UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 2

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE

INGENIERO EN CIENCIAS INFORMÁTICAS

Análisis, Diseño e Implementación del submódulo Investigación Internacional del Sistema de Investigación e Información Policial.

Autores:

Agner de la Cruz Guilarte

Yamil Ponjuan Navarro

Tutores:

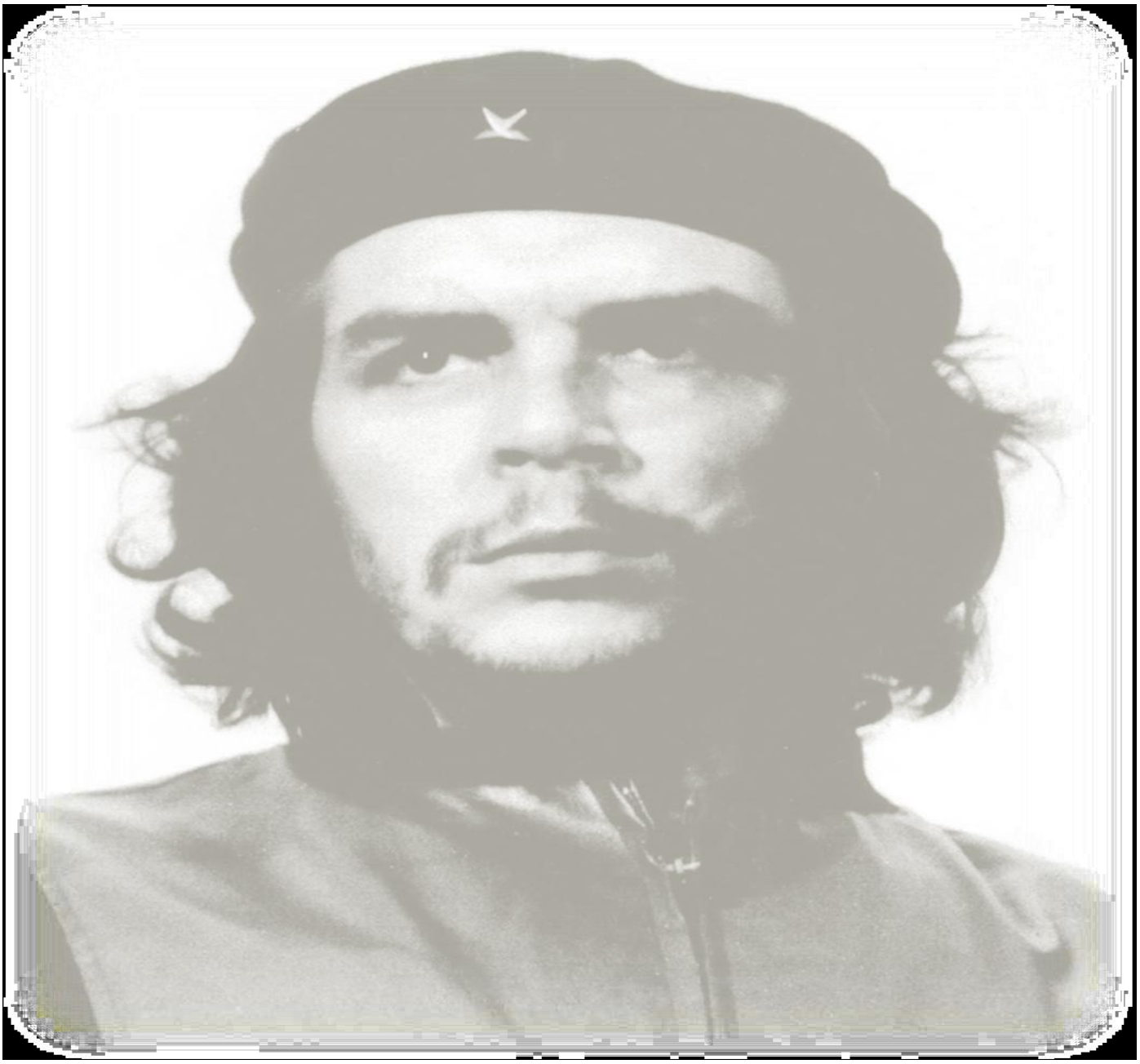
Ing. Kenny López Espinosa
Ing. Irina Cancela Nieto

Declaración de Autoría

Declaramos que _____ y _____ somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) y a la Facultad 2 para que hagan el uso que estimen pertinente con este trabajo.

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

_____	_____
Firma del Autor	Firma del Autor
Agner de la Cruz Guilarte	Yamil Ponjuan Navarro
_____	_____
Firma del Tutor	Firma del Tutor
Ing. Kenny López Espinosa	Ing. Irina Cancela Nieto



“Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio, conscientes de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte.”

AGRADECIMIENTOS

A la Revolución, a la Universidad de Ciencias Informáticas y en especial a la Facultad 2 por hacer realidad nuestro sueño de formarnos como profesionales.

A nuestro tutor Kenny López Espinosa y a las profesoras Maylín Díaz Cabrera y Yadira Marín González por estar siempre dispuestos en los momentos en que necesitamos de su apoyo.

A nuestro compañero de equipo de desarrollo Rolando Gual Pérez, a nuestros amigos, por su consagración y compromiso: al equipo de desarrollo del proyecto CICPC.

Yamil Ponjuan Navarro

A mis padres Virginia Teresa y Lorenzo por su cariño, su dedicación, su apoyo incondicional, sus sacrificios y desvelos.

A mi hermana Damarys, mi adoración, por ser mi mayor inspiración.

A mi familia inmensa por ofrecerme toda una vida llena de amor y darme su apoyo en cada momento.

A Lisandra en especial, por estos maravillosos años, por hacer más intensas mis alegrías y reducir a la mitad mis penas.

A todos mis amigos, los nuevos y los viejos, por hacer de estos años una experiencia increíble e inolvidable, no hubiera sido lo mismo de no haberlos conocido.

A mi compañero de tesis Agner por darme su apoyo en cada momento.

Agner de la Cruz Guilarte

Con un testimonio de eterno agradecimiento por el apoyo moral que desde siempre me brindaron y con el cual he logrado terminar mi carrera profesional, que es para mí la mejor de las herencias, les agradezco:

- A dios que me ha heredado el tesoro más valioso que puede dársele a un hijo "sus padres".
  A mis padres quienes sin escatimar esfuerzo alguno sacrificaron gran parte de su vida para educarme.
    - ✓ Mi madre: Que es el ser más maravilloso de todo el mundo. Gracias por el apoyo moral, el cariño y comprensión que desde niño me has brindado, por guiar mi camino y estar junto a mí en los momentos más difíciles.
    - ✓ Mi Padre: Porque desde pequeño ha sido para mí un gran hombre maravilloso al que siempre he admirado. Gracias por guiar mi vida con energía, esto ha hecho que sea lo que soy.
- A mis hermanos quienes la ilusión de su vida ha sido verme convertido en un hombre de provecho.
- A mis sobrinas, porque el simple hecho de enseñarme sus tareas y decirme: Viste, vamos igual que tú, me da cada día más fuerzas para convertirme en su ejemplo a seguir.
- A mi familia en general, a mi abuelita que la llevo en mi corazón, a Odalis, mi otra madre.
- A mis amigos Denier, Cubilla, Jose, Elpidio, Darien, Lester, Denis, Carlos, Rogelio, Jorgito, Manolo, Salmeron; Alexey: ”Los Mariachis”, a Shampo, Tito, Yadira que me ayudó muchísimo en la tesis, Tati que en 4 años me brindó su apoyo incondicional, a mi novia Maria Luisa, sin olvidar a mi amigo de 5 años y pareja de tesis “El negro”, en fin, a todas aquellas personas que estoy seguro de que HOY comparten conmigo este triunfo.

DEDICATORIA

A nuestros padres, porque este ha sido también su sueño.

RESUMEN

El Cuerpo de Investigaciones Científicas Penales y Criminalísticas (CICPC) de Venezuela, es una institución que tiene como misión garantizar la eficiencia en la investigación del delito, mediante su determinación científica, asegurando el ejercicio de la acción penal que conduzca a una sana administración de justicia.

