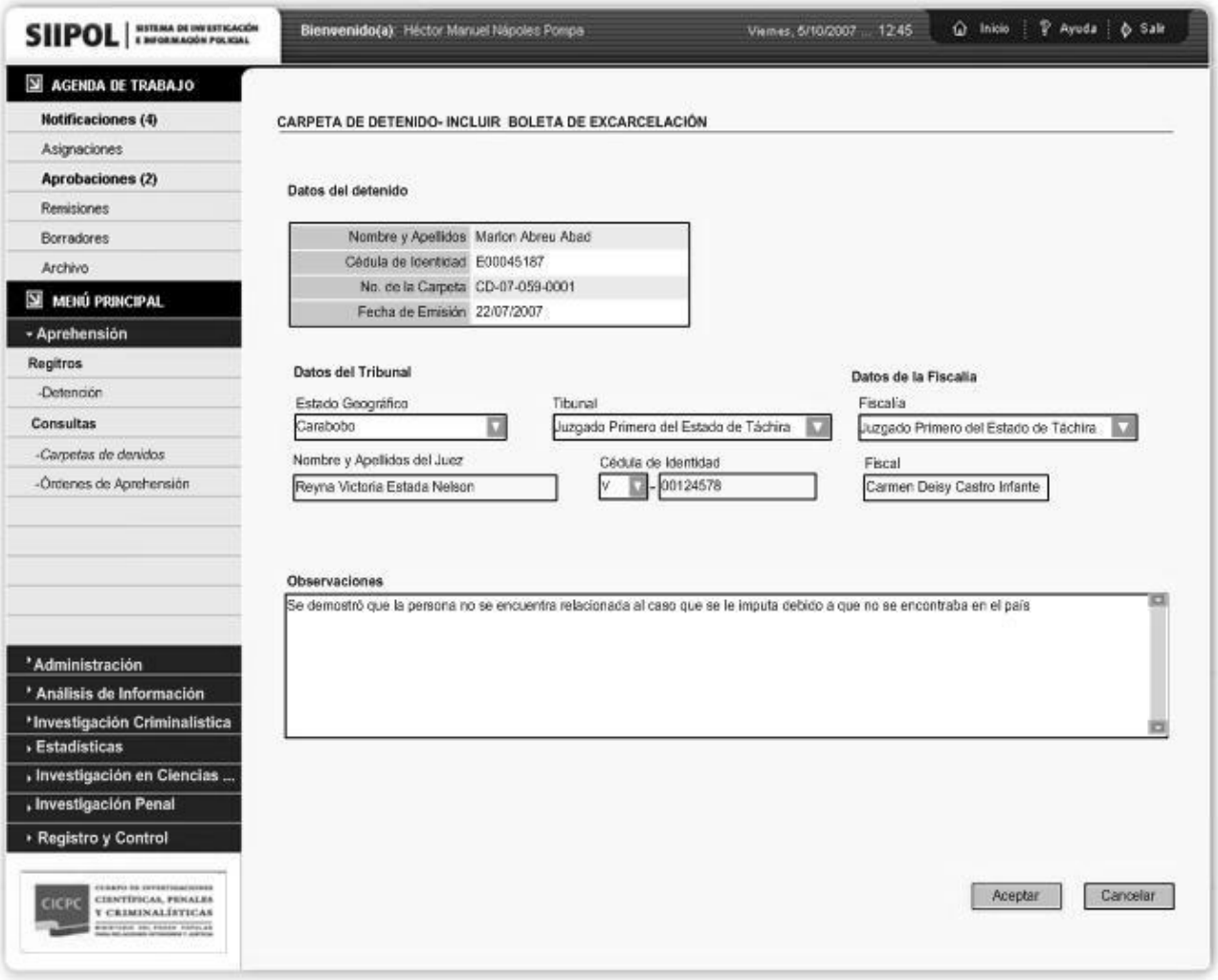


Fig. 2-5 Diseño de interfaz de usuario CU Gestionar Boleta de Excarcelación

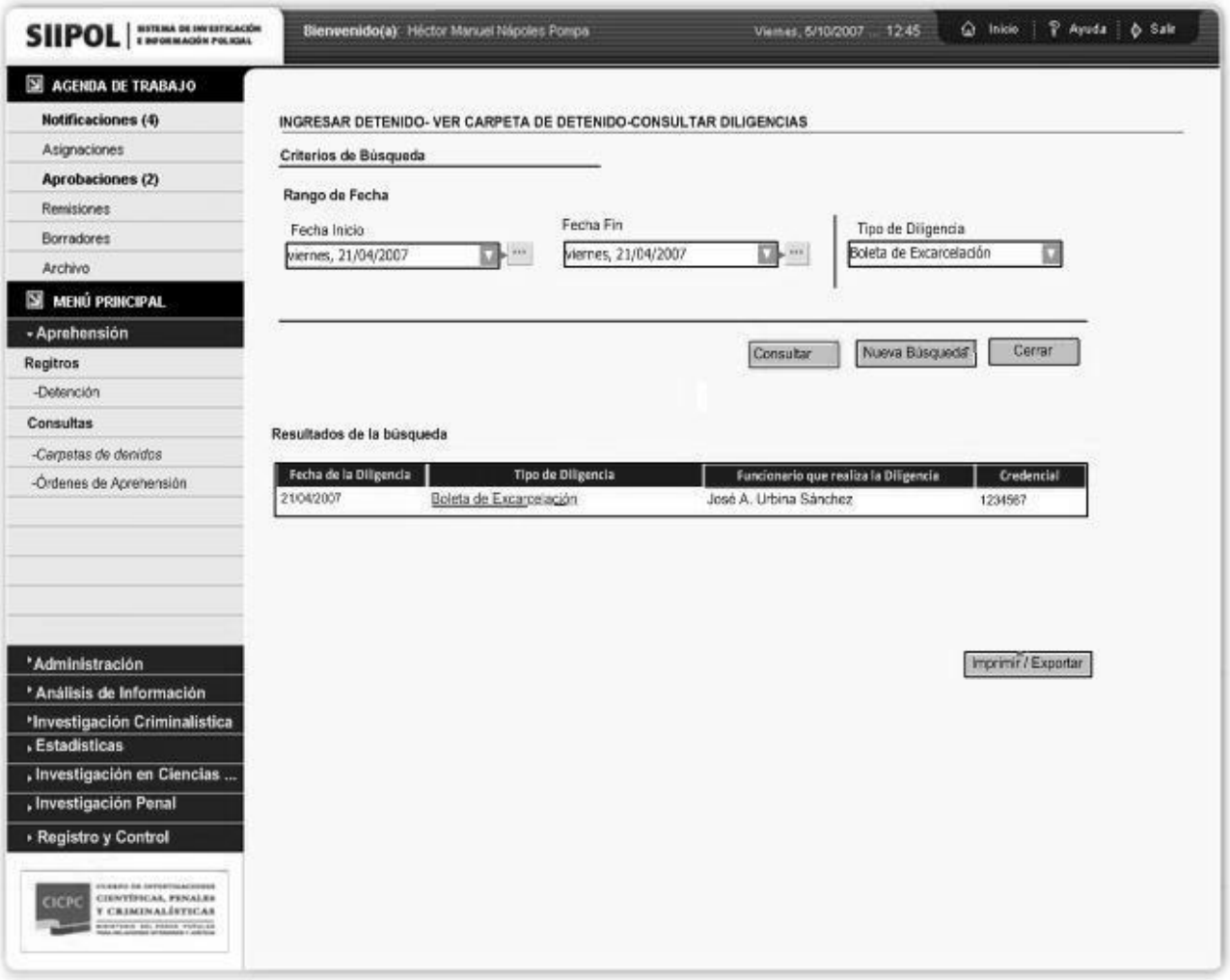


Fig. 2-5 Diseño de interfaz de usuario para el CU Consultar Diligencias sobre Detenido

2.4 Análisis

Como se planteaba anteriormente el análisis se centraba principalmente en el QUE debe hacer el sistema, sin tener que preocuparse por cuestiones propias del lenguaje. Más bien una frecuencia de pasos o acciones que sirven de entrada al diseño, incluso entidades que deberían relacionarse para cumplir una función determinada.

De los temas tratados anteriormente en este capítulo como por ejemplo: los procesos elementales, los requerimientos y los casos de uso, se puede apreciar que en el subsistema de Aprehensión se manejan 4 entidades principales tales como: la Carpeta del Detenido, el Acta de Detenido, la Boleta de Traslado al Retén y la Boleta de Excarcelación; la primera almacena toda la información referente con el detenido dentro del departamento de Aprehensión, como son los datos de la persona detenida los cuales son recogidos en la Planilla de Ingreso así como todas las diligencias que se realizan en dicho departamento; el Acta de Detenido, contiene la información referente al funcionario y a la acción que está acometiendo dicho funcionario con un detenido en específico; la Boleta de Traslado al Retén y la Boleta de Excarcelación para enviar al retén o liberar respetivamente al detenido.

A continuación se mostrarán los diagramas de clases del análisis de los casos de uso descritos anteriormente, aquellos relacionados principalmente con el proceso de ingreso en el departamento de Aprehensión y los vinculados con las acciones que son llevadas con los detenidos dentro del departamento.

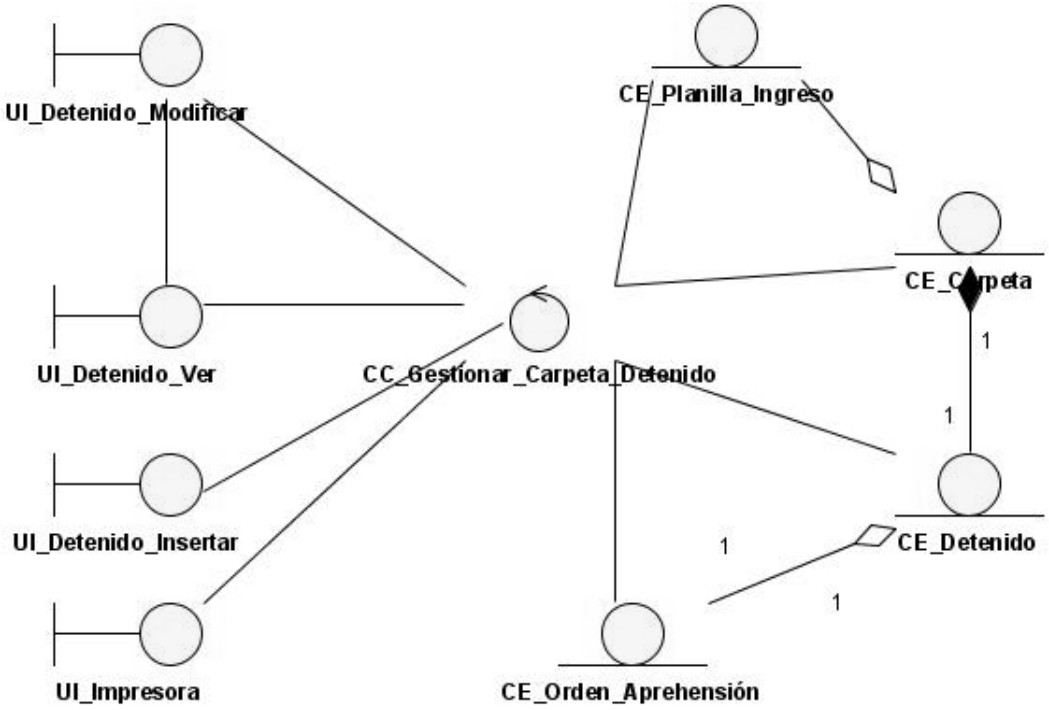


Fig. 2-6 Clases del Análisis CU Gestionar Carpeta de Detenido

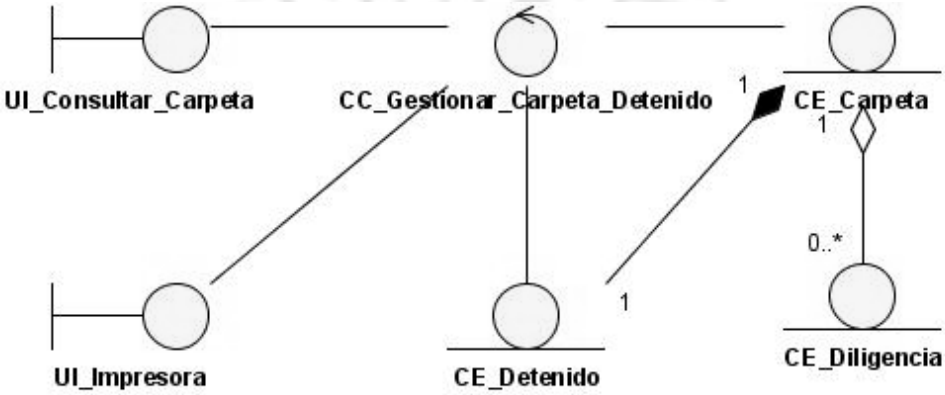


Fig. 2-7 Clases del Análisis CU Consultar Carpeta de Detenido

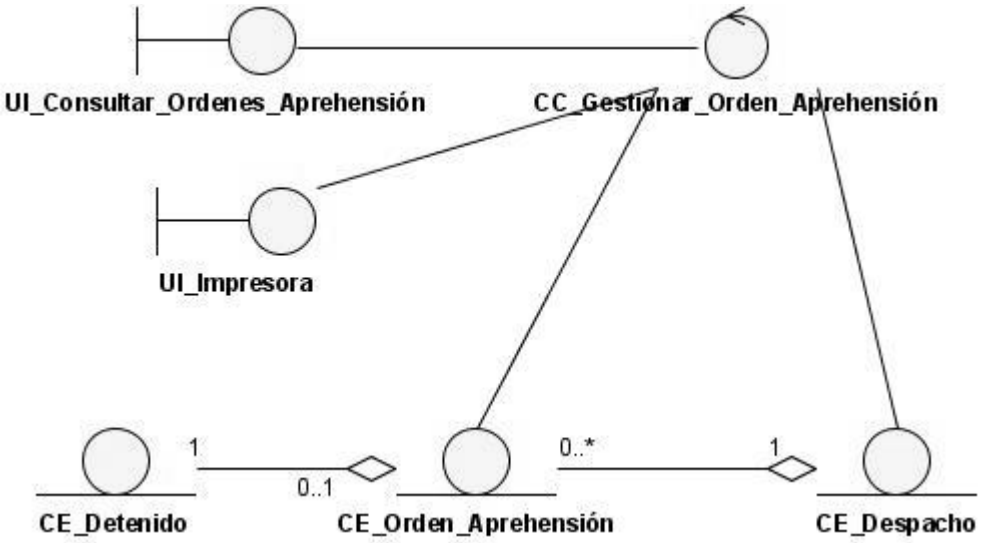


Fig. 2-8 Clases del Análisis CU Consultar Órdenes de Aprehensión

El análisis de estos primeros casos de uso provee del siguiente resultado, y es que, la principal entidad que se maneja es la Carpeta del Detenido, además de que solo se tienen cinco clases interfaz y dos clases controladoras, las cuales podrían llegar a convertirse en la misma. Esta entidad nunca se elimina.

Similar a los casos de uso anteriores son los casos de uso vinculados con las diligencias realizadas dentro del departamento de Aprehensión. Van a tener la misma peculiaridad de que no se pueden eliminar y salvo en el caso del Acta de Detenido que se puede modificar, las demás solo se pueden ver o insertar.

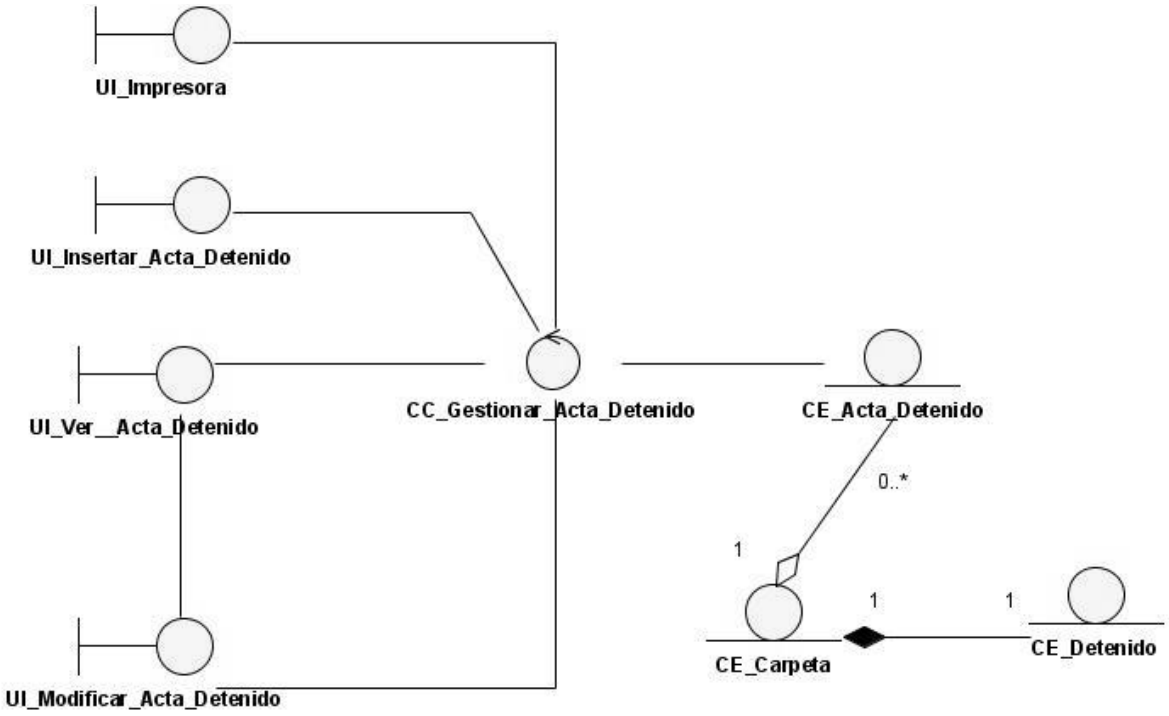


Fig. 2-9 Clases del Análisis CU Gestionar Acta de Detenido

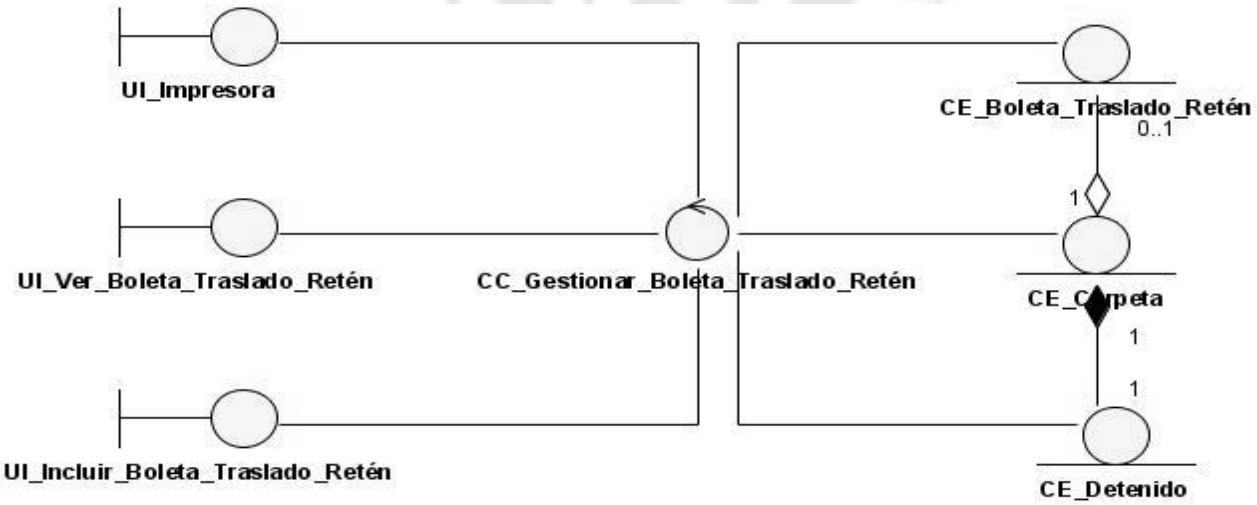


Fig. 2-10 Clases del Análisis CU Gestionar Boleta Traslado Retén

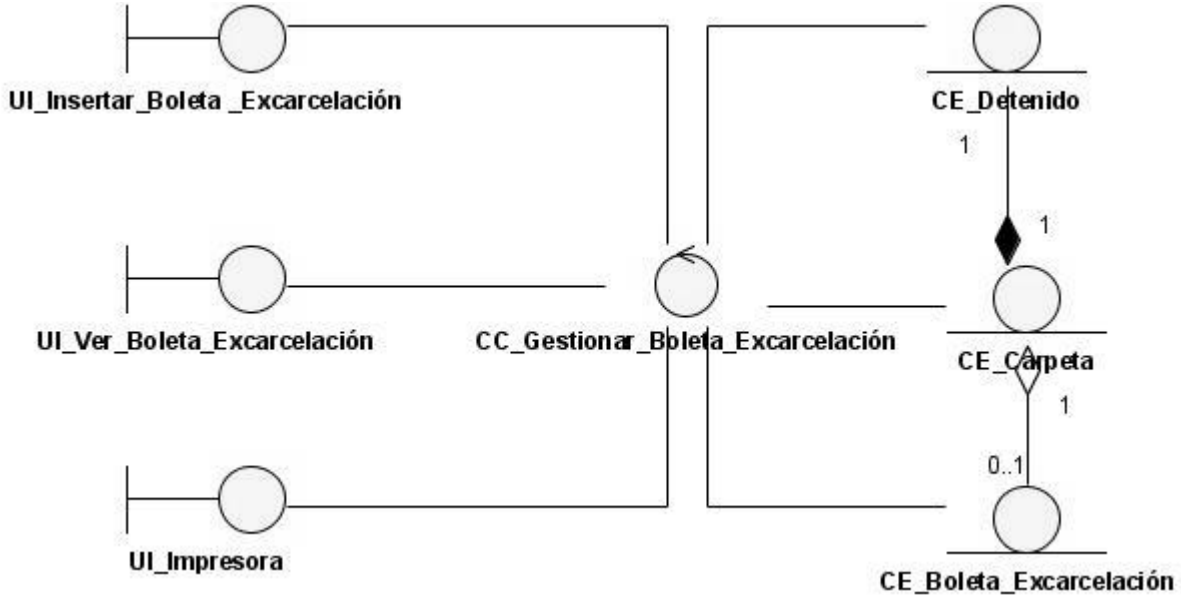


Fig. 2-11 Clases del Análisis CU Gestionar Boleta de Excarcelación

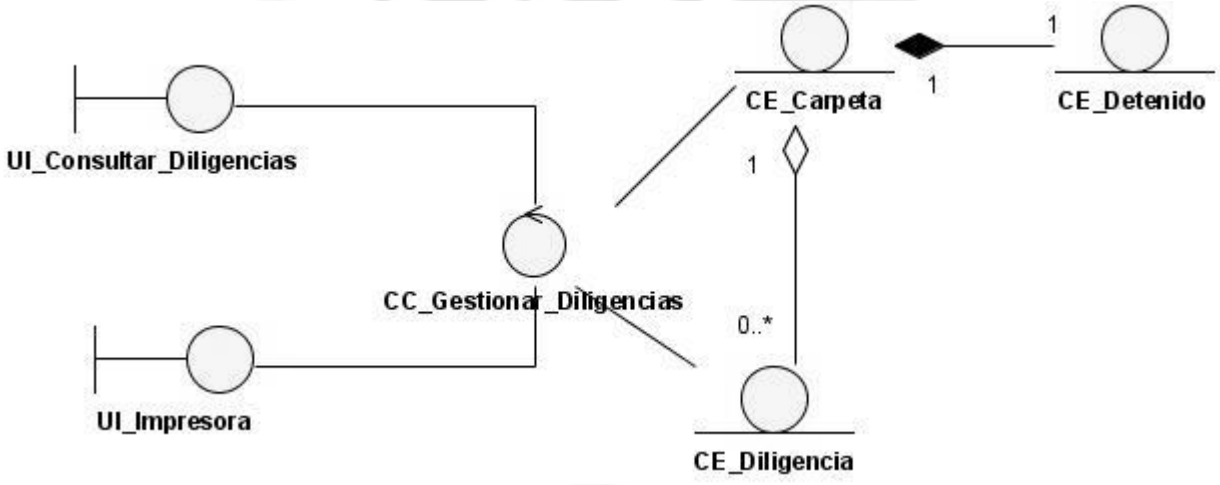


Fig. 2-12 Clases del Análisis CU Consultar Diligencias sobre Detenido

2.5 Diseño

Como se conoce, en el diseño ya más que existir la preocupación en el QUE debe hacer el sistema, los esfuerzos se centran en él COMO el sistema debe funcionar para dar cumplimiento a cada uno de las funcionalidades definidas.

Para esto debe tenerse en cuenta algunos criterios fundamentales tales como: la arquitectura definida por el equipo de arquitectura del proyecto, las buenas prácticas del uso de los patrones de diseño y el buen uso de las facilidades que brindan los frameworks que se están usando para desarrollar la aplicación. Antes de observar los diagramas de clases asociados a cada uno de los casos de uso, se detallaran primeramente diversos puntos de la arquitectura, así como patrones y configuraciones de los frameworks presentes en el desarrollo de la aplicación.

La arquitectura definida para la aplicación fue una arquitectura de **n- capas de tres niveles** la cual ofrece modularidad, escalabilidad, simplifica el desarrollo y aumenta la mantenibilidad del sistema; además de que para algunas capas existen un conjunto de clases que influyen en el desarrollo de la aplicación, en la capa de presentación la clase BaseBean, la cual contiene un conjunto de funcionalidades afines con todas las clases que sirven de respaldo a cada una de las páginas, los beans de respaldo como establece JSF , por lo que todos los beans de respaldo deberán heredar de dicha clase.

Con las clases del paquete de los servicios se relaciona la clase Carga, desarrollada por la arquitectura, la cual permite inicializar aquellos objetos que en el contexto de Hibernate fueron declarados lazy-initilization (una propiedad de Hibernate que permite cargar de la base datos solo los datos que son necesarios). Similar pasa con los objetos responsables del acceso a los datos, los cuales deben heredar del DaoGenerico y del DaoGenericoImpl, la interfaz y la implementación respectivamente, así como todos objetos del dominio que lo harán de la clase EntidadPersistenteBase.

Otros de los patrones que establece la arquitectura es el patrón **MVC**, el mismo permite que los cambios en la vista no afecten el resto de la aplicación, además de que se pueden mostrar distintos tipos de vistas sin afectar al modelo, no obstante JSF permite implementar este patrón especificando la vistas que son

las paginas .jsp, el controlador que son los beans de respaldo y el modelo que son las entidades del dominio.

Se puede apreciar el patrón Indirección, en el mismo se asigna la responsabilidad a un objeto intermedio para que medie entre otros componentes o servicios, y éstos no terminen directamente acoplados. El intermediario crea una indirección entre el resto de los componentes o servicios, este se pone de manifiesto en todas las capas de la aplicación desde la fachada, las clases responsables de los servicios y hasta los objetos de acceso a datos, permitiendo la flexibilidad y extensibilidad del sistema, pues cada una de las clases solo conoce a la interfaz pertinente, desacoplándolas por completo de la respectiva implementación. La implementación de una abstracción puede ser configurada en tiempo de ejecución. Además le es posible a un objeto cambiar su implementación en tiempo de ejecución, desacoplándolos.

En la aplicación aparece la interfaz AprehensionFacade, la cual es usada por los beans de respaldo para realizar las funcionalidades pertinentes, por tanto, lo que conoce el bean de respaldo es a la interfaz que le brindará estos métodos, pero en ningún momento le conocerá como está implementado. Dada esta situación, queda una interrogante ¿Cómo se realiza esta integración?, Spring dentro de las principales funcionalidades que brinda están las inyecciones de dependencia lo cual se traduce a que es el propio contenedor quien se encarga de “inyectarle” a cada interfaz la implementación correspondiente. Para el caso de los beans de respaldo, los cuales son declarados en el contexto de JSF, Spring brinda el VariableResolver, el cual le permite al contexto de JSF buscar los beans en el contexto de Spring; no obstante, esta vía no puede ser usada siempre, ya que en ocasiones será necesario dentro de la propia construcción de los beans de respaldo tener acceso a las clases de la fachada, y como en ese momento no estará creado el bean, aun no tendrá ”inyectada” la implementación correspondiente, por lo que hará falta el uso de otro patrón: ServiceLocator. En este patrón se crea un objeto, el cual será el responsable de abstraer todo el uso de JNDI (Java Naming and Directory Interface) y a diferencia de la inyección de dependencia, el objeto tendrá que ser el responsable de buscar su dependencia en el contenedor.

El patrón Fachada es un patrón GOF de estructura, el cual provee una interface unificada a un conjunto de funciones de un subsistema. Es una interface de alto nivel, para facilitar el uso del subsistema. En el subsistema de Aprehensión aparece la interfaz AprehensionFacade, la cual es el resultado de dicho

patrón, dicha interfaz será el punto de entrada de todos los beans de JSF a cada una de las funcionalidades del subsistema, la implementación de dicha fachada será el AprehensionFacadeImpl la cual tendrá acceso a todos las clases servicios del subsistema para así poder delegar su funcionalidad en cada uno de estos servicios, propiciando que aumente la cohesión del sistema.

También se encuentra el patrón **DAO (data access object o objetos de acceso a datos traducido al español)**, se trata de abstraer y encapsular el acceso a los datos, en este caso a la base de datos donde se encuentran los datos.

En los diagramas de clases del diseño web que se mostrarán a continuación se podrán observar cada una de estas especificaciones, salvo las relaciones con el BaseBean, Carga y el DaoGenerico ya que no fueron desarrolladas en este subsistema, para ganar en claridad en los diagramas y buscando sencillez en los mismos se decidió dividir los diagramas de cada caso de uso en dos partes, la primera donde se especifica todo lo relacionado con el bean de respaldo, la lógica del negocio y el acceso a los datos y en la otra todo lo relacionado con la parte WEB.

Diagramas de clases de Diseño Web

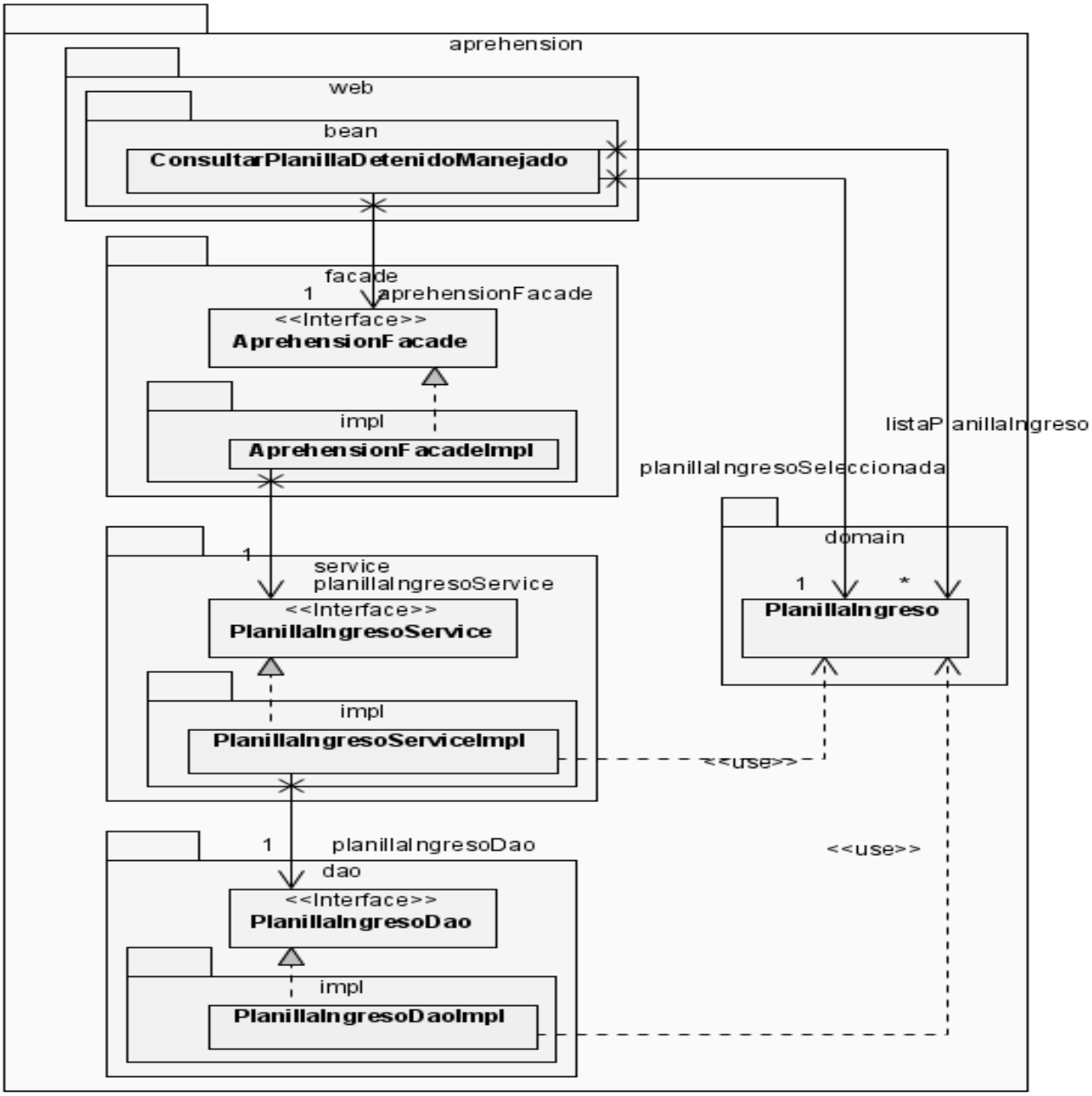


Fig. 2-13 Diagrama de Clases CU Consultar Carpeta de Detenido sin la parte web

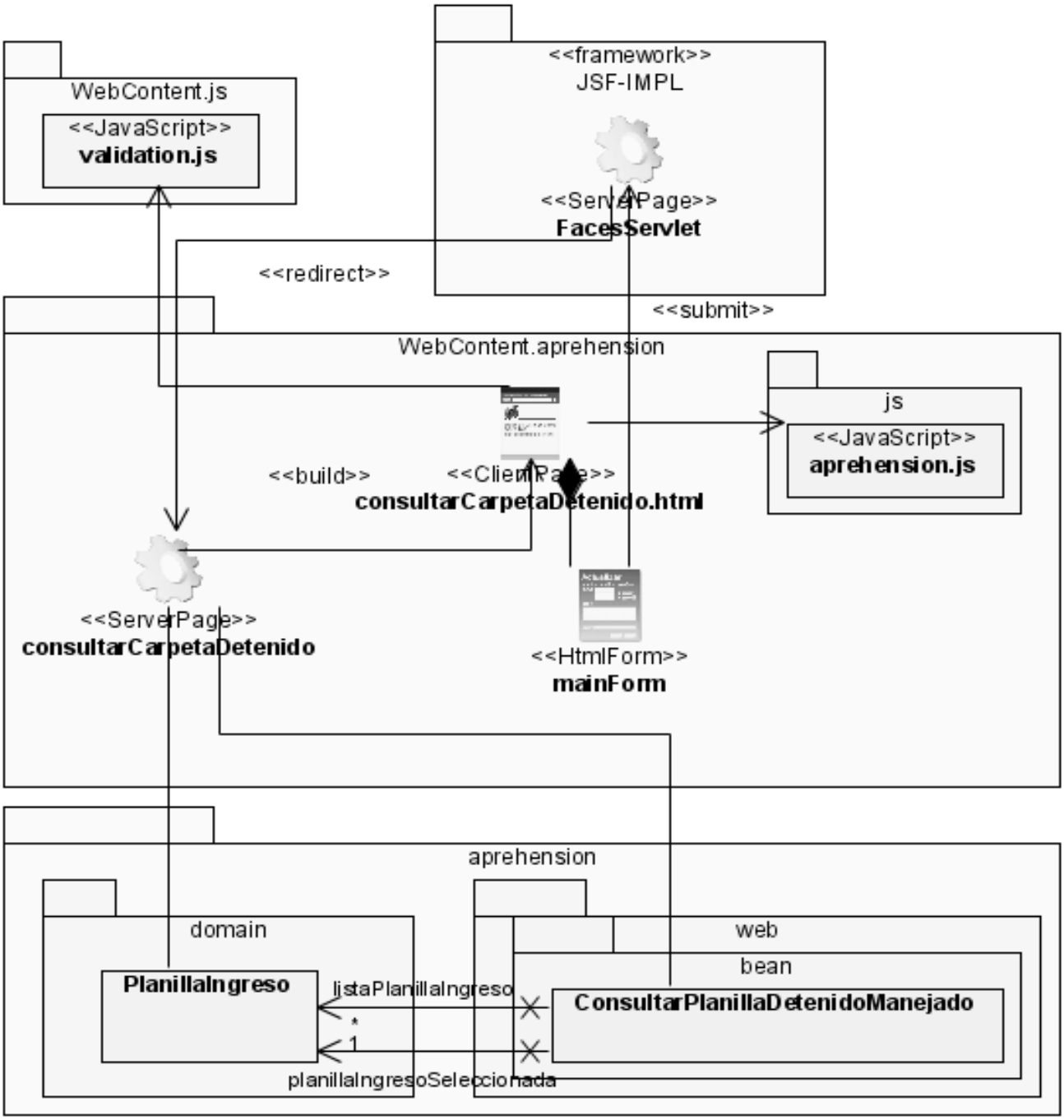


Fig. 2-14 Diagrama de Clases CU Consultar Carpeta de Detenido con la parte web

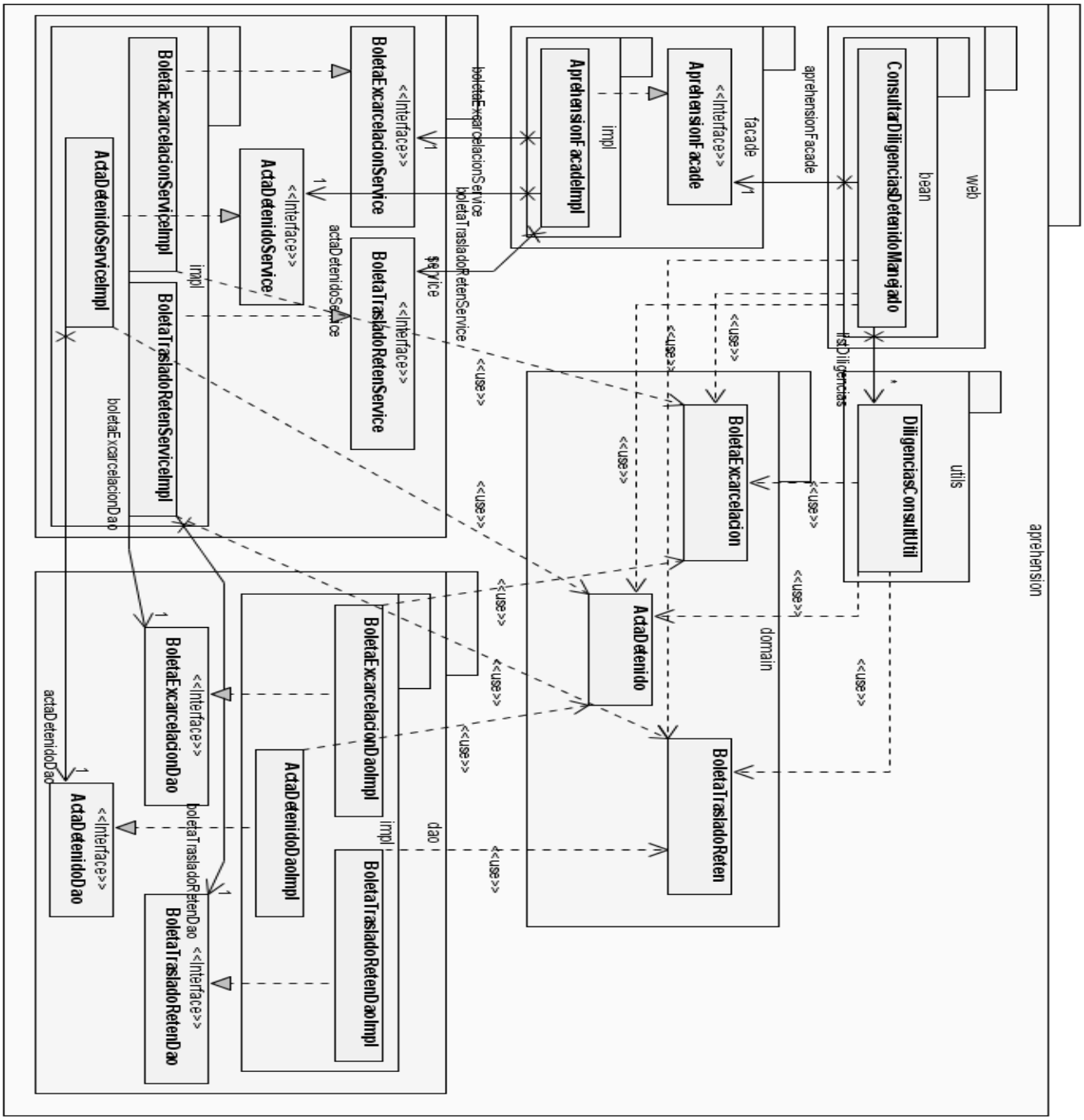


Fig. 2-15 Diagrama de Clases CU Consultar Diligencias sobre Detenido sin la parte web