El CICPC cuenta con el Sistema Integrado de Información Policial (SIIPOL) como aplicación informática, cuya ineficiencia, debido a la tecnología obsoleta que presenta, la demora en la actualización de la información, así como su interfaz poco amigable dificulta de manera considerable los procesos que se llevan a cabo dentro de la institución.

Se espera que con el desarrollo e implantación del submódulo Investigación Internacional perteneciente al módulo INTERPOL en el nuevo sistema, se mejore en tiempo y calidad los procesos que se llevan a cabo en la Dirección de la Policía Internacional adscrita al CICPC, la cual sirve de puente de comunicación entre las diferentes policías en el mundo, además de organizar y llevar el control del trabajo diario de cada funcionario.

El desarrollo de la solución fue guiado por las fases y artefactos que el Proceso Unificado de Desarrollo (RUP) propone y las notaciones establecidas por el Lenguaje de Modelado Unificado (UML), que han sido ajustadas a las necesidades del proyecto CICPC.

ÍNDICE

ÍNDICE DE FIGURAS.

INTRODUCCIÓN

Venezuela ha resultado al menos en una ocasión el país más violento del mundo, con mayor cantidad de víctimas por armas de fuegos, conforme a varios documentos publicados por la Unesco y la Organización de Naciones Unidas, en donde indican que el territorio venezolano supera a Colombia y Brasil, esto según cifras aportadas por el Ministerio del Popular para Interiores y Justicia y el Cuerpo de Investigaciones Científicas Penales y Criminalísticas (CICPC). La tensa situación en las calles y la corrupción en el aparato policial contribuyen a la inseguridad ciudadana y al estado de paranoia generalizados. (1)

En consecuencia, el país suramericano y su presidente Hugo Rafael Chávez Frías como parte de su proyecto para fortalecer la lucha contra la inseguridad ciudadana desde el inicio de su mandato, ordenó una medida de transformación radical al CICPC (2), institución que posee como objetivos generales: optimizar las acciones de Investigación Criminal tendentes a lograr el esclarecimiento de los hechos delictivos, capacitar el capital humano integrado a la Institución, con el fin de alcanzar un alto nivel de eficacia y eficiencia y apoyar las políticas de Estado a través de estrategias dirigidas a la reducción de los delitos en todas sus modalidades. (3)

El CICPC es una institución que tiene como misión garantizar la eficiencia en la investigación del delito, mediante su determinación científica, asegurando el ejercicio de la acción penal que conduzca a una sana administración de justicia (4) y consta dentro de su estructura organizativa con la dirección de Policía Internacional (INTERPOL), que posee como funciones: mantener el enlace con los diferentes organismos policiales del país, así como Oficinas Centrales Nacionales e Internacionales, adoptar las resoluciones de convenios internacionales acordadas por la Organización Internacional de Policía Criminal (O.I.P.C.), representar al país en las reuniones de la Asamblea General de la O.I.P.C. en los casos que así lo decida la Dirección General Nacional, mantener contacto permanente con la Oficina Nacional Identificación y Extranjería (ONIDEX) a fin de disponer de información sobre los movimientos migratorios de ciudadanos, cumplir con los parámetros legales establecidos, ante la solicitud de extradición de ciudadanos y mantener actualizados los registros policiales de venezolanos y extranjeros solicitados internacionalmente para asegurar el adecuado flujo de información. (5)

El CICPC cuenta con el Sistema Integrado de Información Policial (SIIPOL) como aplicación informática, cuya ineficiencia, debido a la tecnología obsoleta que presenta, la demora en la actualización de la información ,así como su interfaz poco amigable para los usuarios que interactúan con ella dificulta de manera considerable los procesos que se llevan a cabo dentro de la institución.

En la actualidad, todavía en algunos despachos del CICPC se trabaja con el Sistema Integrado de Información Policial, el cual no responde en gran medida a las exigencias del trabajo que es desarrollado en esta institución ,entre los que se puede mencionar la Dirección de INTERPOL la misma al no tener automatizado gran parte de sus procesos que como principal problemática trae consigo el deterioro y pérdida de la documentación con el transcurso del tiempo, lentitud en el intercambio de información entre divisiones y departamentos responsables de darle solución en un tiempo breve a los casos investigativos y por otra parte la búsqueda engorrosa de información en el Archivo Internacional debido a los grandes volúmenes existentes de la misma. (6)

Con el propósito de desarrollar un nuevo sistema, modernizado, automatizado, que permita una mejor gestión de la información, la disminución de los tiempos de respuesta de las informaciones solicitadas y cubra los procesos que se desarrollan en las diferentes áreas o departamentos del CICPC, como es el caso de INTERPOL, se firmó un contrato en el marco de las relaciones Cuba – Venezuela a través de la colaboración de los países del ALBA donde surge el SIIPOL como parte del proyecto de modernización del CICPC.

Basado en el estudio realizado del trabajo de diploma “Ingeniería de Requisitos para el módulo Interpol del SIIPOL” se conoció que a partir de la aplicación de la ingeniería de requerimientos basada en un análisis exhaustivo del funcionamiento de la institución, así como sus procesos de negocio, se generó un conjunto de requisitos funcionales y no funcionales, que resumen las capacidades y condiciones que debe cumplir el nuevo sistema para INTERPOL y que dan al traste con el antiguo SIIPOL. (6)

Planteada la situación problemática se puede enunciar el **problema a resolver** a partir de la siguiente interrogante: ¿Cómo garantizar el cumplimiento de los requisitos funcionales y no funcionales asociados al submódulo Investigación Internacional del módulo INTERPOL del SIIPOL?

El **objeto de estudio** determinado para el presente trabajo de diploma se basa en el proceso de desarrollo en Sistemas de Gestión Policial y el **campo de acción** se enmarca en el submódulo Investigación Internacional del módulo INTERPOL perteneciente al SIIPOL.

La **idea a defender** se plantea de la siguiente forma: “Con el análisis, diseño e implementación del submódulo Investigación Internacional se garantizará el cumplimiento de los requisitos funcionales y no funcionales obtenidos en la Ingeniería de Requerimientos”.

La presente investigación tiene como **Objetivo General**: Analizar, diseñar e implementar el submódulo Investigación Internacional perteneciente al módulo INTERPOL del SIIPOL.

Una vez definido el objetivo general y en aras de darle cumplimiento, se plantean los siguientes Objetivos Específicos:

- Analizar los resultados obtenidos en la aplicación de la ingeniería de requerimientos.
- Analizar, definir, estructurar y detallar mecanismos de diseño que garanticen el cumplimiento de las necesidades del submódulo Investigación Internacional perteneciente al módulo INTERPOL del SIIPOL
- Implementar el submódulo diseñado respetando la arquitectura definida para el nuevo SIIPOL.
- Integrar el submódulo implementado al SIIPOL.

Para cumplir con lo antes descrito en esta investigación se han trazado las siguientes tareas:

- Investigación acerca de otras soluciones informáticas o aplicaciones similares, fundamentalmente en aquellas que desarrollen sistemas de Gestión Policial.
- Investigación acerca de la metodología, herramientas y lenguaje a utilizar en el análisis, diseño e implementación del submódulo Investigación Internacional perteneciente al SIIPOL.

- Estudio sobre la Arquitectura del SIIPOL y el modelo de Casos de Uso que da cumplimiento a los requisitos funcionales y no funcionales asociados al módulo INTERPOL perteneciente al SIIPOL.
- Investigación de la aplicación de Patrones de Diseño en soluciones similares.
- Confección de los diagramas de clases de diseño y los diagramas de interacción para los Casos de Uso de mayor complejidad.
- Aplicación de Patrones estudiados que contribuyan a optimizar el modelo de diseño.
- Implementación de los componentes necesarios de forma tal que se dé cumplimiento a los objetivos propuestos.
- Prueba a los componentes desarrollados, así como integrar la solución al SIIPOL.

Con el desarrollo exitoso de las tareas investigativas anteriormente expuestas se espera lograr un software funcional que satisfaga los estándares de calidad requeridos y que cumpla con la totalidad de requisitos funcionales y no funcionales arrojados a partir de la aplicación de la ingeniería de requerimientos a los procesos de Investigación Internacional que ocurren en INTERPOL.

Se utilizarán **métodos científicos** para el correcto desarrollo de la investigación y lograr caracterizar a fondo el objeto de estudio, así como, garantizar el conocimiento del estado del arte, su evolución y relación con otros fenómenos.

Dentro de los métodos científicos a utilizar se encuentra el **histórico-lógico** ya que se realizará el estudio de sistemas gestores de información y aplicaciones policiales desde sus inicios, así como la evolución y desarrollo que han tenido a lo largo del tiempo.

Se utilizará el método **analítico-sintético** para analizar la documentación generada durante la captura de requisitos y extraer de la misma los elementos principales relacionados con el objeto de estudio y el campo de acción de la investigación. Además, se utilizará este método para la recopilación de la bibliografía relacionada con la utilización de patrones de diseño en sistemas gestores de información,

así como, la correspondiente a las diferentes metodologías y herramientas usadas en la construcción de estos sistemas con el fin de obtener las mejores prácticas de desarrollo e integrarlas en la solución del problema científico.

Las **técnicas de investigación** se utilizarán en la obtención de un conjunto de datos importantes que enriquecerán la investigación. El **análisis documental** estará presente en el manejo del volumen de bibliografía a consultar en el estudio de sistemas de gestión policial y en la búsqueda de patrones arquitectónicos y de diseño.

El contenido a desarrollar en el presente trabajo está estructurado en tres capítulos los cuales agruparan los contenidos de la manera siguiente:

Fundamentación teórica: En este capítulo se tratan elementos teóricos de la investigación tales como, Gestión de Información Policial, metodología, lenguaje y herramientas de desarrollo que serán utilizadas para implementar la solución.

Diseño e implementación de la propuesta de solución: En este capítulo se elabora el modelo de diseño y el modelo de datos, los cuales dan paso al modelo de implementación que responden directamente a la solución del problema.

Validación de la solución propuesta: En este capítulo se valida lo implementado en el capítulo anterior, mediante niveles y métodos de pruebas.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se definirán un conjunto de ideas que enmarcarán al lector cualquiera que este sea, en el contexto en el cual se desarrolla la presente investigación. Así mismo se espera lograr un basamento teórico sólido a partir de conceptos y definiciones manejados en la investigación, la exposición de tendencias actuales utilizadas en el mundo para dar respuesta y solución a problemáticas y necesidades similares a las dadas a conocer en este trabajo, que será de gran utilidad en su futuro desarrollo práctico.

Para lograr un trabajo con la calidad requerida se hace necesario un estudio previo, al cual va dedicado este primer capítulo.

1.2 Software de Gestión Policial

En el mundo actual los índices de criminalidad en numerosas partes del mundo resultan alarmantes y cobran a diario miles de vidas en cada rincón del planeta. Ningún país está exento y todos los gobernantes se plantean estrategias a lo largo de su mandato, para disminuir en gran medida estos hechos delictivos.

Hoy día la gestión policial encuentra en la informática una alternativa al verdadero desafío que representa enfrentar las nuevas tendencias de la criminalidad como el crimen organizado, el narcotráfico, el lavado de dinero, el delito medio-ambiental y el ciber-crimen, si bien la informatización no es siempre sinónimo de modernización, es indudable que ella constituye una herramienta necesaria y fundamental. En tal caso los Sistemas de Gestión Policial han venido a revolucionar la manera de procesar la información. Entre los más comunes se encuentran los sistemas AFIS (Automated Fingerprint Identification System).Es un mecanismo ideado por las mayores agencias de seguridad del mundo, como Interpol, que consiste de un sistema informático donde las impresiones digitales de criminales se recolectan, digitalizan y almacenan en una base de datos. Esta base luego se utiliza una y otra vez con fines de investigación, frecuentemente con el resultado positivo de encontrar una huella

coincidente. Tanto la tecnología del equipamiento utilizado (hardware) como los programas que se ejecutan (software) utilizan algoritmos extremadamente avanzados. (7)

SIGEP: Sistema de Gestión Penitenciaria, que dará respuesta a las necesidades de gestión, información y apoyo a la toma de decisiones de la Dirección General de Custodia y Rehabilitación del Recluso (DGCRR), con este sistema automatizado se busca: aumentar la eficacia, profesionalismo y equidad en el sistema penitenciario venezolano, de manera de lograr un incremento de la confianza en el sistema penitenciario en general, generar y diseminar información vital para el funcionamiento de los establecimientos penitenciarios. Y esto permitirá generar estadísticas confiables y actualizadas sobre la situación jurídica de los privados de libertad, condiciones de vida y salud, actividades de rehabilitación y reinserción, la situación operativa, la actividad administrativa, entre otras. El mismo tiene como objetivo:

Desarrollar e implantar un sistema informático que soporte las decisiones estratégicas del Ministerio del Interior y Justicia y de la Dirección General de Custodia y Rehabilitación del Recluso, que contribuya a garantizar el respeto a los derechos de los internos, su actividad de rehabilitación y reinserción en la sociedad. (8)

POL - Gestión Integral de Policía Local es un sistema que cuenta con más de 100 instalaciones en toda España. El sistema se ha desarrollado por informáticos que son además miembros de la Policía Local, está enlazado con PDAS para la gestión de multas y recogida "In situ" de desperfectos y anomalías en la vía pública, sistema de SMS, fax y E-mail integrado, gestión de digitalización y archivo de documentos en formato PDF integrado, identificación masiva de vehículos mediante su conexión con la DGT, sistema de reseña de huellas y lectura e identificación de personas mediante su huella dactilar, sistema de GPS integrado e incluido para localización y seguimiento de patrullas. (9)

SIIPOL- Sistema integrado de información policial: Es el software utilizado por el CICPC para auxiliar el proceso investigativo. Este sistema está desarrollado con el lenguaje de programación Natural, el gestor de base de datos Adabas y servidores de base datos SUN 6500. El mismo tiene como propósito registrar la información originada por los casos policiales, relacionada con: Personas, Objetos,

Vehículos, Armas, Armas Orgánicas, Planillas de Detención (PD1), además gestiona información referente a los radios de INTERPOL y a sus personas y elementos de investigación relacionada, entre otros. Toda esta información que maneja el sistema es texto. Además, utiliza datos de otras instituciones como la Oficina Nacional de Identificación y Extranjería (ONIDEX) y el Instituto Nacional de Transporte Terrestre (INTT), enviados en soportes digitales para actualizar la base de datos del SIIPOL cada cierto tiempo. Con respecto a la seguridad se utiliza un programa llamado Natural Security que valida la gestión de los usuarios, deja trazas de todas las operaciones realizadas y restringe algunas operaciones como la actualización y la eliminación de cualquier información que haya sido introducida. (6)

Numerosas son las naciones que se han beneficiado con la instalación de alguno de estos softwares de gestión policial, tal es el caso de Colombia en su incesante lucha contra el narcotráfico, Argentina, Venezuela, España, Estados Unidos, entro otros, señalando así el aporte significativo que brindan estos sistemas en procesamiento ágil de la información y la disminución de los índices de criminalidad.