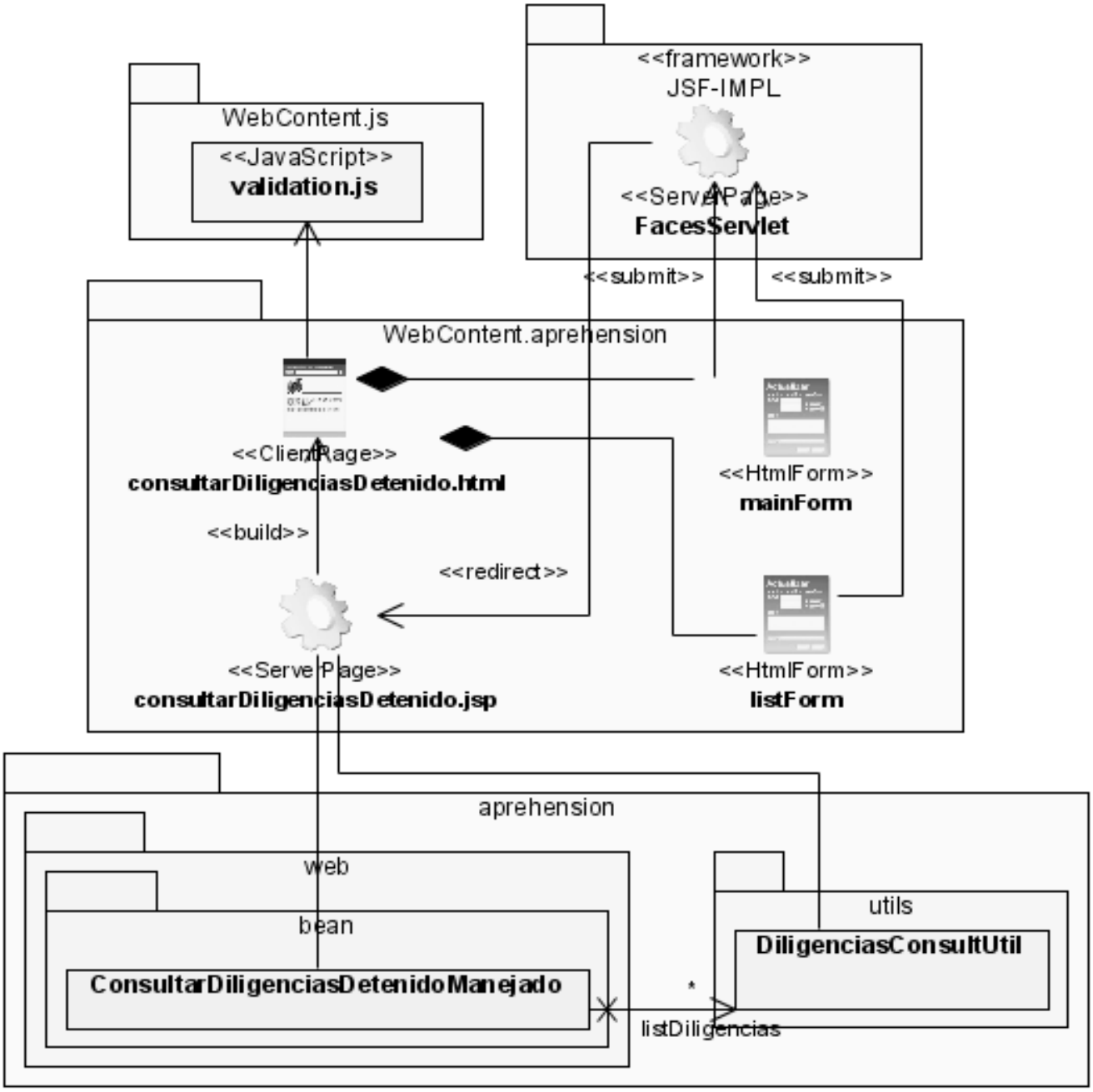


Fig. 2-16 Diagrama de Clases CU Consultar Diligencias sobre Detenido con la parte web

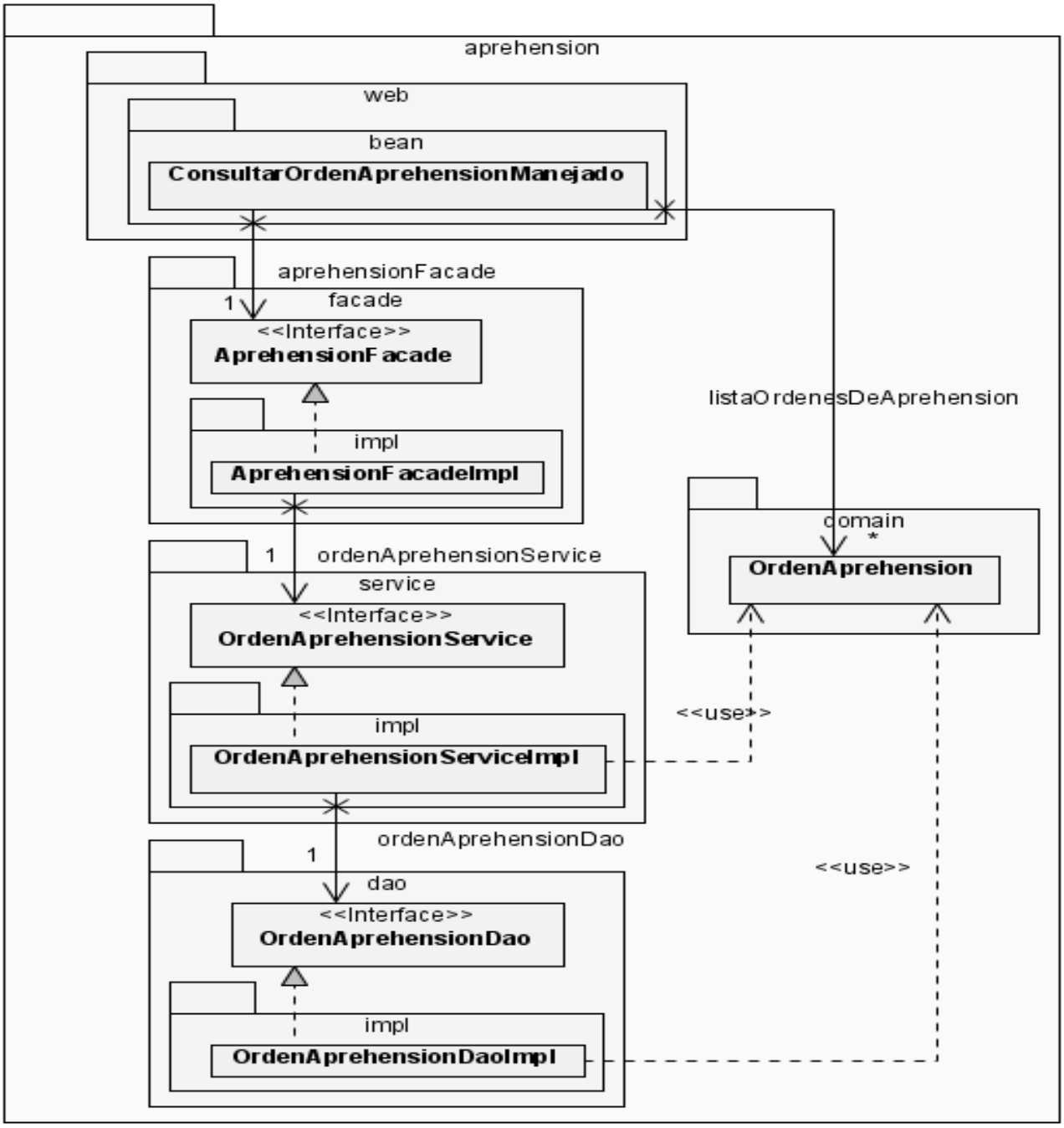


Fig. 2-17 Diagrama de Clases CU Consultar Órdenes de Aprehensión sin la parte web

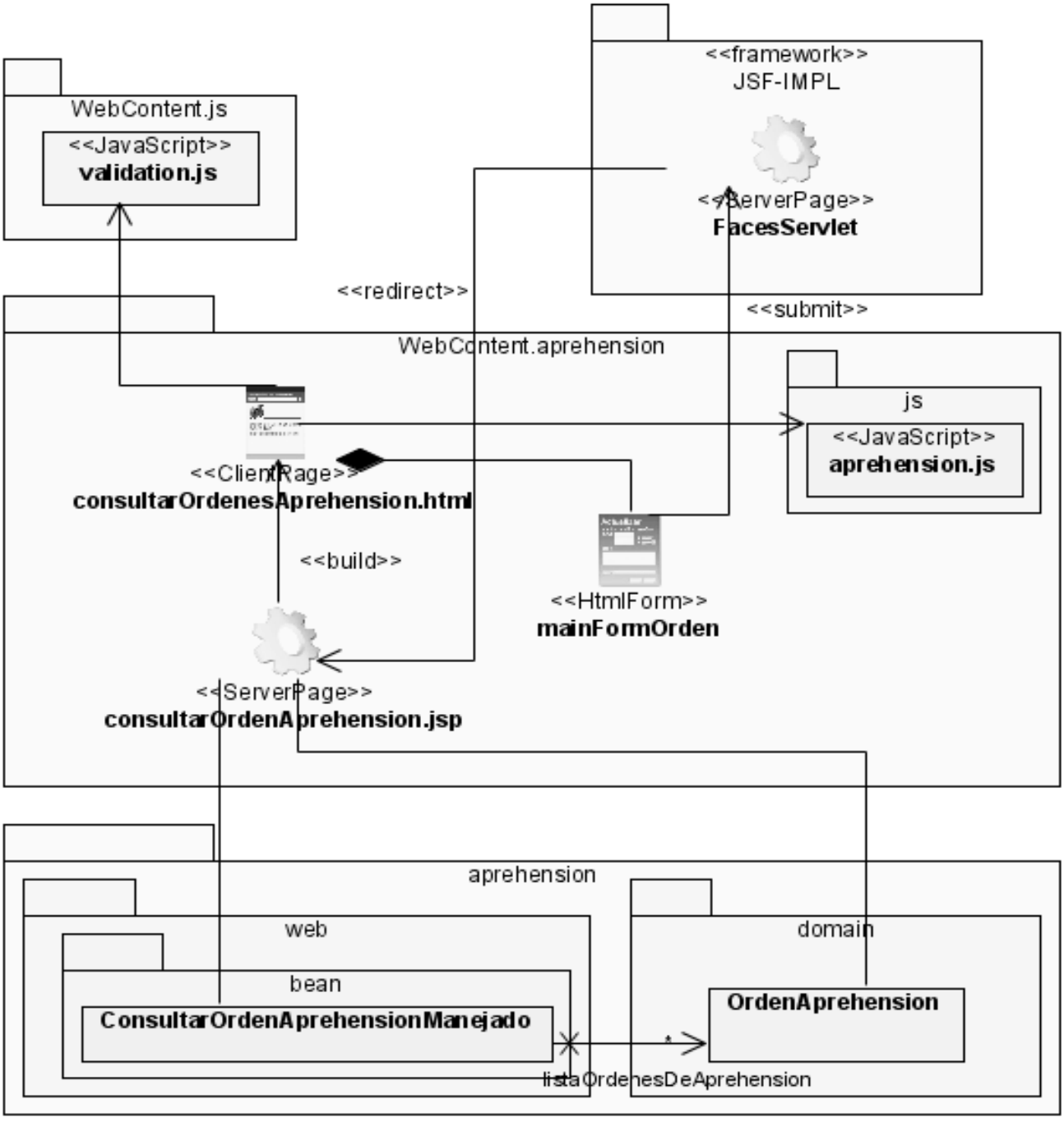


Fig. 2-18 Diagrama de Clases CU Consultar Órdenes de Aprehensión con la parte web

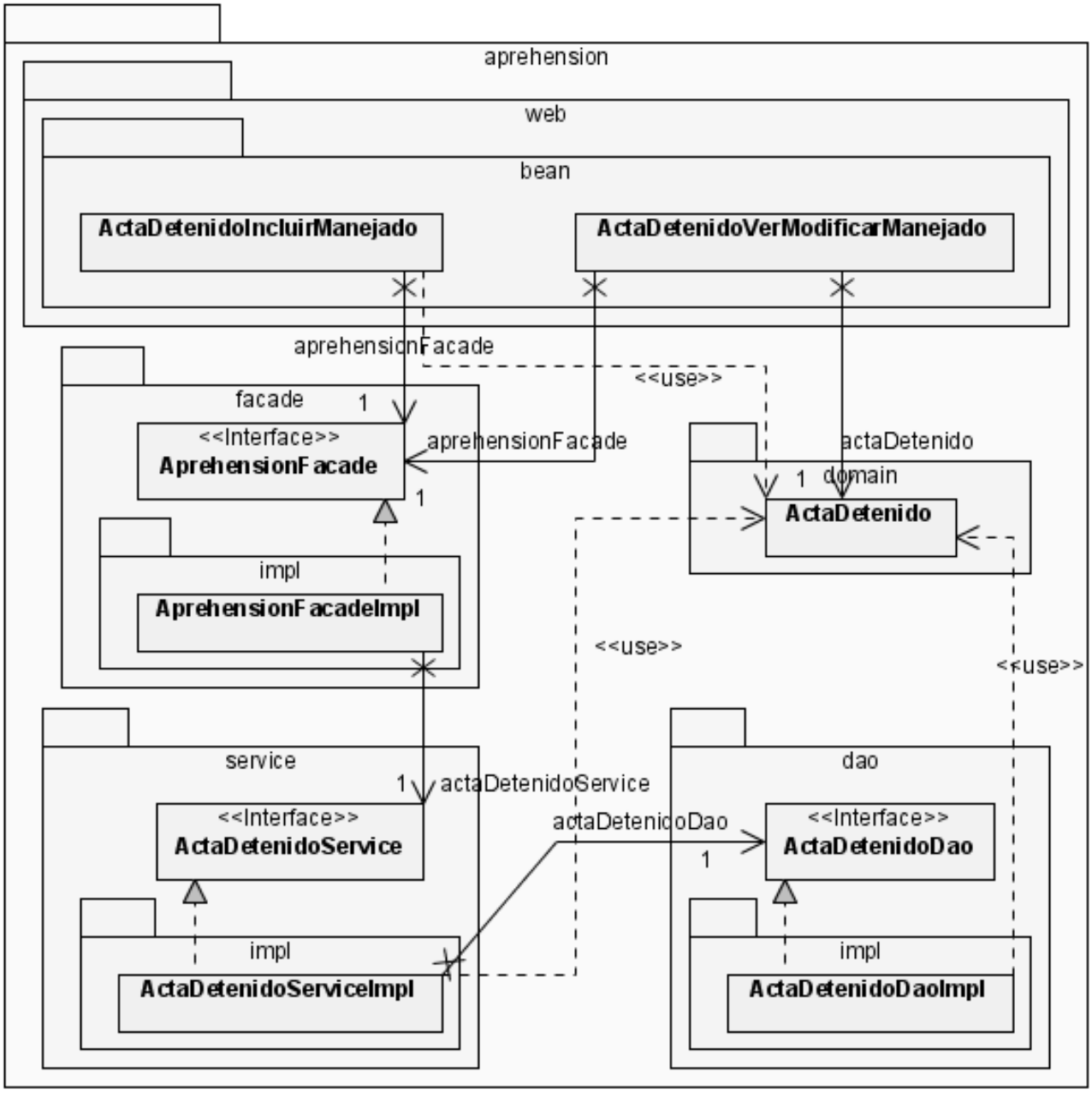


Fig. 2-19 Diagrama de Clases CU Gestionar constancia de acción realizada sobre detenido con la parte web

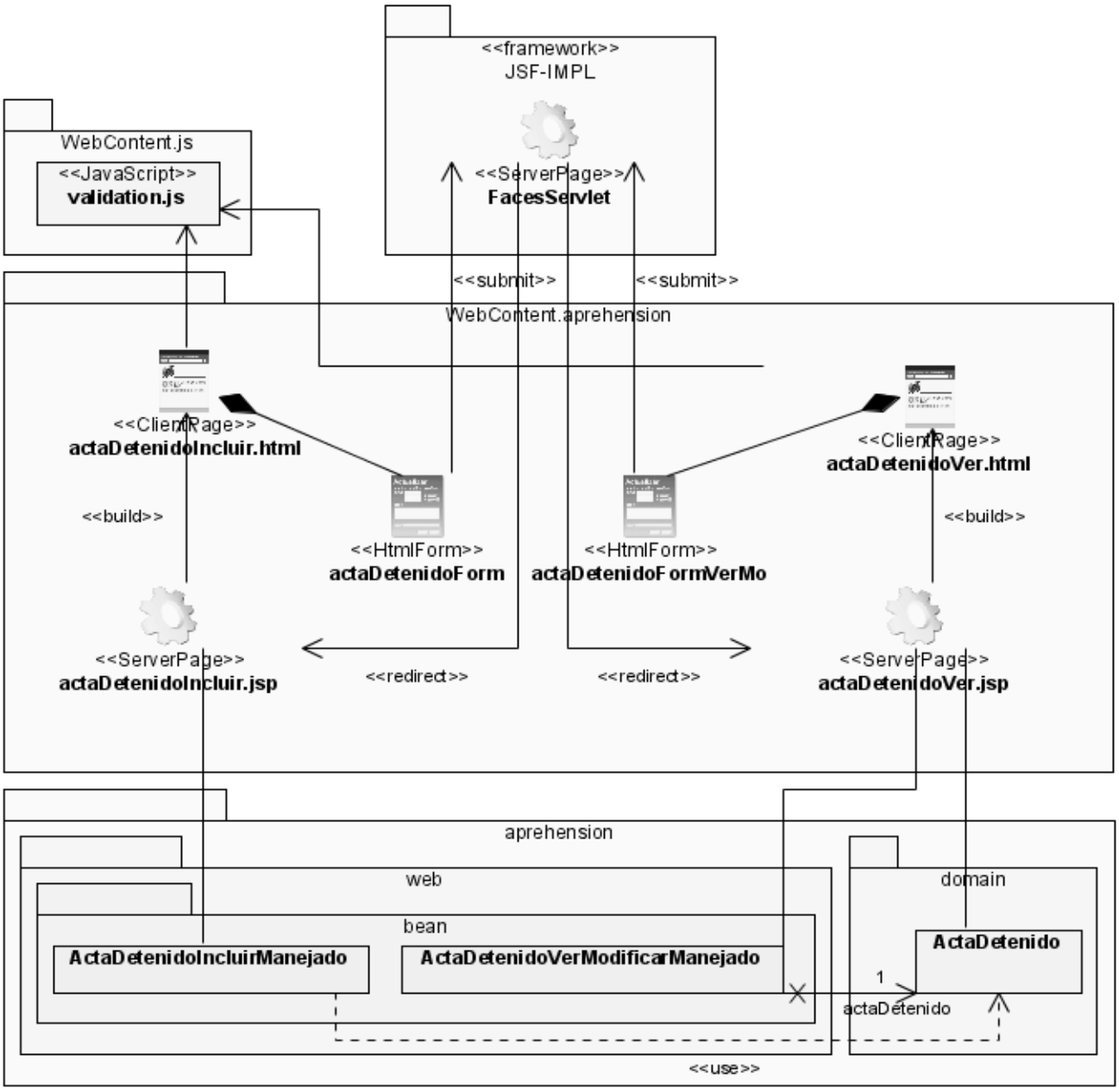


Fig. 2-20 Diagrama de Clases CU Gestionar constancia de acción realizada sobre detenido sin la parte web