1.3 El Cuerpo de Investigaciones Científicas Penales y Criminalísticas

El Cuerpo de Investigaciones Científicas, Penales y Criminalísticas (CICPC) posee como visión ser la Institución indispensable, por su reconocida capacidad científica y máxima excelencia de sus recursos, con la finalidad de alcanzar el más alto nivel de credibilidad nacional e internacional en la investigación del fenómeno delictivo organizado y criminalidad violenta. (4)

El CICPC ostenta entre sus objetivos fundamentales el de optimizar las acciones de investigación criminal con el fin de lograr el esclarecimiento de los hechos delictivos; capacitar el personal integrado en la institución, garantizando de esta forma la eficacia y la eficiencia; elevar el sentido de pertenencia a partir de la práctica de valores indispensables para el CICPC; garantizar las acciones y medios tendentes a mejorar la calidad de vida de sus miembros, en el aspecto educativo, cultural, deportivo, social y económico, consolidar la imagen de profesionalismo de la institución ante la comunidad en general, fundamentada en una gerencia de alta capacidad de respuesta; lograr insertarse en la comunidad internacional como organismo de investigación penal de vanguardia; dotar al capital humano del CICPC de herramientas, mecanismos logísticos y de infraestructura que garanticen el

óptimo desempeño de sus funciones; apoyar las políticas de estado a través de estrategias dirigidas a la reducción de los delitos en todas sus modalidades; fortalecer organizacionalmente la institución y su sinergia con otros organismos de la administración pública nacional y con instituciones privadas. (3)

1.3.1 Procesos de INTERPOL

La dirección de Policía Internacional del CICPC que posee funciones como: Mantener el enlace con los diferentes organismos policiales del país, así como Oficinas Centrales Nacionales e Internacionales. Adoptar las resoluciones de convenios internacionales acordadas por la Organización Internacional de Policía Criminal (O.I.P.C.). Representar al país en las reuniones de la Asamblea General de la O.I.P.C. en los casos que así lo decida la Dirección General Nacional. Mantener contacto permanente con la Oficina Nacional Identificación y Extranjería (ONIDEX) a fin de disponer de información sobre los movimientos migratorios de ciudadanos. Cumplir con los parámetros legales establecidos, ante la solicitud de extradición de ciudadanos. Mantener actualizados los registros policiales de venezolanos y extranjeros solicitados internacionalmente para asegurar el adecuado flujo de información. (5)

Para la comunicación entre fuerzas policiales de todo el mundo utilizan el Sistema Mundial de Comunicación Policial de INTERPOL I-24/7 que permite a los investigadores establecer conexiones entre segmentos de información aparentemente inconexos, lo que facilita las investigaciones y ayuda a resolver los casos. (10)

Posee como principales funciones:

- Ayudar a un funcionario del servicio de inmigración de un aeropuerto a descubrir que un pasaporte presentado por un viajero ha sido objeto de una denuncia de robo;
- Permitir que un funcionario de fronteras o de los servicios de aduanas consulte el número de identificación de un vehículo para determinar si se ha notificado su robo;
- Alertar a las autoridades nacionales acerca de personas buscadas por la justicia que podrían tratar de entrar en el país por vía aérea o marítima.

Se conoce muy poco acerca de las características de este sistema debido a que la confidencialidad, integridad y la seguridad de los datos que almacena es en gran medida el éxito del mismo.

1.4 El Sistema de Investigación e Información Policial (SIIPOL)

El SIIPOL es el software utilizado por el CICPC para llevar a cabo las actividades fundamentales relacionadas a los procesos de investigación policial. Su propósito es apoyar la labor de investigación policial así como el registro de la información originada por los casos policiales en la lucha contra la actividad delictiva y a favor de la seguridad ciudadana, llevar el control de los registros de detención de personas y de los elementos o imágenes asociadas a los casos, contribuyendo de esta forma con la seguridad y organización de la información.

1.5 Metodología, Lenguajes y Herramientas de Desarrollo

El entorno de desarrollo, metodología, lenguaje de programación y modelado, herramienta de modelado, entre otros, definido para la construcción del software, fue producto de un estudio realizado por el equipo de arquitectura del proyecto, y establecido como políticas del proyecto, por lo que la selección del mismo queda fuera del alcance del presente trabajo. Solo se brindará una breve descripción de cada herramienta a utilizar.

1.5.1 Metodología de Desarrollo

Las metodologías se basan en una combinación de los modelos de proceso genéricos (cascada, evolutivo, incremental, etc.). Adicionalmente una metodología debería definir con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas, guías de adaptación de la metodología al proyecto, guías para uso de herramientas de apoyo, etc. Habitualmente se utiliza el término “método” para referirse a técnicas, notaciones y guías asociadas, que son aplicables a una (o algunas) actividades del proceso de desarrollo. (11)

Piattini define a la metodología de desarrollo de software como “un conjunto de procedimientos, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software”. (11)

Con el objetivo de producir eficiente y eficazmente un software que cumpla los requisitos del cliente, un proceso de desarrollo de software tiene como objetivo primordial, elevar la calidad del software en todas las fases por las que este atraviesa, a través de una mayor transparencia y control sobre el proceso.

1.5.1.1 Proceso Unificado de Desarrollo (RUP)

El Proceso Unificado de Desarrollo (RUP), por sus siglas en inglés Rational Unified Process, se ha convertido en un estándar de facto en no pocas ocasiones para líderes y gerentes en la medida en que estos se interesan por mejorar los procesos de desarrollo de software que tienen lugar en sus organizaciones. Los autores del Proceso Unificado de Desarrollo destacan que el proceso de software propuesto por RUP tiene tres características esenciales: (12)

Dirigido por Casos de Uso: Los Casos de Uso son una técnica de captura de requisitos que fuerza a pensar en términos de importancia para el usuario y no sólo en términos de funciones que sería bueno contemplar. Se define un Caso de Uso como un fragmento de funcionalidad del sistema que proporciona al usuario un valor añadido. Los Casos de Uso representan los requisitos funcionales del sistema. En RUP los Casos de Uso no son sólo una herramienta para especificar los requisitos del sistema. También guían su diseño, implementación y prueba. (13)

Centrado en la arquitectura: La arquitectura de un sistema es la organización o estructura de sus partes más relevantes, lo que permite tener una visión común entre todos los involucrados (desarrolladores y usuarios) y una perspectiva clara del sistema completo, necesaria para controlar el desarrollo. La arquitectura involucra los aspectos estáticos y dinámicos más significativos del sistema, está relacionada con la toma de decisiones que indican cómo tiene que ser construido el sistema y ayuda a determinar en qué orden. Además, la definición de la arquitectura debe tomar en consideración elementos de calidad del sistema, rendimiento, reutilización y capacidad de evolución por lo que debe ser flexible durante todo el proceso de desarrollo. La arquitectura se ve influenciada por la plataforma software, sistema operativo, gestor de bases de datos, protocolos, consideraciones de desarrollo como sistemas heredados. Muchas de estas restricciones constituyen requisitos no funcionales del sistema. (13)

Iterativo e incremental: Para esto, la estrategia que se propone en RUP es tener un proceso iterativo e incremental en donde el trabajo se divide en partes más pequeñas o mini proyectos. Permitiendo que el equilibrio entre Casos de Uso y arquitectura se vaya logrando durante cada mini proyecto, así durante todo el proceso de desarrollo. Cada mini proyecto se puede ver como una iteración (un recorrido más o menos completo a lo largo de todos los flujos de trabajo fundamentales) del cual se obtiene un incremento que produce un crecimiento en el producto. (13)

El proceso de ciclo de vida de RUP está estructurado en varios flujos de trabajo que se evidencian a lo largo de cuatro fases inicio, elaboración, construcción y transición. (13)

Fases:

Inicio: Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.

Elaboración: Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales) identificados de acuerdo al alcance definido.

Construcción: Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene 1 o varias versiones del producto que han pasado las pruebas. Se ponen estos entregables a consideración de un subconjunto de usuarios.

Transición: La versión ya está lista para su instalación en las condiciones reales. Puede implicar reparación de errores.

Flujos de Trabajo:

Modelación del negocio: Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.

Requerimientos: Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
Análisis y diseño: Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.

Implementación: Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.

Prueba: Busca los defectos a los largos del ciclo de vida.

Despliegue: Produce un entregable del producto y realiza actividades como empaque, instalación, asistencia a usuarios, etc. para entregar el software a los usuarios finales.
Administración del proyecto: Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.

Administración de configuración y cambios: Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.