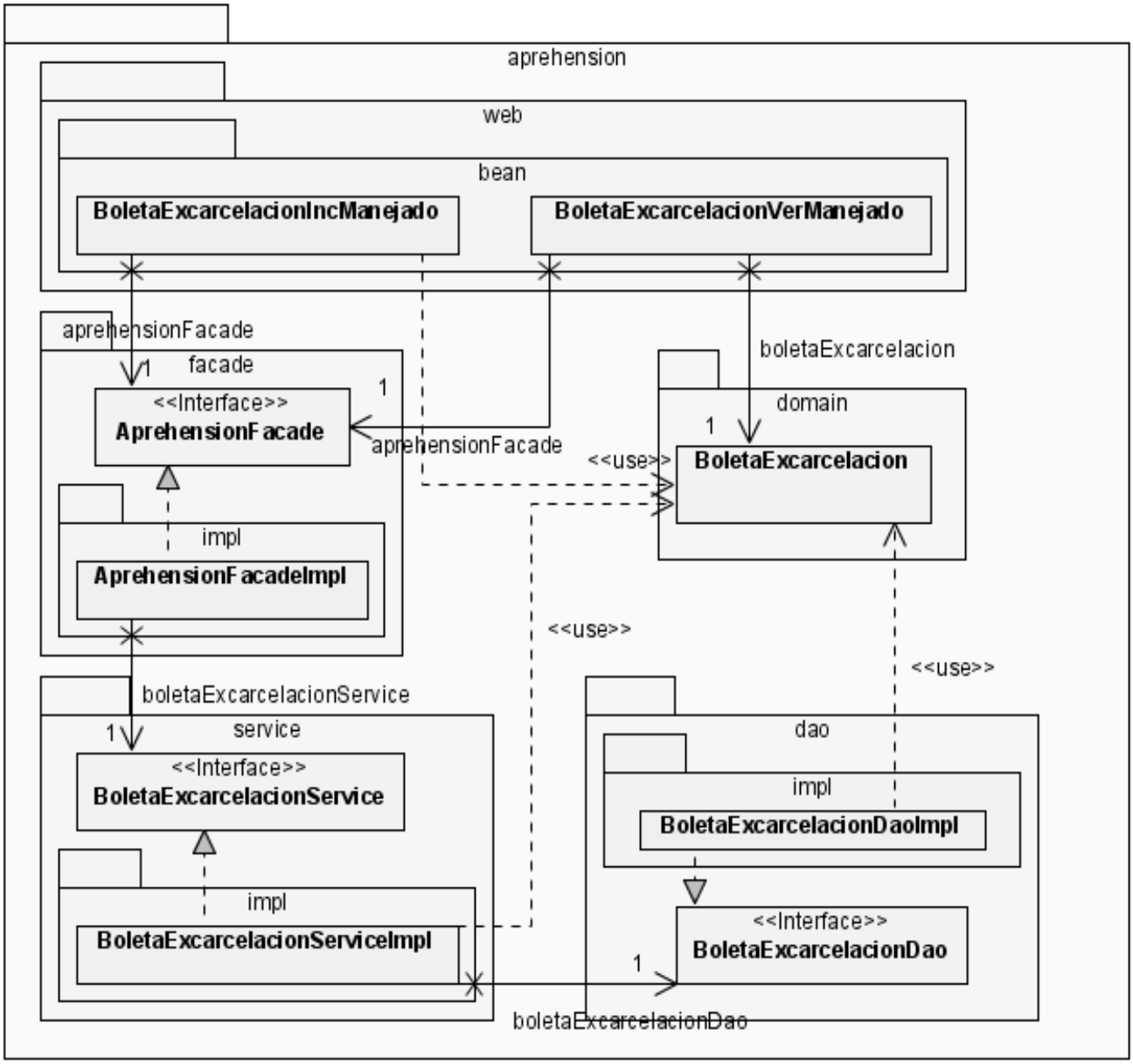


Fig. 2-21 Diagrama de Clases CU Gestionar Boleta de Excarcelación sin la parte web

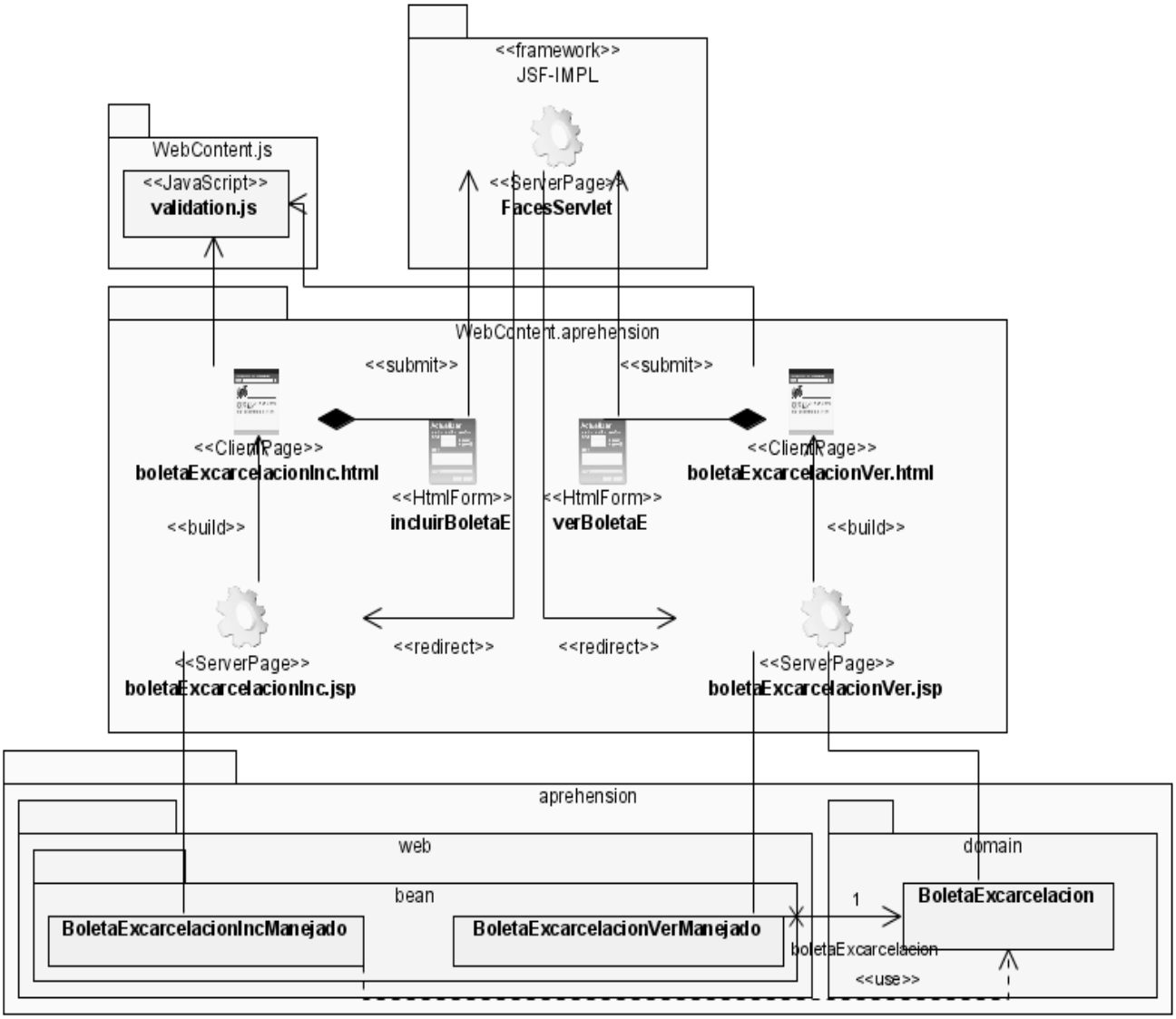


Fig. 2-22 Diagrama de Clases CU Gestionar Boleta de Excarcelación con la parte web

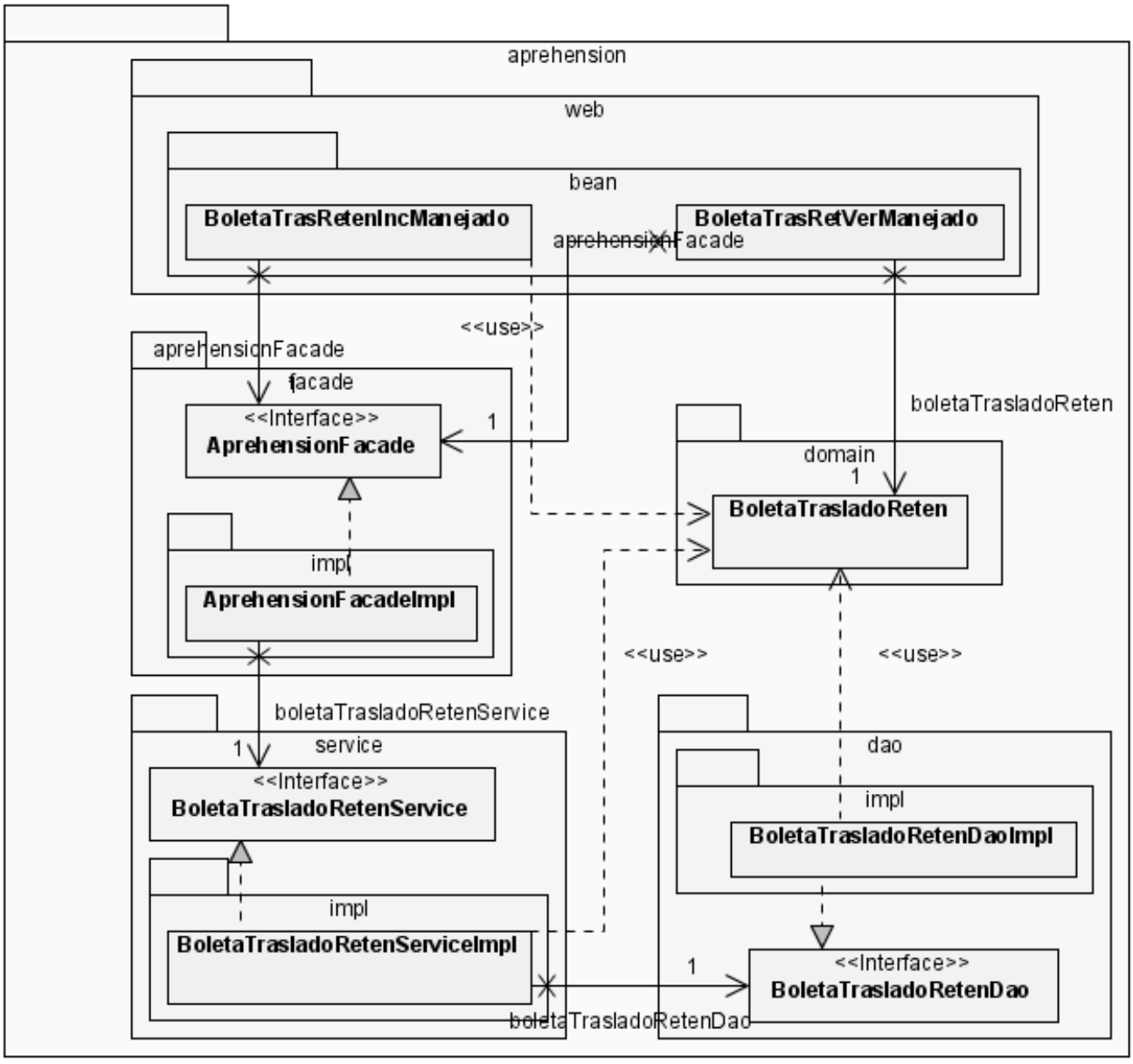


Fig. 2-23 Diagrama de Clases CU Gestionar Boleta de Traslado al Reten sin la parte web

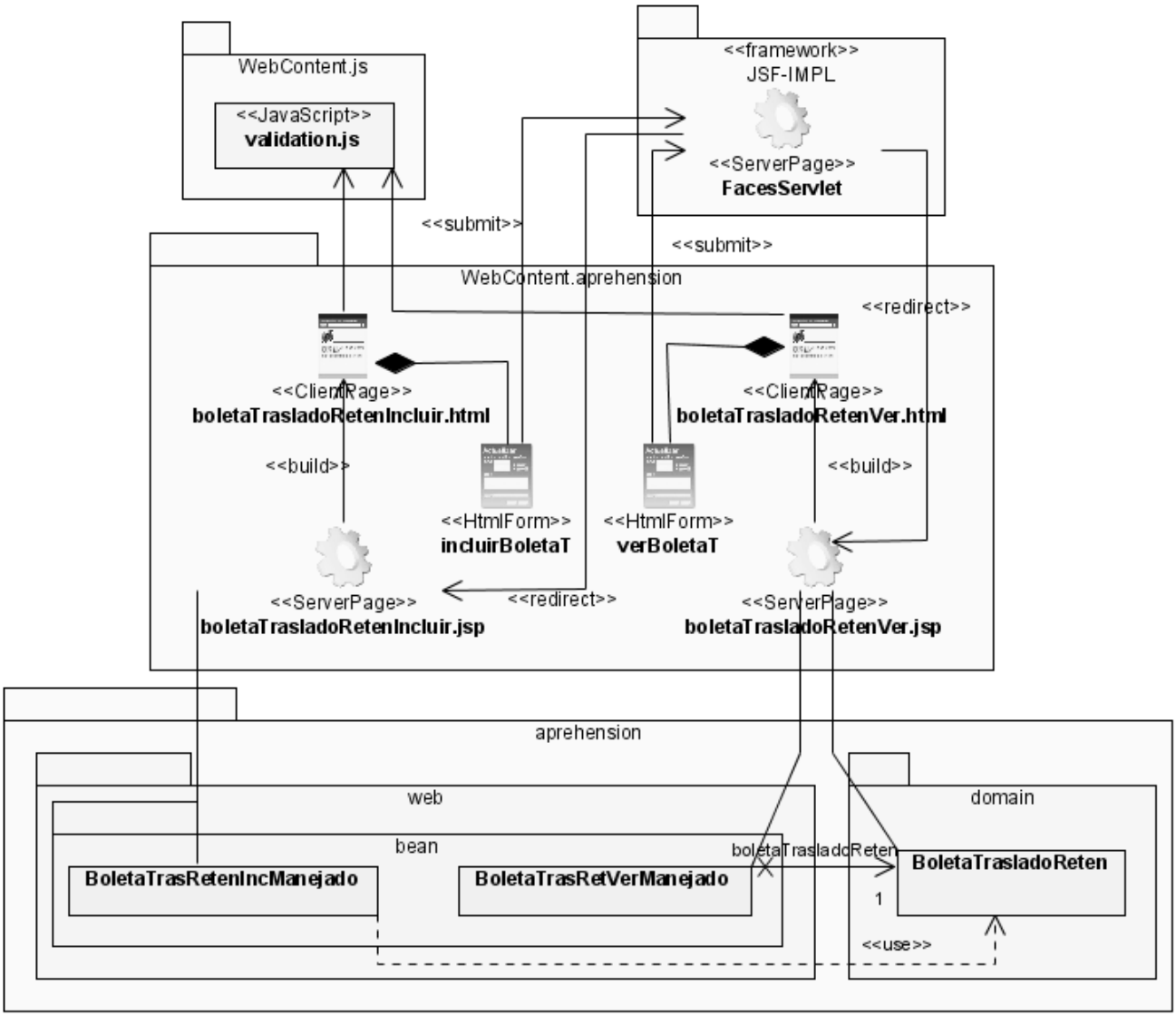


Fig. 2-24 Diagrama de Clases CU Gestionar Boleta de Traslado al Reten con la parte web

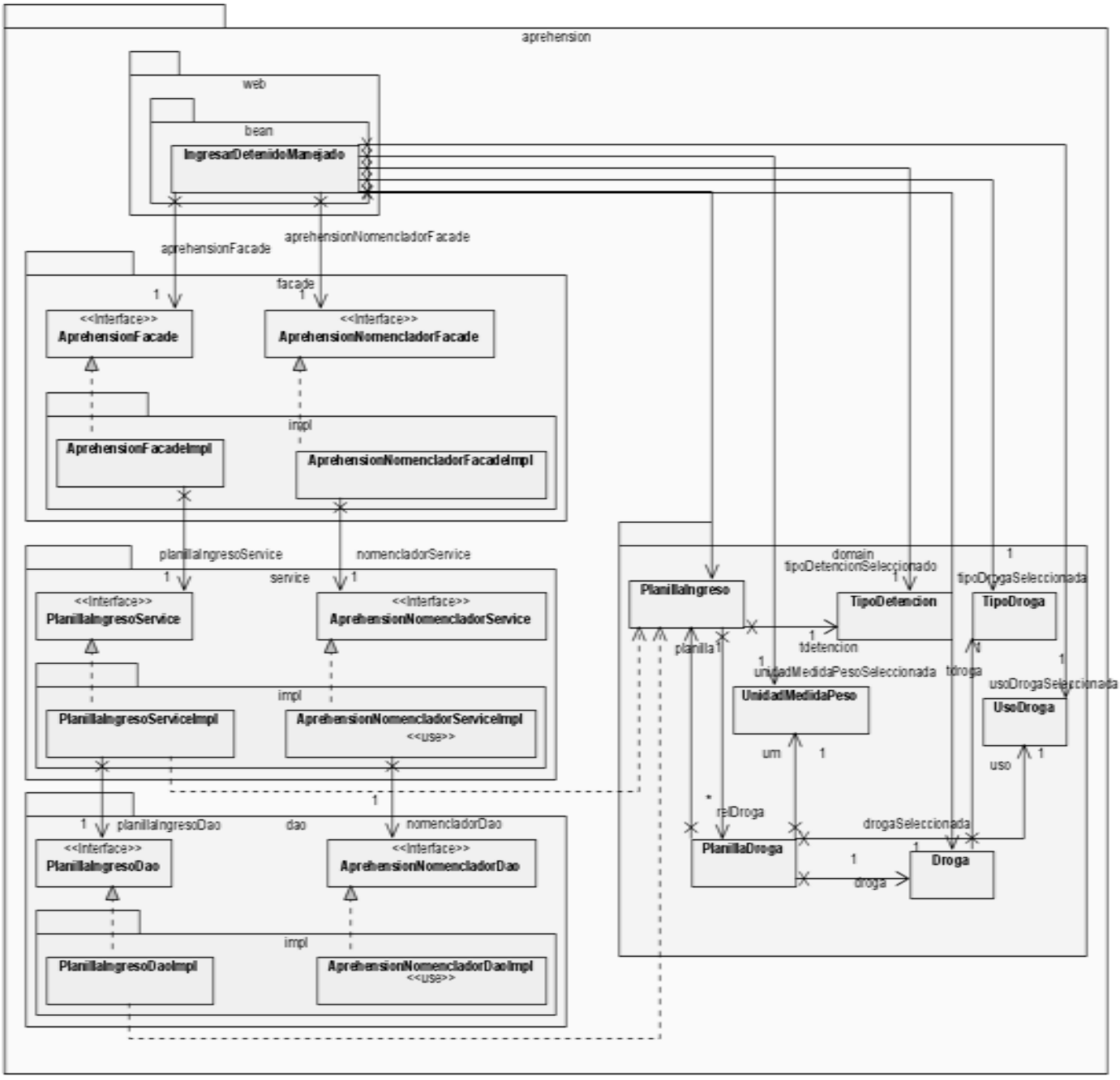


Fig. 2-25 Diagrama de Clases CU Gestionar Detenido sin la parte web

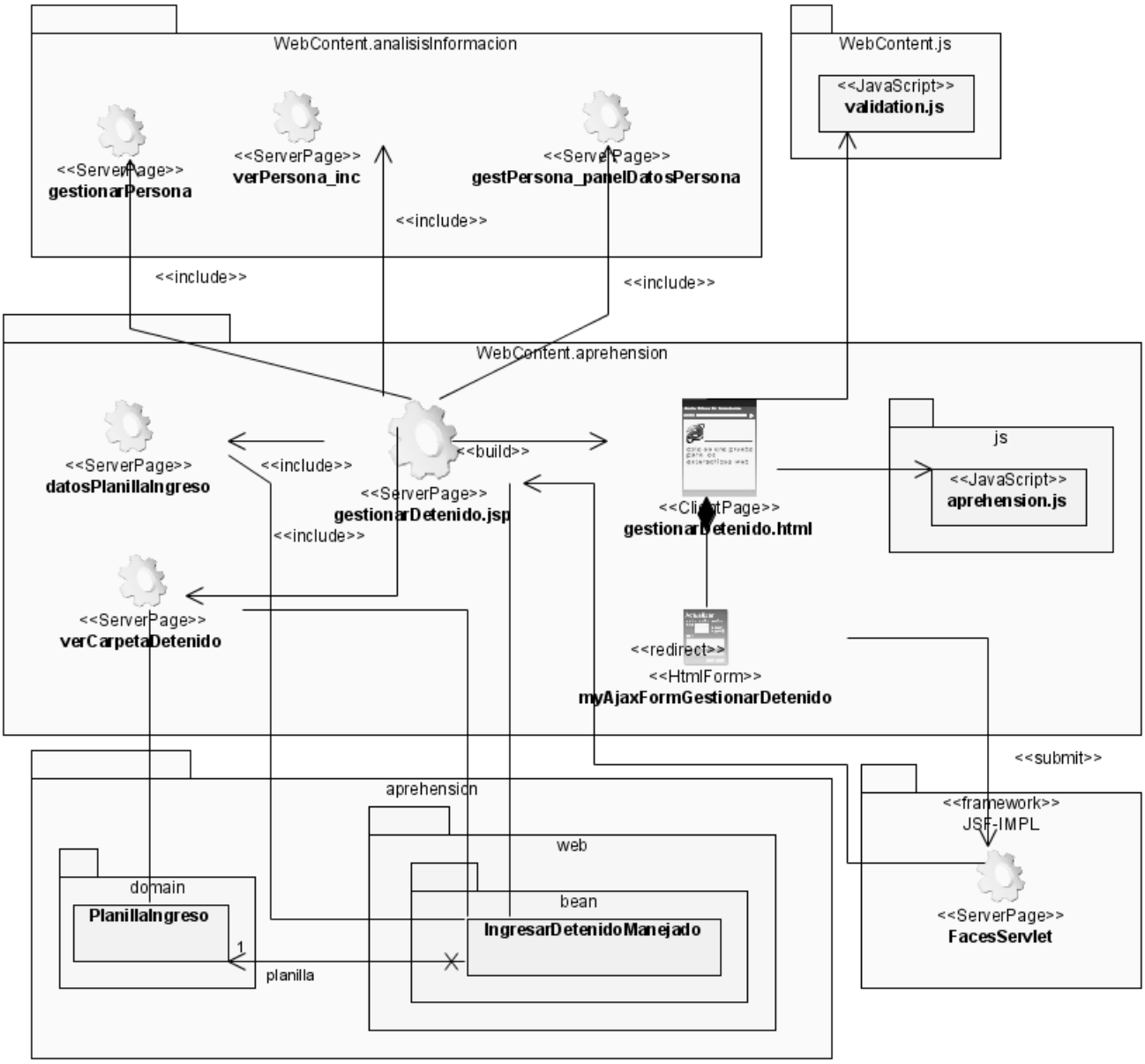


Fig. 2-26 Diagrama de Clases CU Gestionar Detenido con la parte web

Descripciones textuales de algunas de las clases:

Nombre: AprehensionFacadeImpl		
Atributo		**Tipo**
planillaIngresoService		PlanillaIngresoService
ordenAprehensionService		OrdenAprehensionService
actaDetenidoService		ActaDetenidoService
boletaTrasladoRetenService		BoletaTrasladoRetenService
boletaExcarcelacionService		BoletaExcarcelacionService
investigacionPenalFacade		InvestigacionPenalFacade
Para cada responsabilidad:		
Nombre:	obtenerOrdenAprehension(String primerNombre,String segundoNombre, String primerApellido,String segundoApellido, String letraCedula, String noCedula, String noOrden, String noExpediente,Date fechaEmisionInicio, Date fechaEmisionFinal, String estado, String fiscalia, String delito, String tribunal)	
Descripción:	Obtiene las órdenes de Aprehensión dado un conjunto de parámetros.	
Nombre:	ingresarPlanilla(Date fechaRedaccion, Date horaRedaccion, Funcionario funcionario, Persona detenido)	
Descripción:	Permite ingresar una Planilla de Ingreso.	
Nombre:	obtenerPorId(Integer id)	

Descripción:	Obtiene una Orden de Aprehensión por id.
Nombre:	obtenerPorPersona(Integer idPersona)
Descripción:	Obtiene una Orden de Aprehensión por id. de Persona.
Nombre:	obtenerPlanillaDetenido(String noCarpeta, String primerNombre, String segundoNombre, String primerApellido, String segundoApellido, String letraCedula, String noCedula, Date fechaCreacionInicio, Date fechaCreacionFin, Date fechaAprehensionInicio, Date fechaAprehensionFin, TipoDelito tipoDelito);
Descripción:	Obtiene las planillas de detenido que concuerden con el conjunto de parámetros.
Nombre:	obtenerPlanillaDetenido(Integer idPersona)
Descripción:	Obtiene la Planilla de Ingreso por id. de Persona.
Nombre:	guardarPlanillaIngreso(PlanillaIngreso planillaIngreso)
Descripción:	Guarda la Planilla de Ingreso.
Nombre:	obtenerOrdenAprehensionConCaso(Integer idOrden)
Descripción:	Obtiene una orden de Aprehensión con Acta Procesal.
Nombre:	obtenerPlanillaIngreso(Integer idPlanilla)
Descripción:	Obtiene la Planilla de Ingreso por id.
Nombre:	obtenerPlanillaDetenidoPorOrden(Integer idOrdenAprehension)
Descripción:	Obtiene la Planilla de Ingreso por id. de Orden de Aprehensión.

Nombre:	salvarActaDetenido(ActaDetenido actaDetenido)
Descripción:	Salva un acta de detenido.
Nombre:	salvarBoletaTrasladoReten(BoletaTrasladoReten boletaTrasladoReten)
Descripción:	Salva una boleta de traslado al retén.
Nombre:	salvarBoletaExcarcelacion(BoletaExcarcelacion boletaExcarcelacion)
Descripción:	Salva una boleta de excarcelación.
Nombre:	obtenerActasDetenidoByFecha(Integer idPlanilla, Date fechaInicio,Date fechaFin);
Descripción:	Obtiene las actas de detenidos pertenecientes a una carpeta, dado un rango de fecha.
Nombre:	actualizarActaDetenido(ActaDetenido actaDetenido)
Descripción:	Actualiza un acta de detenido.
Nombre:	obtenerBoletaExcarcelacionByCarpeta(Integer idCarpeta);
Descripción:	Obtiene la Boleta de Excarcelación de la Carpeta.
Nombre:	obtenerBoletaTrasladoRetenByCarpeta(Integer idCarpeta)
Descripción:	Obtiene la Boleta de Traslado al Retén de la Carpeta.
Nombre:	obtenerBoletaExcarcelacionByCarpetaFecha(Integer idCarpeta,Date fechaI, Date fechaF)
Descripción:	Obtiene una Boleta de Excarcelación dado una carpeta y un rango de fecha.

Nombre:	obtenerBoletaTrasladoRetenByCarpetaFecha(Integer idCarpeta,Date fechaI, Date fechaF);
Descripción:	Obtiene una Boleta de Traslado al Retén dado una carpeta y un rango de fecha.
Nombre:	obtenerBoletaTrasladoRetenById(Integer idBoleta);
Descripción:	Obtiene una Boleta de Traslado al Retén por id.
Nombre:	obtenerExcarcelacionById(Integer idBoleta)
Descripción:	Obtiene una Boleta de Excarcelación por id

Nombre: IngresarDetenidoManejado	
Atributo	**Tipo**
inclPersona_Detenido	BaseBean
aj_inclPersona_Detenido	Include
aj_inclDatosPlanilla_viewId	String
aj_inclPersona_Detenido_viewId	String
tabPanel	HtmlTabPanel
tablaAcompanantes	UIData
tablaDrogas	UIData
viendo	boolean

incluir	boolean
ver	boolean
modificar	boolean
detenido	Persona
formulaDecadactilar	String
banda	String
planilla	PlanillaIngreso
ordenAprehension	OrdenAprehension
fechaDelito	Date
modusOperandi	String
listaTipoDelito	SelectItem[]
tipoDelitoSeleccionado	TipoDelito
listaNaturalezaDelito	SelectItem[]
naturalezaDelitoSeleccionada	NaturalezaDelito
listaGradoDelito	SelectItem[]
gradoDelitoSeleccionado	GradoDelito
listaEstados	SelectItem[]
estadoSeleccionado	EstadoVenezolano

listaMunicipios	SelectItem[]
municipioSeleccionado	Municipio
listaParroquias	SelectItem[]
parroquiaSeleccionada	Parroquia
calleDelito	String
fechaDetencion	Date
listaEstadosDetencion	SelectItem[]
estadoSeleccionadoDetencion	EstadoVenezolano
listaMunicipiosDetencion	SelectItem[]
municipioSeleccionadoDetencion	Municipio
listaParroquiasDetencion	SelectItem[]
parroquiaSeleccionadaDetencion	Parroquia
calleDetencion	String
listaTipoDetencion	SelectItem[]
tipoDetencionSeleccionado	TipoDetencion
listaCaracteristicaDetencion	SelectItem[]
caracteristicaDetencionSeleccionado	CaracteristicaDetencion
listaAcompanantes	LinkedList<Persona>

listaDrogasEncontradas	LinkedList<PlanillaDroga>
listaDrogas	SelectItem[]
drogaSeleccionada	Droga
listaTipoDroga	SelectItem[]
tipoDrogaSeleccionada	TipoDroga
listaUsoDroga	SelectItem[]
usoDrogaSeleccionada	UsoDroga
listaUnidadMedidaPeso	SelectItem[]
unidadMedidaPesoSeleccionada	UnidadMedidaPeso
cantidadDroga	String
nombreDroga	String
nomencladorFacade	NomencladorFacade
aprehensionNomencladorFacade	AprehensionNomencladorFacade
investigacionPenalFacade	InvestigacionPenalFacade
nomencladorInvestigacionPenalFacade	NomencladorInvestigacionPenalFacade
Para cada responsabilidad:	
Nombre:	incluirAcompanante(ActionEvent event)

Descripción:	Para incluir acompañantes de la detención.
Nombre:	eliminarAcompanante(ActionEvent event)
Descripción:	Para eliminar acompañantes.
Nombre:	incluirDroga(ActionEvent event)
Descripción:	Incluye una planilla de drogas a la detención.
Nombre:	eliminarDroga(ActionEvent event)
Descripción:	Elimina una planilla de drogas.
Nombre:	incluirDiligencia (ActionEvent event)
Descripción:	Redirecciona a la página de incluir una nueva acta de detenido.
Nombre:	consultarDiligencias(ActionEvent event)
Descripción:	Redirecciona a la página de consultar diligencias de la carpeta del detenido.
Nombre:	modificarDetenido(ActionEvent event)
Descripción:	Llama a la página de modificar la persona detenida.
Nombre:	cancelarVer(ActionEvent event)
Descripción:	Cancela la acción de ver una detenido.
Nombre:	cancelar(ActionEvent event)
Descripción:	Cancela incluir un detenido.
Nombre:	cancelarModificacion(ActionEvent event)

Descripción:	Cancela modificar un detenido.
Nombre:	modificar(ActionEvent event)
Descripción:	Activa la página para que se pueda modificar el detenido.
Nombre:	actualizar(ActionEvent event)
Descripción:	Actualiza el detenido.
Nombre:	incluir(ActionEvent event)
Descripción:	Guarda la planilla de ingreso.

Nombre: ActaDetenidoVerModificarManejado	
Atributo	**Tipo**
planillaIngreso	PlanillaIngreso
actaDetenido	ActaDetenido
modificable	boolean
permiso	boolean
descripcion	String
aprehensionFacade	AprehensionFacade
Para cada responsabilidad:	
Nombre:	cerrar(ActionEvent event)

Descripción:	Cancela la acción y retorna a la página anterior.
Nombre:	modificar(ActionEvent event)
Descripción:	Activa la página para que se pueda modificar el acta.
Nombre:	actualizar(ActionEvent event)
Descripción:	Actualiza el Acta de Detenido.
Nombre:	imprimir(ActionEvent actionEvent)
Descripción:	Imprime el Acta de Detenido

Nombre: ActaDetenidoServiceImpl	
Atributo	**Tipo**
actaDetenidoDao	ActaDetenidoDao
carga	Carga
Para cada responsabilidad:	
Nombre:	salvarActaDetenido(ActaDetenido actaDetenido)
Descripción:	Salva un Acta de Detenido.
Nombre:	obtenerActasDetenidoByFecha(Integer idPlanilla,Date fechaInicio, Date fechaFin)
Descripción:	Obtiene las actas de detenido de una carpeta dado un rango de fecha.

Nombre:	actualizarActaDetenido(ActaDetenido actaDetenido)
Descripción:	Actualiza un Acta de Detenido.

Nombre: ActaDetenidoDaoImpl	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	findActaDetenidoById(Integer id)
Descripción:	Obtiene el Acta de Detenido dado un id determinado.
Nombre:	findActasByCarpeta(Integer idCarpeta)
Descripción:	Obtiene todas las actas de detenido dado una carpeta determinada.
Nombre:	obtenerActasDetenidoByFecha(Integer idPlanilla,Date fechaInicio, Date fechaFin)
Descripción:	Obtiene todas las actas de detenido dado una carpeta y un rango de fechas determinado.

Entidades persistentes y diagrama entidad relación.

La persistencia en la aplicación se maneja con la ayuda del framework Hibernate para el cual es necesario realizar un mapeo de cada una de las entidades persistentes con cada una de las tablas correspondientes en la base de datos. Las clases persistentes del subsistema estarán en el paquete domain, y además de las clases vistas anteriormente, se encuentran también las demás complementarias:

los nomencladores de aprehensión, la clase Reten y la clase CentroAtencionRecluso. A continuación se muestra el diagrama de clases persistentes seguido del modelo entidad relación correspondiente.

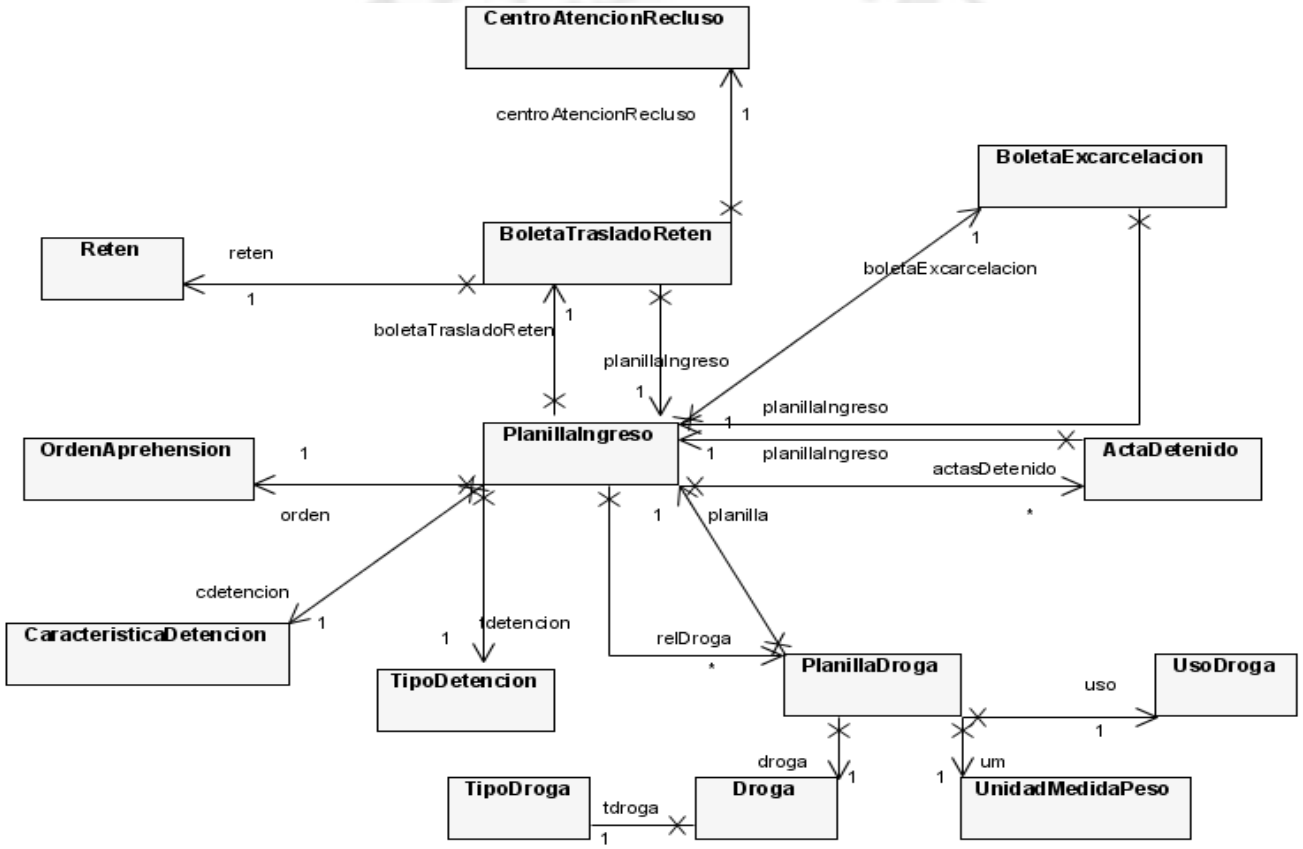


Fig. 2-27 Diagrama de Clases Persistentes

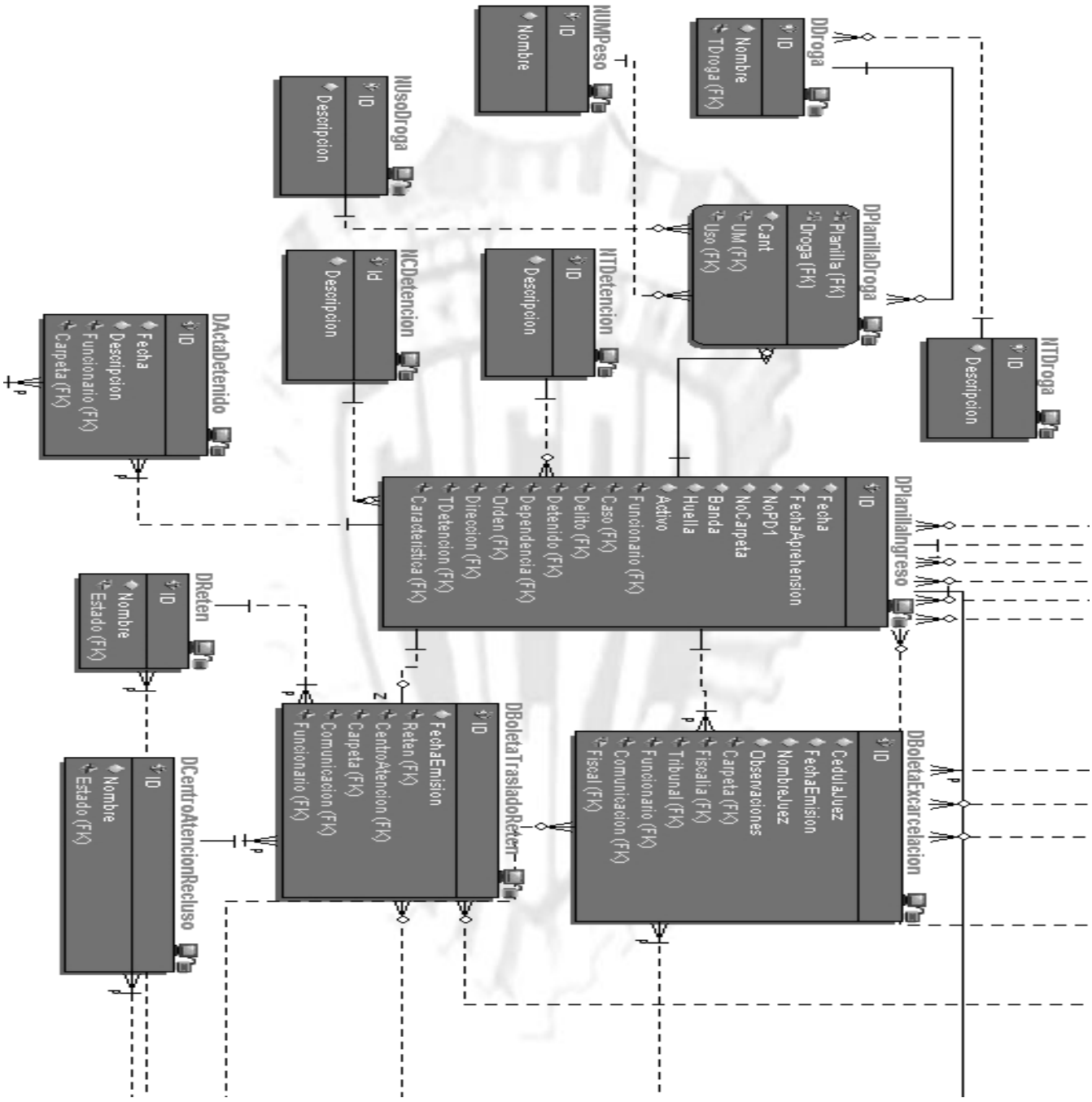


Fig. 2-28 Diagrama Entidad Relación.