Ambiente: Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

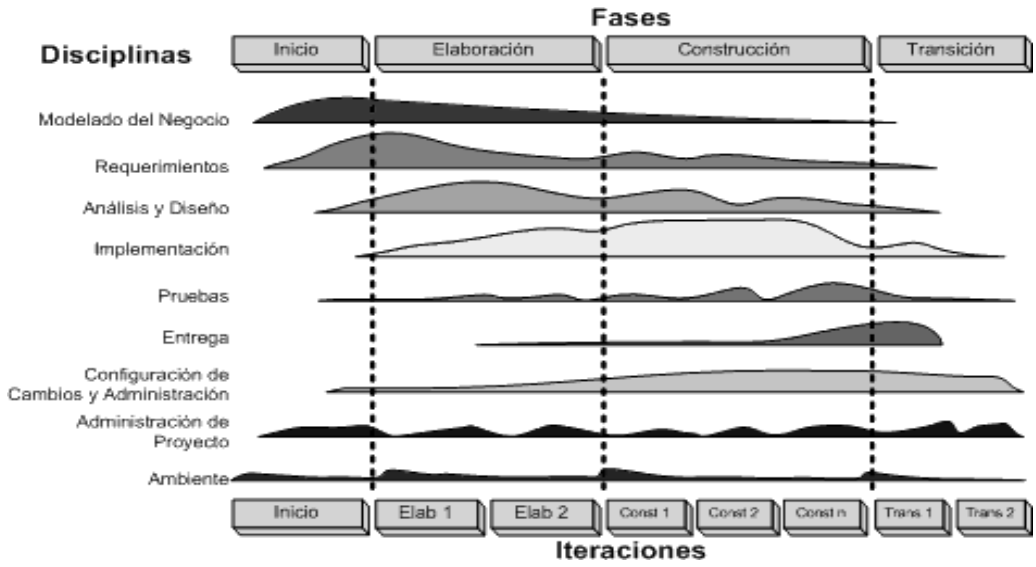


Ilustración 1. Flujos y Fases de RUP.

1.5.2 Lenguaje de Modelado

UML en su versión 2.1 o lenguaje Unificado de Modelado (por sus siglas en inglés, Unified Modeling Language) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. (13)

El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que es un lenguaje, cuenta con reglas para combinar tales elementos. Es importante destacar que UML no es una guía para realizar el análisis y diseño orientado a objetos, es decir, no es un proceso. UML es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos. (13)

La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. Es importante destacar que un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema. (13)

1.5.3 Herramientas de Modelado

Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. Como es sabido, los estados en el Ciclo de Vida de desarrollo de un Software son: Investigación Preliminar, Análisis, Diseño, Implementación e Instalación. (14)

Como herramienta CASE de modelado con UML se utilizó Visual Paradigm for UML 6.4 Enterprise Edition la cual es una herramienta de diseño UML y herramienta CASE diseñada para la ayuda al desarrollo de software. VP-UML soporta los principales estándares de la industria tales como Lenguaje de Modelado Unificado (UML), SysML, BPMN, XMI, etc ofrece un completo conjunto de herramientas de los equipos de desarrollo de software necesario para los requisitos de la captura, software de planificación, la planificación de controles, el modelado de clases, modelado de datos, etc (15) .Permite la integración con entornos de desarrollo como Eclipse/IBM WebSphere, JBuilder, NetBeans IDE, Oracle JDeveloper, BEA Weblogic y de control de versiones como subversión. Una de las características más novedosas de su uso es que brinda la posibilidad de sincronización del modelo de diseño y el código fuente en todo el ciclo de desarrollo del software al ser integrado con el Eclipse; posee una herramienta de generación de reportes en formato PDF o HTML configurable y selectivo.

Está disponible en varias ediciones, cada una destinada a unas necesidades: Enterprise, Professional, Community, Standard, Modeler y Personal.

1.5.4 Lenguajes de Programación

Un lenguaje de programación es aquel elemento dentro de la informática que nos permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador para que este pueda comunicarse con los dispositivos hardware y software existentes. (16)

1.5.4.1 Java

El lenguaje de programación Java, fue diseñado por la compañía Sun Microsystems Inc., con el propósito de crear un lenguaje que pudiera funcionar en redes computacionales heterogéneas (redes

de computadoras formadas por más de un tipo de computadora, ya sean PC, MAC's, estaciones de trabajo, etc, y que fuera independiente de la plataforma en la que se vaya a ejecutar. Esto significa que un programa de Java puede ejecutarse en cualquier máquina o plataforma. El lenguaje fue diseñado teniendo en cuenta las siguientes características: (17)

Simple: elimina la complejidad de los lenguajes como "C" y da paso al contexto de los lenguajes modernos orientados a objetos. La filosofía de programación orientada a objetos es diferente de la programación convencional.

Robusto: maneja la memoria de la computadora. No hay necesidad de preocuparse por apuntadores, memoria que no se esté utilizando, etc. Java realiza todo esto sin necesidad de que el programador se lo indique.

Seguro: tiene ciertas políticas que evitan se puedan codificar virus con este lenguaje. Existen muchas restricciones, especialmente para los applets, que limitan lo que se puede y no puede hacer con los recursos críticos de una computadora.

Portable: como el código compilado de Java (conocido como byte code) es interpretado, un programa compilado de Java puede ser utilizado por cualquier computadora que tenga implementado el intérprete de Java.

Independiente a la arquitectura: al compilar un programa en Java, el código resultante es un tipo de código binario conocido como byte code. Este código es interpretado por diferentes computadoras de igual manera, solamente hay que implementar un intérprete para cada plataforma. De esa manera Java logra ser un lenguaje que no depende de una arquitectura computacional definida.

Dinámico: no requiere que se compilen todas las clases de un programa para que este funcione. Al realizar una modificación a una clase, Java se encarga de realizar un Dynamic Bynding o un Dynamic Loading para encontrar las clases.

1.5.5 Plataforma de Desarrollo

En un proyecto de software, para seleccionar la plataforma en la que se desarrollará el mismo, se toma en cuenta el entorno de ejecución. Cuando se necesita una solución informática multiplataforma, de alto rendimiento, escalabilidad y seguridad, la decisión más provechosa es sin lugar a dudas, la plataforma Java Enterprise Edition (J2EE) (18); por esta razón, se toma como plataforma de desarrollo para el SIIPOL.

J2EE, es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en lenguaje de programación Java, con arquitectura de N niveles, distribuida, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. J2EE define un estándar para el desarrollo de aplicaciones empresariales multicapa diseñado por Sun Microsystems. J2EE simplifica además, las aplicaciones empresariales basándolas en componentes modulares y estandarizados, proveyendo un completo conjunto de servicios a estos componentes, y manejando varias de las funciones de la aplicación de forma automática, sin necesidad de una programación compleja. (18).

Con esta elección se garantiza que el SIIPOL sea una solución informática potente, robusta, eficiente y capaz de integrarse a otras aplicaciones dentro de la amplia gama de sistemas en el territorio venezolano.

1.5.6 Frameworks utilizados en la Solución

El concepto framework se emplea en muchos ámbitos del desarrollo de sistemas software, no solo en el ámbito de aplicaciones Web. Podemos encontrar frameworks para el desarrollo de aplicaciones médicas, de visión por computador, para el desarrollo de juegos, entre otros. En general, con el término framework, nos estamos refiriendo a una estructura 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 podemos añadirle las últimas piezas para construir una aplicación concreta. (19)

Para el desarrollo del sistema SIIPOL se llevó a cabo una selección de frameworks que brindan soporte a cada una de las capas de la aplicación, de las cuales se expondrá una breve descripción.

1.5.6.1 Capa de Presentación

JavaServer Faces (JSF) en su versión 1.2, constituye un marco de trabajo (*framework*) de interfaces de usuario del lado de servidor para aplicaciones web basadas en tecnología Java y en el patrón MVC (Modelo Vista Controlador). (20)

Los principales componentes de la tecnología JavaServer Faces son: (20)

- Representar componentes de interfaz de usuario (UI-*User Interface*) y manejar su estado
- Manejar eventos, validar en el lado del servidor y convertir datos
- Definir la navegación entre páginas
- Soportar internacionalización y accesibilidad, y
- Proporcionar extensibilidad para todas estas características.
- Una librería de etiquetas JavaServer Pages (JSP) personalizadas para dibujar componentes UI dentro de una página JSP.