2.6 Conclusiones

En este capítulo se realizó el análisis y el diseño de todo el Subsistema de Aprehensión del SIIPOL, teniendo en cuenta las descripciones de los casos de uso como principal entregable de entrada y obteniendo los diagramas de clases del análisis y los diagramas de clases del diseño como resultado. Para estos últimos se tuvo en cuenta principalmente las facilidades que los frameworks aportaban y los patrones de diseño que sin forzar al programador ayudan a la solución propuesta; finalizando con el diagrama entidad relación. Además como se pudo apreciar en las entidades que se manejan no hay grandes volúmenes de información salvo en la entidad PlanillaIngreso que es la responsable de almacenar todo lo que ocurre en el departamento.

CAPÍTULO 3: IMPLEMENTACIÓN

3.1 Introducción

En el presente capítulo se explicarán los temas relacionados con la implementación del Subsistema Aprehensión, iniciando con una breve síntesis de algunos de los componentes que fueron desarrollados por la arquitectura, el uso de los frameworks de desarrollo, la implementación de las interfaces de usuario, así como el tratamiento de excepciones. Además del diagrama de despliegue y el diagrama de componentes pertenecientes al subsistema.

3.2 Reseña

La aplicación ha sido desarrollada en varios paquetes de implementación, el paquete domain, que es donde se encuentran las entidades del dominio; el paquete dao, donde se encuentran las clases responsables del acceso a datos; el paquete service, donde están ubicadas las clases que son responsables de los servicios; el paquete fachada, donde se encuentran las fachadas del subsistema (la fachada de aprehensión y la fachada de los nomencladores de aprehensión); cada uno de estos paquetes lleva dentro un paquete impl que es el paquete donde se almacenan las implementaciones de las respectivas interfaces de cada uno de los paquetes mencionados. Además está el paquete web el cual tiene todo lo relacionado con la parte web de la aplicación, principalmente con la relacionada con el framework JSF. Este paquete está estructurado en varios paquetes: bean, con los beans de respaldo; convertidor, con los convertidores desarrollados para JSF, bundle, donde se guardan los ficheros .properties que poseen los mensajes y los textos de cada una de las páginas; y el paquete jsfconfig donde se guardan los ficheros de configuración de JSF. El subsistema cuenta además con el paquete config, donde se ubican todos los ficheros .xml responsables de las configuraciones de los frameworks Spring o Hibernate, o sea, ahí se encuentran los ficheros de configuración de las transacciones, los de los beans de spring, los ficheros de mapeo de Hibernate, los de los mapeos para los procedimientos almacenados

que se usan dentro de la aplicación buscando una mayor optimización del potente gestor de bases de datos Oracle.

Gracias a la ayuda del framework AJAX4JSF, la aplicación ha podido ser desarrollada de manera interactiva, apartándose bastante de las llamadas web tradicionales, y haciendo gran uso de las peticiones asíncronas brindadas por AJAX.

Convirtiéndose prácticamente en una aplicación de escritorio, no existe la navegación de una página a otra, ya que la aplicación está desarrollada para que solo haya una página, la cual funcionará como escritorio de trabajo e irá cambiando parte de su contenido dependiendo de la funcionalidad que sea requerida por el usuario.

En la capa de presentación se hace un uso de algunos componentes de la arquitectura como el cicpc:tiempo, y el cicpc:datascroller, el primero para la fecha y el tiempo, el segundo para paginar los datos; además de que para la mayoría de las validaciones se usan las validaciones por CSS brindadas por la arquitectura, que no es más que definir el tipo de validación que se necesita para un determinado componente, en el styleClass, y llamarla cuando se active alguna acción específica, se valida tanto en el cliente, con la ayuda de un framework de javascript, como desde el servidor.

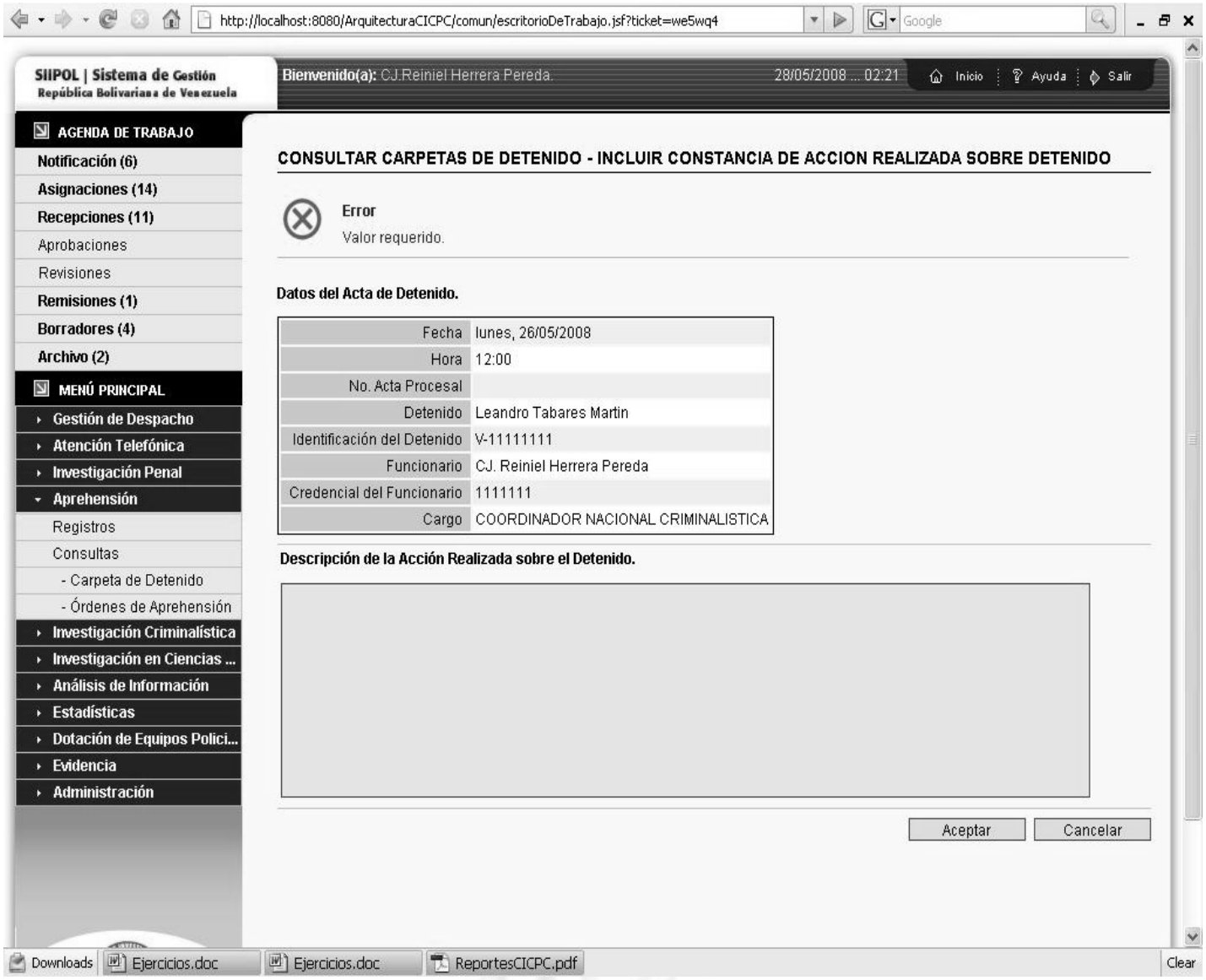


Fig. 3-1 Ejemplo de validaciones por CSS y tratamiento de errores.

Las transacciones son tratadas por Spring por el módulo AOP, el cual se centra en la programación orientada a aspectos, simplemente se declara en el fichero applicationContext-transaction.xml el método y

el tipo de transacción que tendrá. En la aplicación, para una mayor integridad de los datos, se han declarado transaccionales todos los métodos de la fachada.

Otro de los aspectos medulares de la aplicación es el acceso a los datos de la base datos, Hibernate brinda buen soporte para esto brindando la interfaz Criteria y Query, o accediendo a los objetos a través del árbol de objetos, pero no se puede dejar de mencionar el HQL (Hibernate Query Languge), que es el lenguaje de consultas brindado por Hibernate para recuperar objetos desde la base de datos; a pesar de esto en muchas ocasiones en la aplicación se hace uso de los procedimientos almacenados del gestor de base datos buscando un mayor rendimiento de la aplicación, para esto en la aplicación existe una clase desarrollada por la arquitectura que se encarga de la integración con los procedimientos almacenados, GenericProcedure, a esta clase se le especifican dos aspectos fundamentales: en la construcción se le pasa como parámetros, el nombre del cursor, el nombre del procedimiento, una referencia al datasource, una referencia al objeto que se encargara de especificar cómo se mapeará la entidades que se devuelven por el procedimiento, además de la especificación de todos los parámetros que recibirán del procedimiento, para después en el método execute() recibir sus valores y poder devolver el resultado pertinente como se muestra a continuación, en la clase de acceso a dato. Para las órdenes de aprehensión se ha creado una instancia de la clase GenericProcedure, para lograr una mayor eficiencia:

Map resultado = consultarOrdenAprehension.execute(parametros);

lista= (List<OrdenAprehension>) resultado.get(consultarOrdenAprehension.getNombreCursor());

El objeto consultarOrdenAprehension es una instancia de la clase GenericProcedure que es atributo de la clase OrdenAprehensionDaoImpl.

A continuación se muestra el diagrama de paquetes de cómo está estructurado el subsistema y una serie de diagramas de componentes que muestran su estructura interna.

3.3 Diagrama de paquetes y componentes.

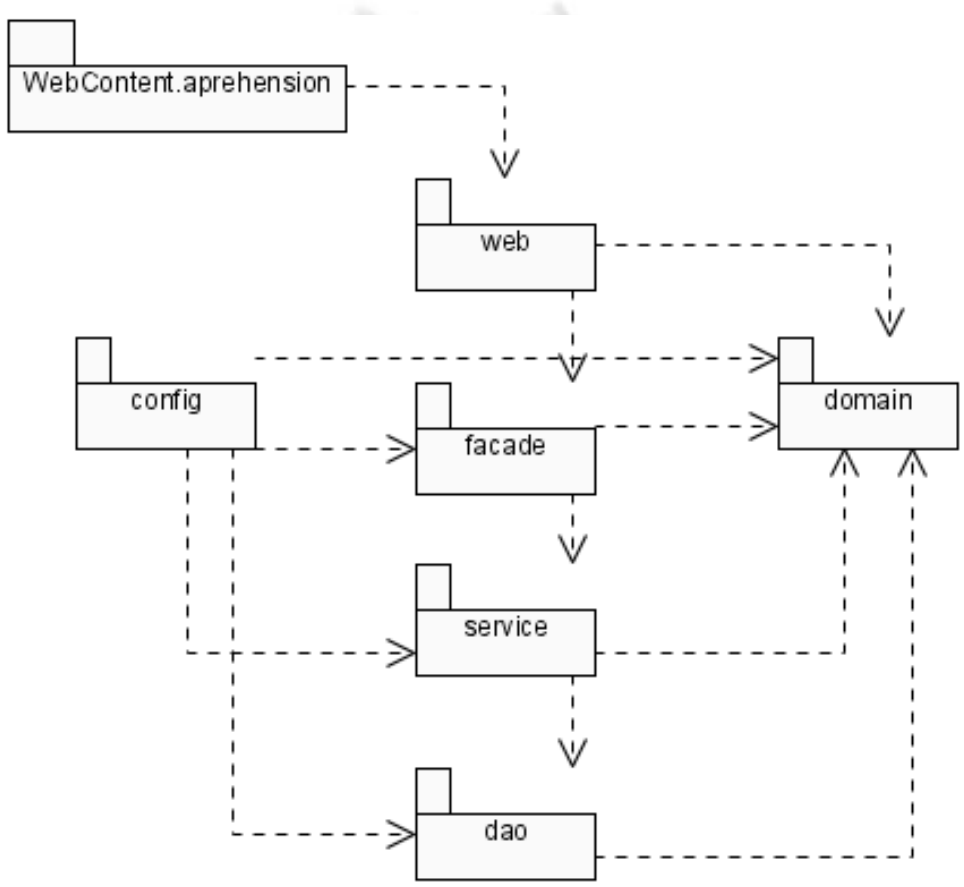


Fig. 3-2 Diagrama de Paquetes del Subsistema.

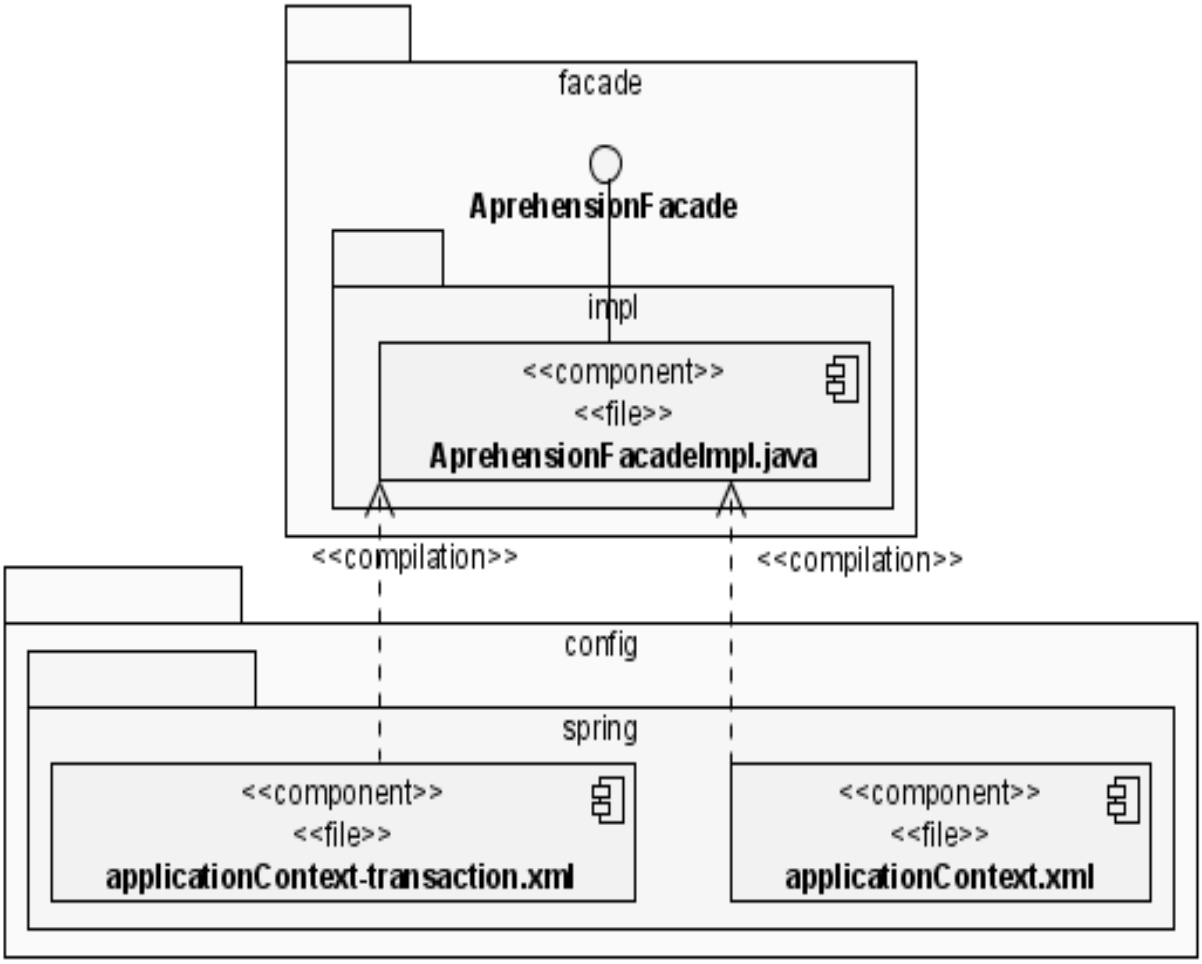


Fig. 3-3 Paquete facade con los componentes del paquete config relacionados.

Como se aprecia en la figura en el paquete facade que expone la interfaz AprehensionFacade con su respectiva implementación, además el paquete config donde aparecen los componentes applicationContext-transaction.xml donde se especifican los criterios para las transacciones y el aplicationContext.xml el cual almacena la configuración de los beans de Spring de todo el subsistema.

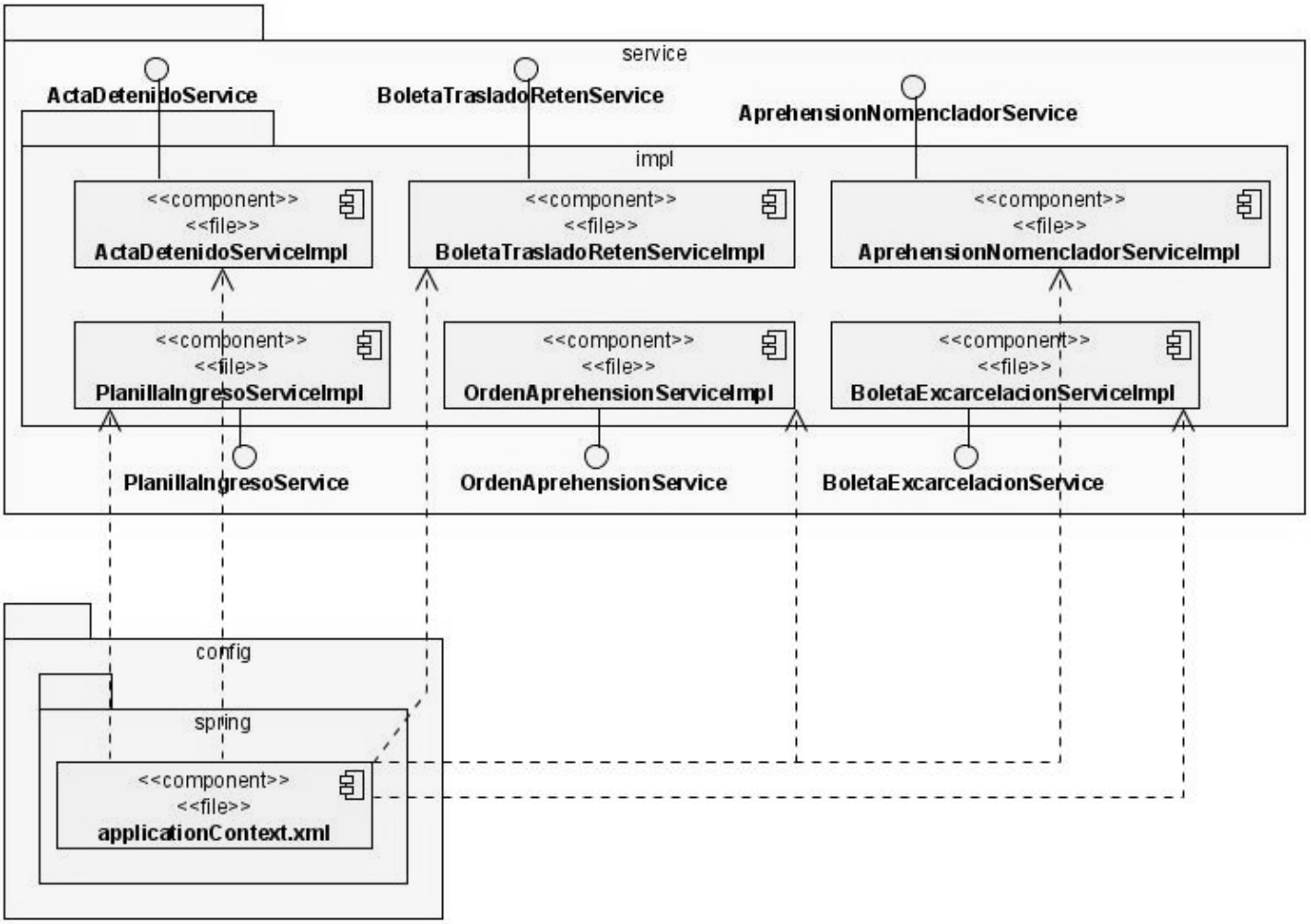


Fig. 3-4 Paquete service y su relación con el paquete config.

Aquí se muestran cada una de las clases del paquete service con sus respectivas implementaciones, al igual que los componentes del paquete facade tienen relación con el componente applicatioContext.xml pues como se explicaba anteriormente ahí se almacenan las configuraciones de los beans de Spring.

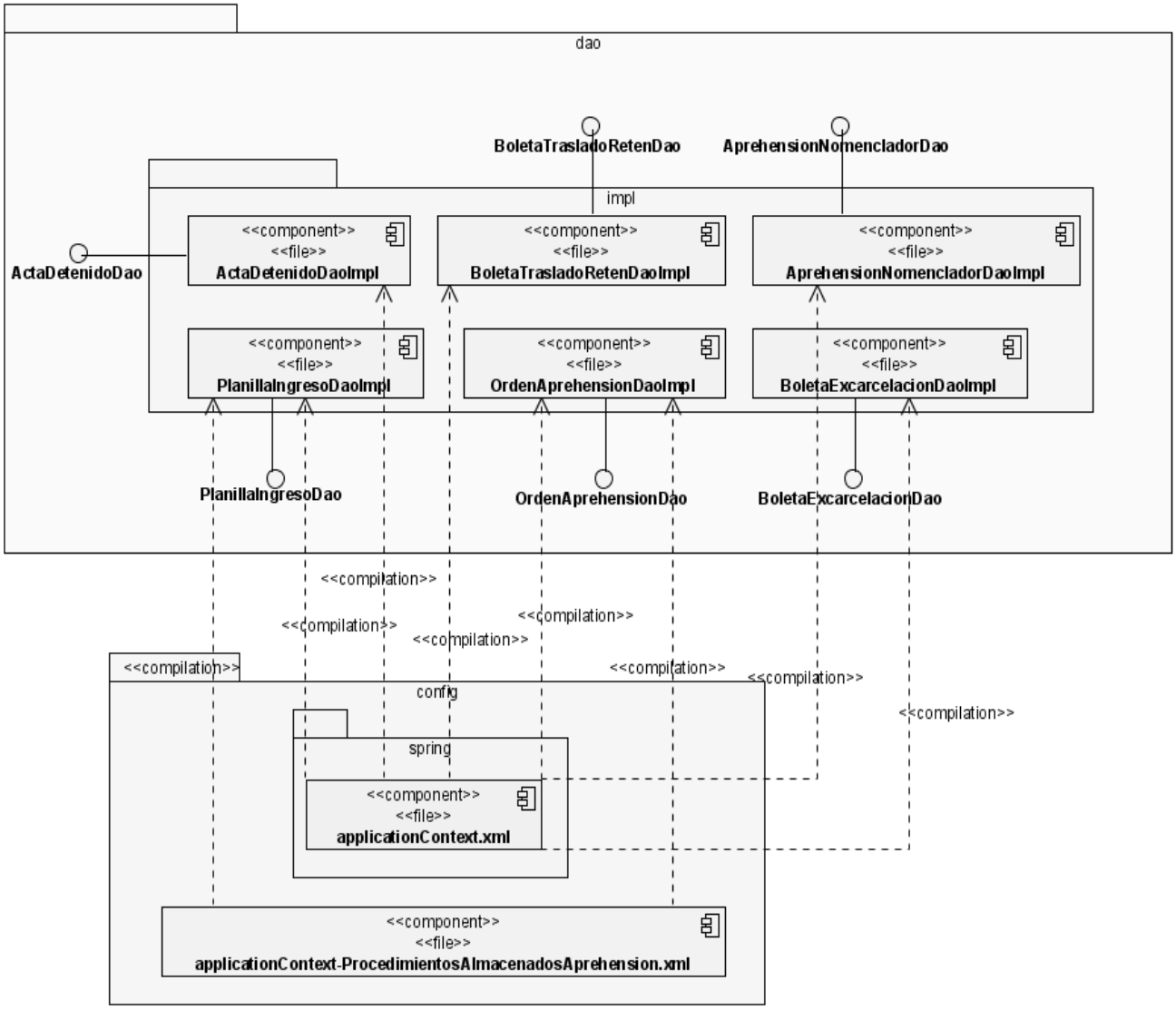


Fig. 3-5 Paquete acceso a datos y su relación con el paquete config.

Este paquete tiene de particular, que se relaciona con el componente applicationContext-ProcedimientosAlmacenadosAprehension.xml. Dicho componente es el responsable de la configuración de todos los procedimientos almacenados usados en el subsistema, tanto para la configuración del GenericProcedure como del mapeador.

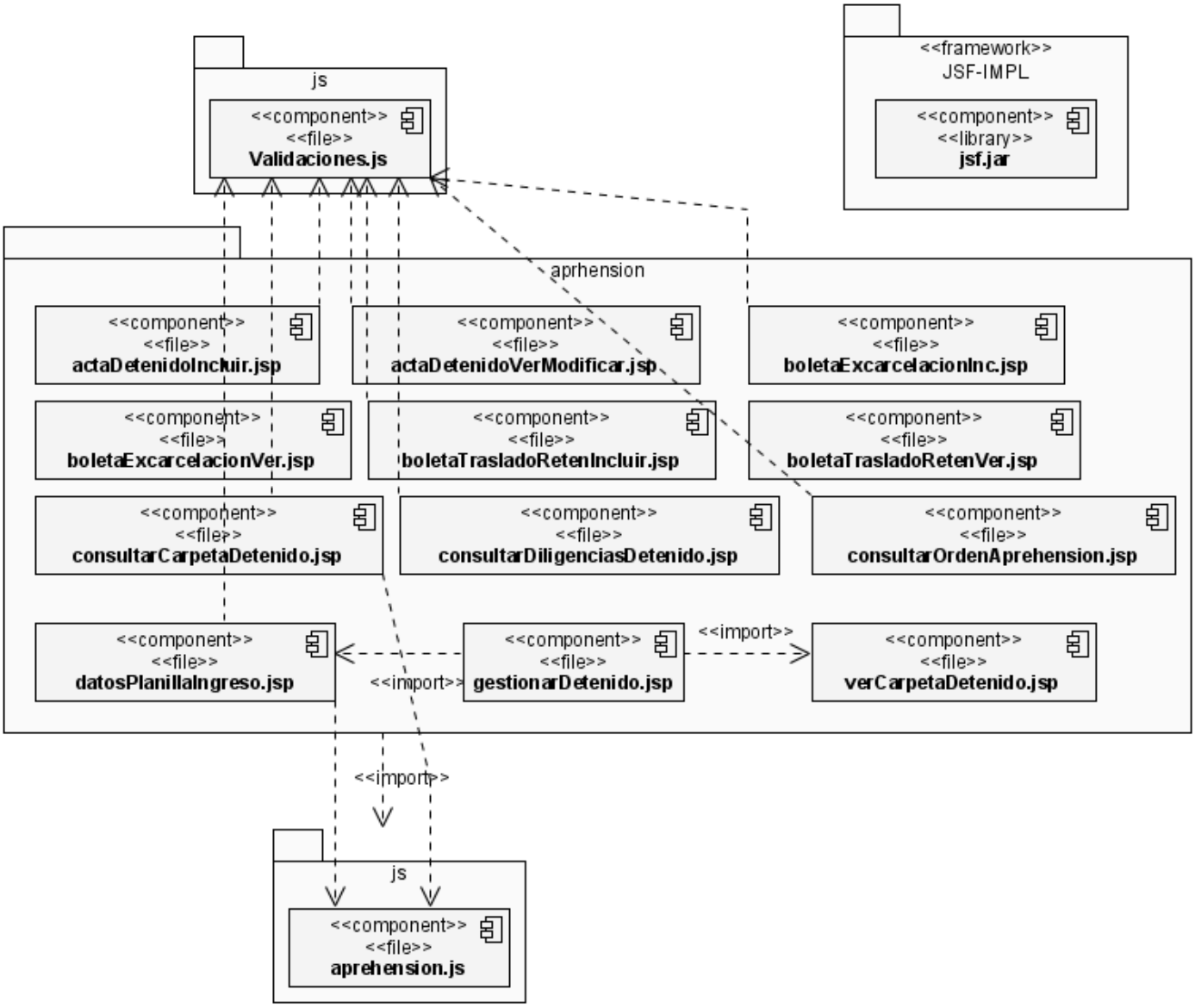


Fig. 3-6 Paquete de las paginas .jsp

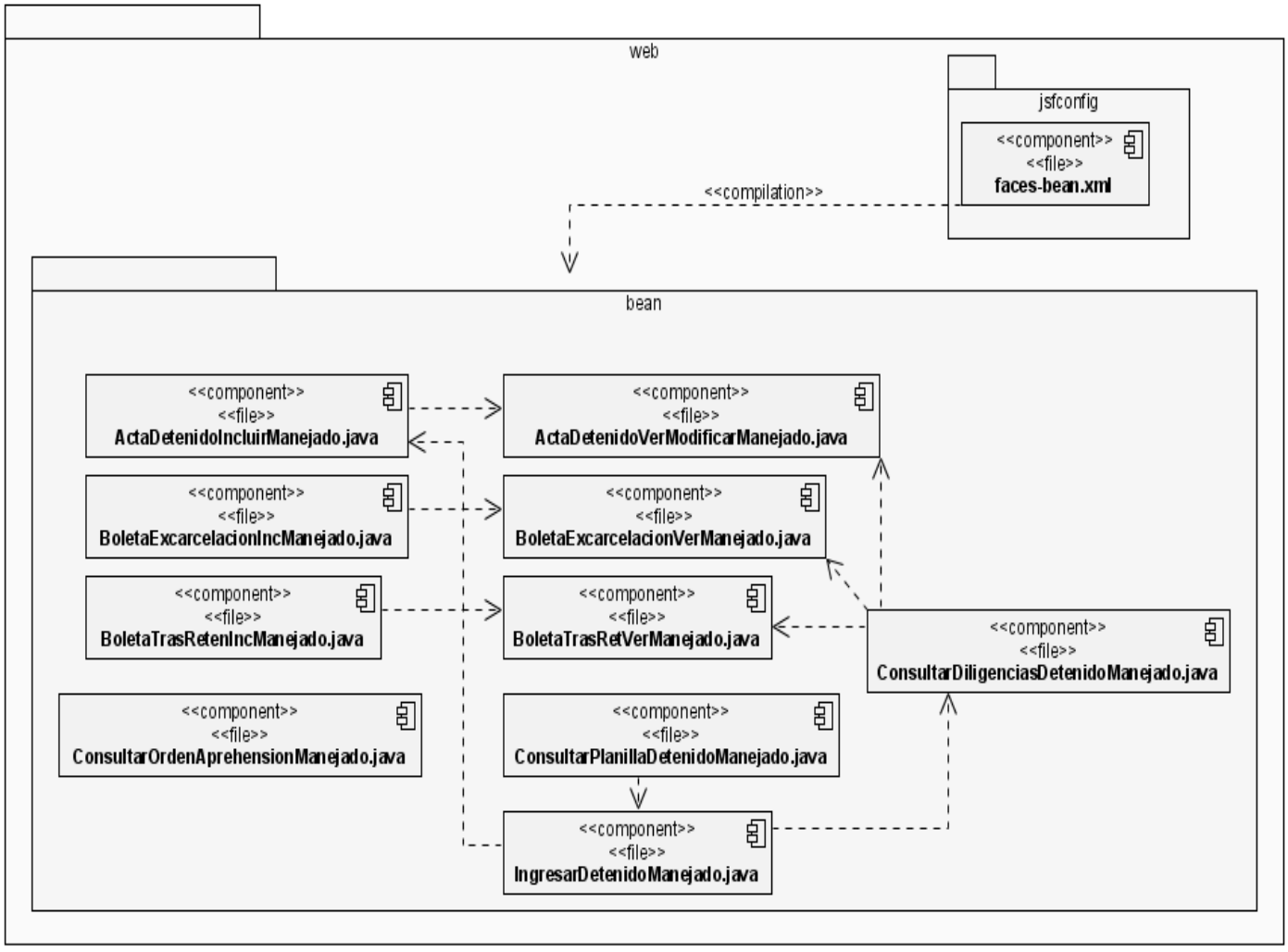


Fig. 3-7 Paquete de web con sus beans de respaldo.

En las dos figuras mostradas anteriormente se hace un resumen de la capa de presentación del subsistema el cual está formado por las páginas .jsp dentro del paquete WebContent.aprehension y los beans de respaldo asociados a cada una de las páginas dentro del paquete bean en el paquete web.

Finalmente la aplicación queda distribuida del siguiente modo:

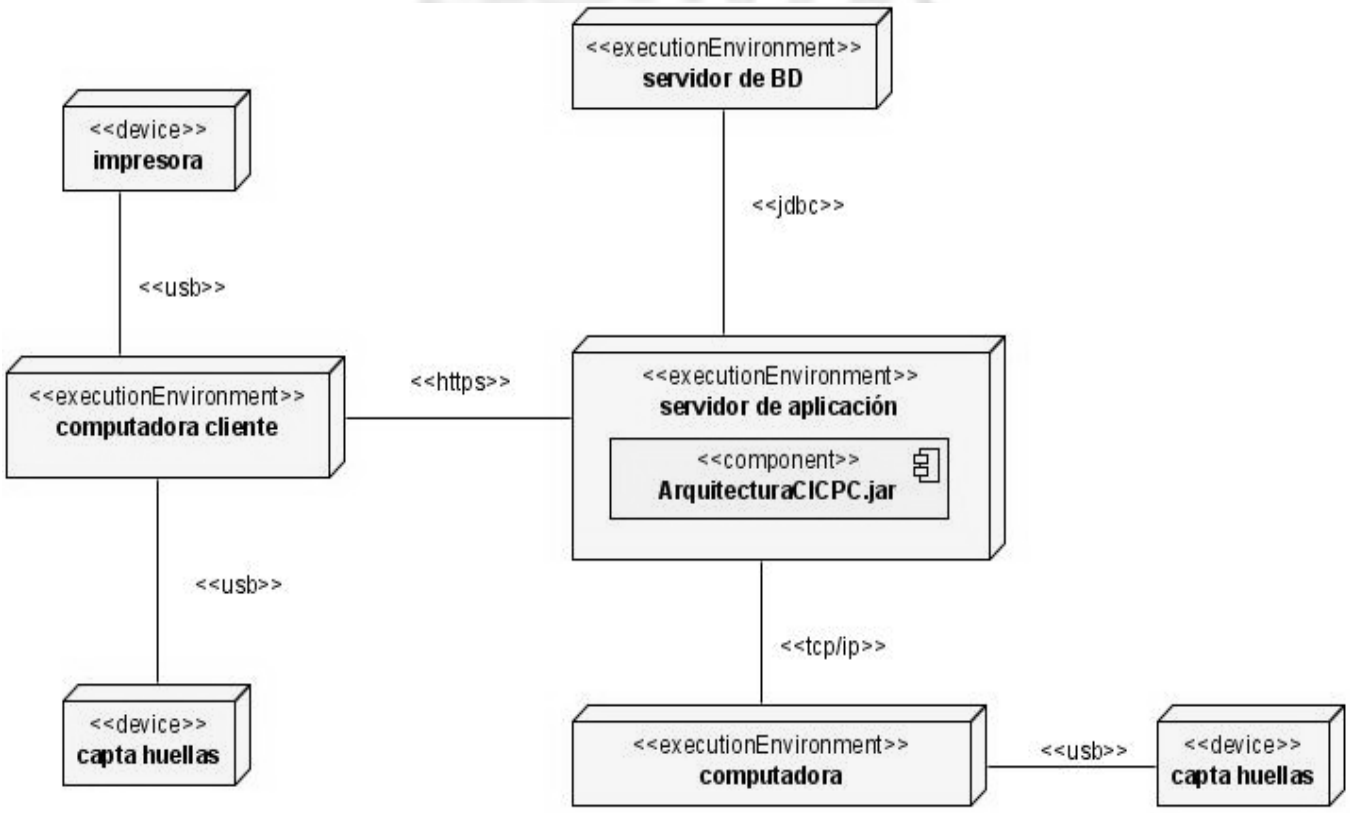


Fig. 3-8 Diagrama de Despliegue de la aplicación.

3.4 Conclusiones

Este capítulo estuvo centrado principalmente en cómo se implementaron algunas de las principales características del sistema, comenzando por la capa de presentación enfatizando en cómo se le daba funcionalidad asincrónica a la aplicación, y las validaciones de datos corruptos así como otros temas vinculados con las transacciones y el acceso a los datos. Finaliza presentando algunos de los diagramas de componentes, mostrando de alguna manera como queda estructurada la aplicación.

CONCLUSIONES

La investigación recoge todo proceso de desarrollo del Módulo Aprehensión del nuevo sistema SIIPOL, iniciando por el análisis profundo de las tecnologías actuales, del cual se obtuvo la propuesta de solución; dando paso a un estudio detallado de cada uno de los procesos que se llevan a cabo dentro del departamento de Aprehensión, teniendo en cuenta cada uno de los artefactos generados al aplicarle Ingeniería de Requerimientos al Departamento de Aprehensión.

Basado en la metodología de desarrollo RUP, a lo largo de todo el trabajo se transita por los flujos de trabajo de análisis y diseño e implementación, generando en cada flujo los principales artefactos que se necesitaban para el desarrollo, logrando obtener una aplicación funcional que responde a cada uno de los requerimientos obtenidos.

RECOMENDACIONES

- Realizar un proceso de refactorización en todos los casos de uso ajustándose también a las demás factorizaciones ejecutadas por los demás módulos del SIIPOL.
- Que todos los casos de uso pasen por el flujo de trabajo de pruebas, ya que a pesar de habérsele hecho pruebas unitarias a cada uno, no han podido ser probados con grandes volúmenes de datos en la base de datos y tampoco con una gran cantidad de usuarios conectados concurrentemente.
- Iniciar una investigación a nivel nacional, de cómo son los procesos policiales que se llevan a cabo, para, de ser posible desarrollar una herramienta que apoye en las soluciones a cada uno de los hechos delictivos, buscando aportar un poco más al medio social y ganar en experiencia acerca de los sistemas policiales.
- Tener en cuenta este trabajo para proyectos futuros, siempre y cuando se cumpla con las normas de confidencialidad establecidas.

REFERENCIAS BIBLIOGRÁFICAS

1. **Yunexis Rodríguez Baryolo, Miguel Ángel Monagas Reyes.** *Trabajo de Diploma: Ingeniería de Requerimientos del proceso de Criminalística del CICPC.* Ciudad de la Habana : UCI, 2007.

2. **Mapas Digitales S.A.** Dmapas. *Dmapas.* [En línea] 2008. [Consultado el: 10 de enero de 2008.] http://dmapas.cl/productos_stegpol.htm.

3. **Molpeceres, Alberto.** Procesos de desarrollo: RUP, XP y FDD. *Willi.net.* [En línea] 2002. [Citado el: 10 de enero de 2008.] http://www.willydev.net/descargas/articulos/general/cualxpfddrup.PDF.

4. **Kruchten, Philippe.** *Rational Unified Process, An Introduction, Third Edition.* 2003.

5. **Penadés, Patricio Letelier y Mª Carmen.** *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* España : Universidad Politécnica de Valencia, 2006.

6. **María A. Mendoza Sanchez. Ing. Informático, Microsoft Certified Professional ,Analísta y Desarrolladora - TeamSoft.** *Metodologías De Desarrollo De Software.* Peru : s.n., 2004.

7. **James Rumbaugh, Ivar Jacobson, Grady Booch.** *El lenguaje Unificado de Modelado. Manual de Referencia.* s.l. : Addison Wesley.

8. **UCI, departamento de Programación.** *Programacion 3: Conferencia 1.* 2008.

9. **UCI, Cátedra Historia de la Informática.** *Conferencia de Historia de la Informática: Historia de los lenguajes de programación.* Ciudad de la Habana : s.n., 2008.

10. **Programación en castellano.** PHP en Castellano. *PHP en Castellano.* [En línea] 1999-2008. [Citado el: 17 de Enero de 2008.] http://www.programacion.net/php/.

11. **Jalon, Javier Garcia de.** *Aprenda Java como si estuviera en primero.* San Sebastian : Escuela Superior de Ingenieros Industriales, 2000.

12. **Eclipse.** Eclipse. *Eclipse.* [En línea] 2008. [Consultado el: 11 de febrero de 2008.] http://www.eclipse.org/.

13. **NetBeans.** NetBeans. *NetBeans.* [En línea] 2008. [Consultado el: 12 de febrero de 2008.] http://www.netbeans.org.

14. Sitio Web oficial del producto Visual-Paradigm. *Sitio Web oficial del producto Visual-Paradigm.* [En línea] 2008. [Consultado el: 12 de enero de 2008.] http://www.visual-paradigm.com/product/vpuml/.

15. **IBM.** IBM. *IBM.* [En línea] 2008. [Citado el: 10 de Febrero de 2008.] http://www.ibm.com/es/es/.

16. **Markiewicz, Marcus Eduardo y de Lucena, Carlos J.P.** Crossroads The ACM Student Magazine. *Crossroads The ACM Student Magazine.* [En línea] [Consultado el: 20 de febrero de 2008.] http://www.acm.org/crossroads/espanol/xrds7-4/frameworks.html.