Ajax4JSF es una extensión Open Source del estándar JavaServer Faces que añade las capacidades Ajax a las aplicaciones JSF sin necesidad de escribir código JavaScript. Presenta mejoras sobre los propios beneficios del framework JSF incluyendo el ciclo de vida, validaciones, facilidades de conversión y el manejo de recursos estáticos y dinámicos. También permite definir un evento en una página que invoca una petición Ajax y luego las áreas de la página deberían sincronizarse con el árbol de componentes JSF después de que la petición Ajax cambie los datos en el servidor. (21)

Los tags de Ajax4jsf permiten indicar una lista de identificadores de componentes que serán re-rendereados cuando la llamada Ajax sea procesada, actualizando automáticamente regiones independientes de la página que se hayan indicado previamente. (21)

Dentro de las características más significativas de Ajax4jsf se encuentran: (21)

- Soporte Ajax para páginas con componentes JSF que tienen comunicación asíncrona y actualizaciones de porciones de página.
- Event listeners, validaciones, transformaciones y mensajes.
- La capacidad de habilitar Ajax a componentes JSF sin cambios en el propio componente.
- Características de Ajax pero sin código JavaScript.
- Permite crear componentes propios con soporte built-in para Ajax.
- Trabaja en el lado del servidor.

RichFaces es una librería de componentes visuales para JSF, escrita en su origen por Exadel y adquirida por JBoss. Además, RichFaces posee un framework avanzado para la integración de funcionalidades Ajax en dichos componentes visuales, mediante el soporte de la librería Ajax4JSF. (22)

Son características de RichFaces las siguientes:

- Se integra perfectamente en el ciclo de vida de JSF,
- Incluye funcionalidades Ajax, de modo que nunca vemos el JavaScript y tiene un contenedor Ajax propio,
- Contiene un set de componentes visuales, los más comunes para el desarrollo de una aplicación web rica (Rich Internet Application), con un número bastante amplio que cubren casi todas nuestras necesidades,
- Soporta facelets,
- Soporta css themes o skins,
- Es un proyecto open source, activo y con una comunidad también activa.

1.5.6.2 Capa de Lógica de Negocio

Spring en su versión 3.0.5, es un framework que ha venido a revolucionar la manera de programar aplicaciones Java debido a la facilidad de crear componentes reutilizables, además de integrarse fácilmente con otros frameworks como lo son Hibernate, iBatis, Struts, entre otros; formando una poderosa herramienta para el desarrollo de aplicaciones empresariales. Spring constituye además un

framework de código abierto que interviene en todas las capas arquitectónicas de una aplicación J2EE, y brinda soporte a Java Server Faces. (23)

Spring posee dos características fundamentales, la Inversión de Control (IoC) y la Programación Orientada a Aspectos (AOP) (24).La Inversión de Control mediante la Inyección de Dependencia (DI) permite inyectar las dependencias de un bean en el momento de su creación utilizando un manejador externo, mediante esta técnica la IoC promueve el bajo acoplamiento de las clases. La Programación Orientada a Aspectos complementa la Programación Orientada a Objetos, proponiendo otra manera de pensar sobre la estructura de un programa. La AOP descompone la estructura de un programa en aspectos convirtiéndose en servicios separados de la lógica de negocio que se ejecutan de manera trasversal a la funcionalidad base. La aplicación de la AOP proporciona un código más limpio al separar aquellas funcionalidades inherentes al sistema de las funcionalidades características del negocio.

Las principales ventajas que proporciona este framework son:

- ✓ Manejo de Beans con contexto de aplicación: Puede organizar de forma efectiva nuestros objetos de la capa central, manejar las conexiones por nosotros y puede eliminar la proliferación de los Singleton.

- ✓ Manejo de Transacciones Declarativo: Ofrece el manejo de transacciones declarativas sin utilizar un contenedor EJB. De esta forma, el control de transacciones se puede aplicar a cualquier POJO. El control de transacciones de Spring no está atado a JTA (Java Transaction API) y puede funcionar con diferentes estrategias de transacción (25)

- ✓ Árbol de excepciones de acceso a datos: Proporciona un magnífico árbol de excepciones en lugar de SQL Exception. Para poder utilizar este árbol de excepciones, se debe definir un traductor de excepciones de acceso a datos dentro del fichero de configuración de Spring.

- ✓ Elimina la necesidad de usar distintos y variados tipos de ficheros de configuración.

1.5.6.3 Capa de Acceso a Datos

Hibernate en su versión 3.6, es un framework ORM (Object-Relational Mapping) para Java, que brinda una capa de persistencia objeto/relacional y un generador de sentencias sql. Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. De una manera muy rápida y optimizada puede generar bases de datos en cualquiera de los entornos soportados: Oracle, DB2, MySql, etc. (26)

Hibernate tiene como objetivo fundamental solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el orientado a objetos y el relacional. Para lograr esto le permite al desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Con esta información Hibernate hace posible a la aplicación manipular los datos de la base operando sobre objetos, con todas las características de la programación orientada a objetos. Hibernate genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos. Está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. También tiene la funcionalidad de crear la base de datos a partir de la información disponible. Asimismo ofrece un lenguaje de consulta de datos llamado HQL (Hibernate Query Language), al mismo tiempo que una API para construir las consultas programáticamente (conocida como "criteria"). Hibernate además es Open Source y la licencia del producto está eximida de costo. (26)

1.5.7 Entorno de Desarrollo Integrado

Una vez seleccionada la plataforma J2EE solo restaría determinar el Entorno de Desarrollo Integrado (IDE). Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, o sea, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. El lenguaje Visual Basic, por ejemplo, puede ser usado dentro de las aplicaciones de Microsoft Office, lo que hace posible escribir sentencias Visual Basic en forma de macros para Microsoft Word. (27)

1.5.7.1 Eclipse

El Eclipse es una plataforma de desarrollo de software multilenguaje que contiene un IDE y un sistema de plug-ins para extenderlo. Está escrito en Java y brinda la funcionalidad de programar en este lenguaje por defecto a través del plug-in básico JDT. Además de extender los lenguajes que puede soportar el IDE, el sistema de plug-ins permite adicionar otras funcionalidades útiles como editores visuales de distintos tipos de archivos, internacionalización, conexión con repositorios de control de versiones, etc. (28)

El Eclipse se distribuye bajo la Eclipse Public License, y es por tanto código abierto y software libre.

Debido a este sistema, el IDE Eclipse es muy popular y existen plug-ins de muchos tipos para disimiles funciones. Las principales compañías detrás de los frameworks utilizados (fundamentalmente SpringSource y JBoss) proveen plug-ins que aceleran el desarrollo de software utilizando Eclipse como base. (28)

El SDK de Eclipse incluye por defecto el JDT (Java Development Tools), que ofrece un compilador incremental de Java y un modelo completo de archivos Java. Estas funcionalidades son muy importantes, dado que permiten aplicar técnicas avanzadas de refactorización y análisis de código (28)

1.5.8 Sistema Gestor de Base de Datos

Oracle 10g, es básicamente una herramienta cliente/servidor para la gestión de base de datos la gran potencia que tiene y su elevado precio hace que solo se vea en empresas muy grandes y multinacionales, por norma general. (29)

Oracle es un Sistema Gestor de Bases de Datos con características objeto-relacionales, que pertenece al modelo evolutivo de SGBD. Sus características principales son las siguientes: (30)

- Entorno cliente/servidor.
- Gestión de grandes bases de datos.
- Usuarios concurrentes.
- Alto rendimiento en transacciones.
- Gestión de la seguridad.

- Autogestión de la integridad de los datos.
- Portabilidad.
- Conectabilidad.

1.6 Propuesta de Solución

A lo largo de todo el proceso de desarrollo de software se tiene como meta la realización eficaz de un producto software que reúna las exigencias del cliente, utilizando para su desarrollo tecnologías, computadoras, sistemas operativos, estructuras de red, entornos de desarrollo, lenguajes de programación, frameworks y herramientas para garantizar mayor eficiencia, facilidad de mantenimiento, facilidad de uso, interoperabilidad, seguridad y mejor tiempo de respuesta.

Como Propuesta de solución para el SIIPOL se define una aplicación informática cliente-servidor que hace uso del protocolo http para la conexión con el servidor de aplicación y este a su vez se conectara al servidor de bases de datos.

Ilustración 2. Propuesta de Solución.