17. **David Geary, Cay Horstmann.** *Core JavaServer™ Faces, Second Edition.* s.l. : Prentice Hall, 2007.

18. *Conferencias de ISW.* **UCI, departamento de Ingeniería de Software.** curso escolar 2007-2008.

19. Adictos al trabajo. *Adictos al trabajo.* [En línea] 9 de Abril de 2007. [Consultado el: 10 de diciembre de 2007.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=Ajax4Jsf.

20. **Apache Software Foundation .** The Apache Software Foundation. *The Apache Software Foundation.* [En línea] 2000-2008. [Citado el: 13 de Marzo de 2008.] http://struts.apache.org/.

21. **Walls, Craig y Breidenbach, Ryan.** *Spring in Action, Second Edition.* s.l. : Manning Publications Co., 2008. 1-933988-13-4.

22. **Red Hat Inc.** SeamFramework.org. *SeamFramework.org.* [En línea] 2008. [Consultado el: 15 de febrero de 2008.] http://www.seamframework.org/.

23. **González, Héctor Suárez.** *Manual Hibernate.* s.l. : Java Hispano, 2003.

24. **CHRISTIAN BAUER, GAVIN KING.** *Java Persistence with Hibernate.* s.l. : Manning Publications Co., 2007.

25. **Apache Software Foundation .** Ibatis. *Ibatis.* [En línea] 2006-2007. [Consultado el: 20 de Enero de 2008.] http://ibatis.apache.org/.

26. **Adaptive Path Inc.** Adaptive Path. *Adaptive Path.* [En línea] 2008. [Consultado el: 20 de Enero de 2008.] http://adaptivepath.com/ideas/essays/archives/000385.php.

27. **Walls, Craig y Breidenbach, Ryan.** *Spring in Action.* s.l. : Manning Publications Co., 2005. 1-932394-35-4.

28. **García, Aitor.** *Guía del autoestopista a Hibernate.* 2003.

29. **Sun Microsystems, Inc.** Sun Microsystems. *Sun Microsystems.* [En línea] 1994-2008. [Consultado el: 5 de marzo de 2008.] http://java.sun.com/.

30. **Scrum Alliance, Inc.** ScrumAlliance. [En línea] [Citado el: 11 de febrero de 2008.] http://www.scrumalliance.org/.

31. **Jacobson, I. y Booch, G. y Rumbaugh, J.** *"El Proceso Unificado de Desarrollo de software".* s.l. : Addison Wesley, 2000. 84-7829-036-2.

32. **Allen, Dan.** *Seam in Action.* s.l. : Manning Publications, 2008 .

33. **Advanced Development Methods, Inc (ADM).** SCRUM. [En línea] 2008. [Consultado el: 11 de Febrero de 2008.] http://www.controlchaos.com/.

BIBLIOGRAFÍA

- **Agile Modeling and eXtreme Programming (XP). Agile Modeling.** [En línea] 3 de Marzo de 2007. http://www.agilemodeling.com/essays/agileModelingXP.htm.
- **Allen, Dan.** *Seam in Action.* s.l. : Manning Publications, 2008 .
- **BAUER, CHRISTIAN y KING, GAVIN.** *Java Persistence with Hibernate.* s.l. : Manning Publications Co., 2007.
- **Duvall, Paul, Matyas, Steve y Glover, Andrew**. *Continuous Integratio, 2007,* Addison Wesley,
- **Falkner, Jayson y Jones, Kevin,** *Servlets and JavaServer Pages***.** 2004, Addison Wesley**.**
- **Geary , David y Horstmann , Cay.** *Core JavaServer™ Faces, Second Edition.* s.l. : Prentice Hall, 2007.
- **González, Héctor Suárez.** *Manual Hibernate.* s.l. : Java Hispano, 2003.
- **Grame, D. y Pascarello, E. .***Ajax in Action.*2006. Manning.
- **Husted, Ted; Dumoulin, Cedric; Franciscus, G. y Winterfeldt, D.**, *Struts in Action,* 2003, Manning.
- **Jacobson, I. y Booch, G. y Rumbaugh, J.** *"El Proceso Unificado de Desarrollo de software".* s.l. : Addison Wesley, 2000. 84-7829-036-2.
- **JACOBSON, Ivar; BOOCH, Grady, RUMBAUGH, James,** *El Proceso Unificado de Desarrollo de Software***.**2000**.**
- **Larman, Craig**. *UML y Patrones. Introducción al análisis y diseño orientado a objetos*, 2003, Prentice Hall
- **Mann , Kito D..** *Java Server Faces*.2005.Manning.
- **Massol, V., Husted, T.,** *JUnit in Action,*2004, Manning.
- **Patrick Peak, nick Heudecker**. *Hibernate Quickly*. 2006.Manning
- **PRESSMAN, R. S**. *"Ingeniería del Software. Un Enfoque Práctico".* 1998.

- **Pressman, Roger;** *Ingeniería de software. Un enfoque práctico***.** 2002**.** McGraw.Hill/Interamericana de España.
- **Rational Unified Process***. Ayuda del RUP. Suite del Rational* 2003**.**
- **Richardson, Chris,** *POJOs in Action,*2006, Manning.
- **RUMBAUGH, James, JACOBSON, Ivar; BOOCH, Grady,** *El lenguaje unificado de modelado***,** 2000**,** Addison Wesley.
- **Shore, James y Warden, Shane**. *The Art of Agile Development*, 2007, O´Reilly.
- **W.S.HUMPHREY**. "*Introduction to the Personal* Software ***Process****"*. Addison-Wesley, 1997,
- **Walls, Craig y Breidenbach , Ryan.** *Spring in Action,*2005,Manning.
- **Walls, Craig y Breidenbach, Ryan.** *Spring in Action, Second Edition.* s.l. : Manning Publications Co., 2008. 1-933988-13-4.

GLOSARIO

Acegi: Acegi Scurity System provee de las capacidades de autorización y autenticación a los proyectos desarrollados con el uso del framework Spring.

Acta de Detenido: Documento que es emitido por un funcionario cuando realiza alguna acción sobre un detenido.

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

AJAX: Java Script asíncrono y XML es la unión de varias tecnologías que juntas pueden lograr cosas realmente impresionantes como GoogleMaps, Gmail, el Outlook Web Access o algunas otras aplicaciones muy conocidas: **AJAX, en resumen, es el acrónimo para Asynchronous JavaScript + XML** y el concepto es: Cargar y renderizar una página, luego mantenerse en esa página mientras scripts y rutinas van al servidor buscando, en background, los datos que son usados para actualizar la página solo re-renderizando la página y mostrando u ocultando porciones de la misma. (26)

Ajax4JSF: Ajax4jsf es una librería open source que se integra totalmente en la arquitectura de JSF y extiende la funcionalidad de sus etiquetas dotándolas con tecnología Ajax.

AOP: Aspect-Oriented Programming, traducido al español: Programación Orientada a Aspectos, paradigma de programación que separa la lógica del negocio de aspectos de servicios, tales como la seguridad, las transacciones, entre otros. (21)

Bean: En el lenguaje Java es un componente software reutilizable que evita programar los distintos componentes uno a uno. Existen con la finalidad de ahorrar tiempo al programar.

Boleta de Excarcelación: Boleta que se emite por un tribunal para liberar a un detenido.

Boleta de Traslado al Retén: Boleta que es emitida por el tribunal para enviar a un detenido a un retén.

CICPC: Cuerpo de Investigaciones Científicas, Penales y Criminalísticas, es el organismo venezolano responsable de llevar a cabo las investigaciones asociadas a cada uno de los hechos delictivos.

DAO: Data Access Object, traducido al español: Objeto de Acceso a Datos, es un Patrón de diseño de clases en ingeniería de software que permite a quien lo aplique suministrar una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos.

Departamento de Aprehensión: Es el área dentro del CICPC responsable de las personas cuando son detenidas hasta que son liberados o enviados al retén.

EJB: Enterprise Java Bean, especificación de Java publicada por Sun Microsystems en marzo de 1998, extendía la noción de los componentes de Java del lado del servidor, brindando una mayor cantidad de servicios. (27)

Framework: Los frameworks orientados al objeto (llámense simplemente frameworks) son la piedra angular de la moderna ingeniería del software. El desarrollo del framework está ganando rápidamente la aceptación debido a su capacidad para promover la reutilización del código del diseño y el código fuente (source code). Los frameworks son los Generadores de Aplicación que se relacionan directamente con un dominio específico, es decir, con una familia de problemas relacionados. (16)

GUI: Graphic User Interface.

Hibernate: Hibernate ofrece la *Persistencia Relacional para Java*, que para los no iniciados, proporciona unas muy buenas maneras para la persistencia de sus objetos de Java a y desde una base de datos subyacente. Más que ensuciar con SQL tus objetos y convertir consultas a y desde los objetos de primera magnitud, Hibernate puede preocuparse de todo ese maremágnum por ti. Tú utilizas solamente a los objetos, Hibernate se preocupa del SQL y de que las cosas terminan en la tabla correcta. (28)

HQL: Hibernate Query Language, lenguaje de consultas de Hibernate, similar al SQL pero trabaja a nivel de objetos no de tablas.

IoC: Inversion of Control, traducido al español Inversión de Control, técnica usada por Spring donde en lugar de los objetos buscar sus dependencias, el propio contenedor se responsabiliza por buscarlas. (27)

JNDI: (nombre de Java Naming and Directory Interface) Nombre utilizado para acceder a un recurso que se ha registrado en el servicio de nombres JNDI. (29)

JSF: Java Server Faces, framework de interfaz de usuario para aplicaciones web desarrolladas en java.

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

RUP: Metodología de desarrollo de software. Es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del 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. (18)

SIIPOL: Sistema informático que actualmente se encuentra desplegado en algunas de las áreas del CICPC, las siglas responden a Sistema Integrado de Información Policial y con la nueva restructuración pasa a nombrarse Sistema de Investigación e Información Policial.

SOA: Service-oriented architecture, traducido al español: Arquitectura Orientada a Servicios.

Spring: Es un framework open source, creado por Rod Johnson y descrito en su libro *Expert One-on-One: J2EE Design and Development*. (27)

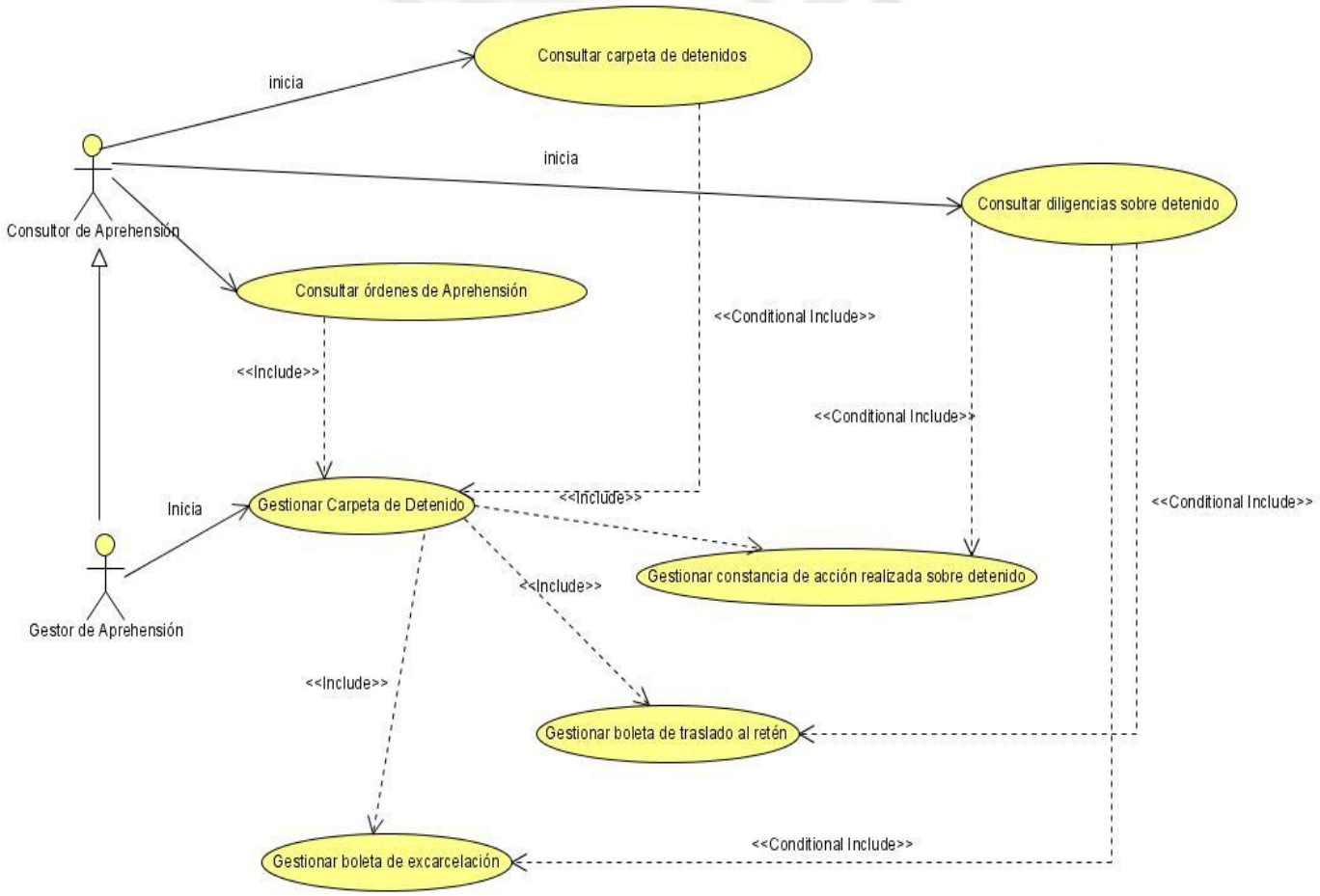
Subsistema de Aprehensión o Módulo de Aprehensión: Es la parte del proyecto de software de CICPC encargada de brindarle soporte a los procesos que se realizan dentro del Departamento de Aprehensión.

Threads: Hilos de procesamiento.

UML: Lenguaje Unificado de Modelado, es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software.

ANEXOS

Anexo 1: Diagrama de CU



Anexo 2: Ejemplo de IBATIS

Person.java

```
package examples.domain;

//imports omitidos….

public class Person {

private int id;

private String firstName;

private String lastName;

private Date birthDate;

private double weightInKilograms;

private double heightInMeters;

public int getId () {

return id;

}

public void setId (int id) {

this.id = id;

}

//…se asume que aquí se tienen otros métodos get y set.…

}
```

Person.sql

```
CREATE TABLE PERSON(

PER_ID NUMBER (5, 0) NOT NULL,

PER_FIRST_NAME VARCHAR (40) NOT NULL,

PER_LAST_NAME VARCHAR (40) NOT NULL,

PER_BIRTH_DATE DATETIME ,

PER_WEIGHT_KG NUMBER (4, 2) NOT NULL,

PER_HEIGHT_M NUMBER (4, 2) NOT NULL,

PRIMARY KEY (PER_ID))
```

Person.xml

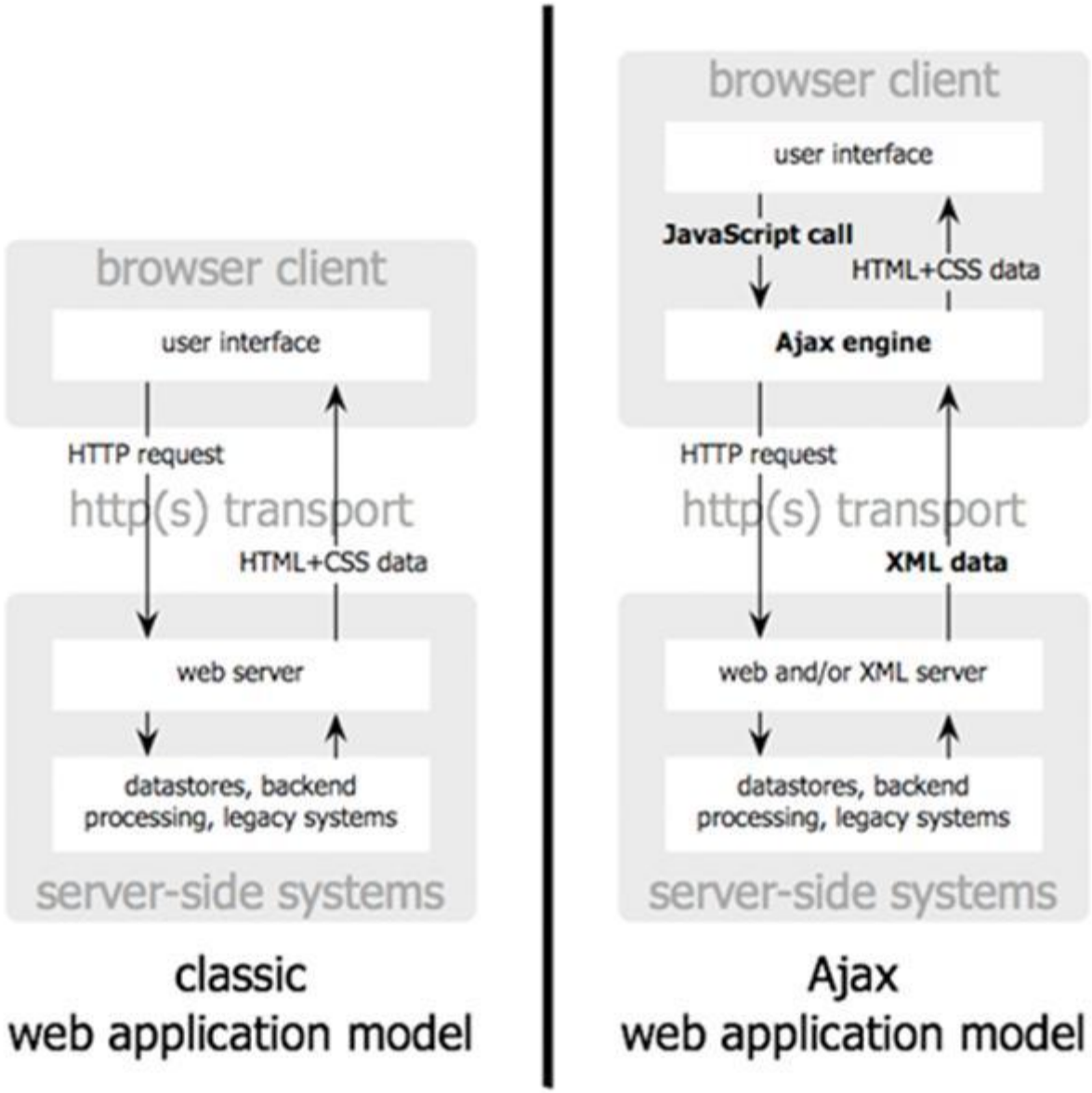
```
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE sqlMap

PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"

"http://ibatis.apache.org/dtd/sql-map-2.dtd">

<sqlMap namespace="Person">

<select id="getPerson" resultClass="examples.domain.Person">

SELECT

PER_ID as id,

PER_FIRST_NAME as firstName,
```

```
PER_LAST_NAME as lastName,

PER_BIRTH_DATE as birthDate,

PER_WEIGHT_KG as weightInKilograms,

PER_HEIGHT_M as heightInMeters

FROM PERSON

WHERE PER_ID = #value#

</select>

<insert id="insertPerson" parameterClass="examples.domain.Person">

INSERT INTO

PERSON (PER_ID, PER_FIRST_NAME, PER_LAST_NAME,

PER_BIRTH_DATE, PER_WEIGHT_KG, PER_HEIGHT_M)

VALUES (#id#, #firstName#, #lastName#,

#birthDate#, #weightInKilograms#, #heightInMeters#)

</insert>

<update id="updatePerson" parameterClass="examples.domain.Person">

UPDATE PERSON

SET PER_FIRST_NAME = #firstName#,

PER_LAST_NAME = #lastName#, PER_BIRTH_DATE = #birthDate#,

PER_WEIGHT_KG = #weightInKilograms#,
```

```
PER_HEIGHT_M = #heightInMeters#

WHERE PER_ID = #id#

</update>

<delete id="deletePerson" parameterClass="examples.domain.Person">

DELETE PERSON WHERE PER_ID = #id#

</delete>

</sqlMap>
```

```
Person person = (Person) sqlMap.queryForObject (“getPerson”, personPk);

sqlMap.update(“updatePerson”, person);

sqlMap.delete (“deletePerson”, person);
```

Anexo 3: Comparación entre las aplicaciones web tradicionales y las que tienen Ajax.



Anexo 4: Patrón MVC