Dicha aplicación será construida siguiendo el estilo n-capas: Es un estilo de programación, su objetivo primordial es la separación de la capa de presentación, capa de negocio y la capa de datos. (31)

Ilustración 3.Patrón n capas

1.7 Conclusiones

A partir del estudio realizado como parte del marco teórico se dio a conocer un conjunto de elementos que marcan el punto de partida sobre el cual se apoyara la investigación para iniciar la construcción de un software que cumpla con los objetivos propuestos. Luego de un proceso de búsqueda detallada referente a algunos sistemas similares como: AFIS, SIGEP y POL que pudieran apoyar la elaboración de la propuesta de solución, se evidenció la ausencia de un sistema que sustituya las actuales prestaciones del obsoleto SIIPOL para la dirección de policía internacional del CICPC. El submódulo investigación internacional será implementado de acuerdo con las tecnologías y herramientas más usadas en el campo de la informática expuestas anteriormente, se opta por la decisión de desarrollar una aplicación web, sobre el lenguaje Java. El entorno de desarrollo integrado por la plataforma J2EE, el IDE Eclipse, el gestor de base de datos Oracle, los frameworks JSF, Spring e Hibernate y la herramienta de modelado Visual Paradigm, así como el uso de un estilo arquitectónico en 3 capas. El proceso como tal será orientado por la metodología RUP la cual constituye una guía de cómo se debe desarrollar una aplicación de tal escala, permitirá el desarrollo claro y fluido de un sistema construido sobre bases sólidas y un entorno de desarrollo establecidos por la arquitectura y la dirección del proyecto vistas con anterioridad en la propuesta de solución.

CAPÍTULO 2. DISEÑO E IMPLEMENTACIÓN DE LA PROPUESTA DE SISTEMA.

2.1 Introducción

En este capítulo se desarrollará la propuesta de solución dada a conocer con anterioridad. En la solución, el SIIPOL cuenta con el módulo INTERPOL el cual se divide en tres submódulos, los cuales son Investigación Internacional, Archivo y Solicitudes. El submódulo Investigación Internacional se encarga de gestionar cualquier tipo de información sobre la investigación internacional, relacionada con personas y elementos.

Para automatizar los procesos de INTERPOL se definieron requisitos que son capacidades y condiciones con las cuales debe ser conforme el sistema. (32) Estos comprenden necesidades de información y control, funcionalidad del producto y comportamiento, rendimiento general del producto, diseño, restricciones de la interfaz y otras necesidades especiales. (33)

A continuación los requisitos funcionales y no funcionales respectivamente:

Requisitos funcionales:

- Apertura un Inicio de Investigación.
- Ver los datos de un Inicio de Investigación.
- Asociar Elementos de Información.
- Modificar los datos de un Inicio de Investigación.
- Validar la integridad de los datos introducidos por el usuario.
- Mantener informado al usuario de los resultados de las operaciones.
- Ver los datos de un Expediente.
- Crear Tarjeta de Interpol.
- Ver los datos de las Diligencias.
- Incluir una Difusión.
- Ver los datos de una Difusión.
- Crear Tarjeta de Archivo Internacional.
- Modificar los datos de una Difusión.

- Incluir un Acta de Entrega.
- Ver los datos del Acta de Entrega.
- Asociar Funcionario al Acta de Entrega.
- Asociar Personas Externa al Acta de Entrega.
- Modificar los datos del Acta de Entrega.
- Incluir una Minuta de Información.
- Ver los datos una Minuta de Información
- Asociar Elementos a la Minuta de Información (Persona, Arma, Vehículo, Arma).
- Modificar los datos de la Minuta de Información.
- Incluir una Línea de Investigación.
- Ver los datos de una Línea de Investigación.
- Modificar los datos de una Línea de Investigación.
- Incluir una Respuesta de Localización de Elemento.
- Ver los datos de una Respuesta de Localización de Elemento.
- Modificar los datos de una Respuesta de Localización de Elemento.
- Consultar libro de control de investigación
- Ordenar ascendente por fecha y número de expediente el listado de controles de investigación
- Ordenar descendentemente por fecha y número de expediente el listado de controles de investigación
- Consultar información de Interpol
- Ordenar ascendentemente el listado de informaciones de Interpol
- Ordenar descendentemente el listado de informaciones de Interpol
- Incluir una Información de Interpol
- Ver los datos de una Información de Interpol
- Asociar Elementos de Información.
- Modificar los datos de una Información de Interpol
- Desestimar una Información de Interpol
- Validar la integridad de los datos introducidos por el usuario.

- Mantener informado al usuario de los resultados de las operaciones
- Ver los datos de un Dosier
- Mantener informado al usuario de los resultados de las operaciones

Requisitos no funcionales:

Seguridad:

•RNF 1. Un actor solo tiene acceso a una información interpol según sus permisos en el sistema.

Fiabilidad:

•RNF 2. EL botón que permite validar un número de comunicación cuando se incluye una información de Interpol no se mostrara hasta no escoger la entidad emisora y luego de validado se deshabilitará el campo No. Comunicación.

Para cumplir estos requisitos el submódulo Investigación Internacional cuenta con doce casos de uso donde se encuentran asociados todos los requisitos, los cuales son: CU Ver Expediente de Interpol, CU Ver Dosier, CU Gestionar Respuesta de Localización de Elementos, CU Gestionar Minuta de Información, CU Gestionar Línea de Investigación, CU Gestionar Inicio de Investigación de Interpol, CU Gestionar Información de Interpol, CU Gestionar Difusión, CU Gestionar Acta Entrega, CU Consultar Información de Interpol, CU Consultar Documentos de Interpol y CU Consultar libro de control de investigación de Interpol.

Para el desarrollo de la solución, como se establece en la metodología utilizada se realizó modelo de diseño, modelo de datos y por último modelo de implementación.

2.2 Modelo de Diseño

El objetivo de un modelo de diseño es producir un modelo o representación del sistema que será implementado. Como macro actividad posterior al análisis esta se encarga de refinarlo por lo que este debe ser suficiente para que el sistema pueda ser erigido sin ambigüedades. Además, el modelo de diseño sirve como abstracción de la implementación del sistema y es, de este modo, utilizada como una entrada fundamental de las actividades de implementación. En este apartado cobran especial

atención los patrones de diseño representando soluciones a problemas comunes en el desarrollo del software. (34)

2.2.1 Aspectos significativos del Diseño.

El diseño de la solución del submódulo Investigación Internacional se encuentra regido por una arquitectura común para todo el proyecto; esta, como se menciona anteriormente es de n-capas de tres niveles la cual ofrece desarrollo en paralelo, una aplicación más robusta debido al encapsulamiento, flexibilidad, escalabilidad y una mayor facilidad de mantenimiento y soporte.

La aplicación define varios roles para familias específicas de elementos. Cada una de estas familias o roles, se encarga de tareas particulares, se encuentra en lugares específicos y se nombra de una forma específica.

En cada módulo se encuentra un paquete **común** donde se encuentran las funcionalidades o clases útiles repetidas en el módulo en cuestión.

Todos los módulos y submódulos tienen una estructura conformada por:

Config: Este paquete contendrá toda la configuración necesaria para realizar las operaciones del módulo

Dao: Encargados de manejar la persistencia de una entidad cualquiera, tanto para leer como para escribir en la BD

Facade: Las fachadas son los elementos encargados de proveer el punto básico de aplicación por parte de subsistemas y módulos externos. Además, sirven para separar la lógica de cada módulo de las llamadas de los demás.

Web: Agrupa las clases para el trabajo con la presentación como beans manejados, convertidores, validadores y demás clases útiles.

Util: Paquete que contiene cualquier clase útil necesaria en el ámbito del módulo o submódulo en cuestión.

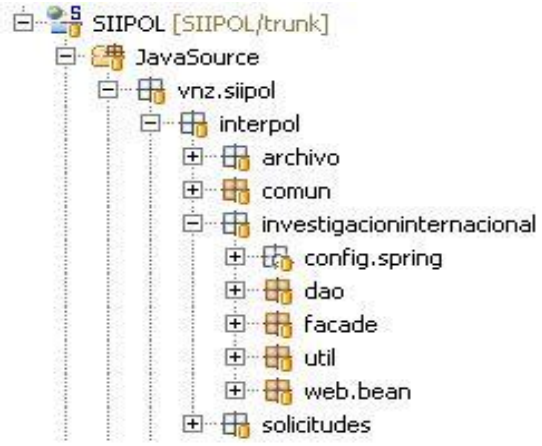


Ilustración 4. Estructura de Paquetes de la Solución.

Durante el diseño de la solución se tuvieron en cuenta algunas clases significativas como son:

Nomenclador: Esta es una clase particular del sistema, que representa todos los nomencladores del mismo. Esta clase es especial debido a que unifica todos los conceptos del sistema que pueden modelarse como nomencladores. Esta decisión de diseño fue tomada debido a que en versiones anteriores la aplicación mostraba un gran gasto de memoria permanente y fue necesario reducir la cantidad de clases del sistema, siendo los nomencladores un claro candidato a ello dado que constituían un conjunto de clases estructural y funcionalmente similares. (28)

El **nomenclador** se relaciona fuertemente con las clases de estereotipo Entidad, pero a diferencia de estas sí realiza lógica. También es única en el sentido de que se mantiene un caché con información de todos los nomencladores existentes en memoria constantemente para acelerar las consultas sobre los mismos (que son muy numerosas). (28)

BaseBean: Constituye la raíz de la jerarquía de beans de respaldo de JSF. Proporciona muchas funcionalidades necesarias para la capa de presentación, como formato y visualización de mensajes, limpieza de beans del historial de navegación de la sesión, resolución de variables, acceso al contexto de JSF y muchas otras. (28)

ComunFacade: Constituye la raíz de la jerarquía de las fachadas de todos los módulos. Proporciona muchas funcionalidades necesarias para la capa de presentación, como funcionalidad CRUD y funciones para consultas básicas. Esta clase tiene la característica de que, a diferencia de las demás jerarquías de la aplicación, en esta no es posible utilizar la fachada común directamente porque todos los módulos necesitan una fachada (aunque sea una marker interface vacía), porque la existencia de una fachada es la que define la existencia misma del módulo. Para usar la fachada común directamente, el llamador no puede pertenecer a ningún subsistema de la aplicación, en otro caso, debe usar la fachada de su subsistema. (28)

EntidadPersistenteBase: Esta clase determina la raíz de la jerarquía de Entidades que se encuentra en el paquete **domain**. Las clases de este estereotipo no suelen realizar ninguna lógica, así que el papel de esta clase se reduce a garantizar la existencia de algunos atributos obligatorios para todas las entidades como el id o el campo activo, y a brindar un tope único desde el cual agrupar a todas las entidades del sistema. En calidad de esto último es muy útil como restricción en consultas genéricas y para restringir las llamadas a métodos en daos o fachadas. (28)

DaoGenerico: Esta clase es la raíz de la jerarquía de daos de la aplicación. Contiene numerosas funcionalidades, incluyendo soporte para todas las funcionalidades CRUD, soporte para el API simplificado de consulta, así como métodos auxiliares de uso obligatorio que determinan la forma que deben tener las Criterias, HQL y SQL de la aplicación. También soportan el paginado y constituye el punto único de acceso a las sesiones de Hibernate y la fuente de datos configurada para la aplicación. Esta clase es la más importante de su capa, y debido a lo numeroso de los servicios que provee su jerarquía es la que más se ajusta al precepto arquitectónico de usar la clase genérica lo más posible y heredar solo para manejar los casos especiales. (28)

2.2.2 Patrones.

Un patrón de diseño es una abstracción de una solución en un alto nivel. Constituye la respuesta a un problema de diseño no trivial que es efectiva (resolvió el problema satisfactoriamente en ocasiones anteriores) y reusable (puede aplicarse a diferentes problemas de diseño en distintas circunstancias). Cuando se tienen conocimientos sobre el comportamiento y los beneficios de los patrones de diseño, uno de los errores más comunes que suele cometerse es querer aplicarlos en todo momento en el sistema que se esté desarrollando, provocando así su uso excesivo muchas veces sin realizar un previo análisis para determinar si sería efectivo realmente y aportaría la reusabilidad, extensibilidad y mantenimiento que se necesita en la solución desarrollada.

Para la construcción de dicha solución como se explica en el capítulo anterior se usaron diferentes framework. Dichos framework incorporan muchos patrones como es el caso de Jsf con el patrón arquitectónico **MVC** (Modelo Vista Controlador), que separa los datos, la interfaz de usuario y la lógica del control en tres elementos diferentes permitiendo construir aplicaciones más grandes así como fácil de modificar y mantener debido a la clara separación de tareas.

Con el propósito de lograr la simplicidad y eficiencia en el sistema, se seleccionaron un conjunto de patrones de diseño a utilizar en la solución, cuyas principales características serán enunciadas seguidamente.

Composite View: Un objeto vista que está compuesto de otros objetos vista, es usado por ejemplo cuando se quiere incluir un subview.jsp dentro de una página .jsp. (35)

Facade: Un patrón común, sencillo de implementar, que consiste en una interfaz única presentada por cualquier componente (más significativamente, un subsistema o un módulo) de forma que se aíslan las complejidades de implementación de la misma de aquellos otros componentes que la utilizan. Las fachadas permiten el trabajo entre varios equipos de forma independiente, hasta que es necesario cambiar el dominio común de varios módulos o los métodos mismos de la fachada. (28)

Domain Model: Este patrón descrito por Fowler consiste básicamente en usar las técnicas de diseño orientado a objetos para modelar mediante clases el dominio del problema. Un modelo de dominio consiste en una trama de objetos interconectados, una gran parte de los cuales presentan estado y comportamiento y cada uno de ellos modela un concepto del problema. El Modelo de Dominio es una vía para enfrentar con éxito negocios complejos y constituye el núcleo de la capa de negocio y de todo el sistema. La implementación de dicho Modelo de Dominio usa POJOs (Plain Old Java Object) para permitir el uso más sencillo de los frameworks elegidos y la migración de estos a versiones superiores a medida que el proyecto avance, así como la introducción de nuevas tecnologías y el desfasaje de las obsoletas. (28)

Front Controller: Provee un punto de partida centralizado que controla y manipula las solicitudes a recursos web, incluyendo la invocación de servicios de seguridad como la autenticación y la autorización, delegando el procesamiento de negocios a los objetos responsables. El patrón Controlador Frontal dirige la elección a la vista apropiada y realiza la manipulación de los errores. (28)

Singleton (instancia única): El mismo es quizás el más sencillo de los patrones y garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a esta instancia. El patrón de diseño singleton está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella. (37)

Iterator: Provee una forma de acceder secuencialmente a los elementos de un objeto sin exponer su representación interna. El lenguaje de programación Java implementa esta funcionalidad en la mayoría de las colecciones con las que cuenta. Este patrón es una vía de separar el recorrido a través de un conjunto de nodos. Por lo general, la programación genérica es una buena forma de implementar iteradores. (28)

DAO (Data Access Object): Concentran la lógica especial de persistencia de las entidades del dominio de la aplicación. Los DAO contienen todas las sentencias de interacción de la aplicación con la base de datos, así como su transformación para la presentación. (28)

2.2.3 Diagramas de clases del Diseño.

La abstracción es una de las formas fundamentales en las que los humanos se enfrentan a la complejidad. La arquitectura es la estructura u organización de los componentes del programa, la manera en que estos componentes interactúan, y la estructura de datos que utilizan los componentes. Los patrones describen una estructura de diseño que resuelve un problema de diseño particular dentro de un contexto especifico y en medio de “fuerzas” que pueden tener un impacto en la manera en la que se aplica y utiliza el patrón, de manera tal que el diseñador pueda decidir si este es el necesario o el que puede usar para su trabajo actual.

A continuación se detalla el diseño de los casos de uso seleccionados, los mismos son: CU Ver Expediente de Interpol, CU Gestionar Línea de Investigación, CU Gestionar Inicio de Investigación de Interpol, CU Consultar Documentos de Interpol.

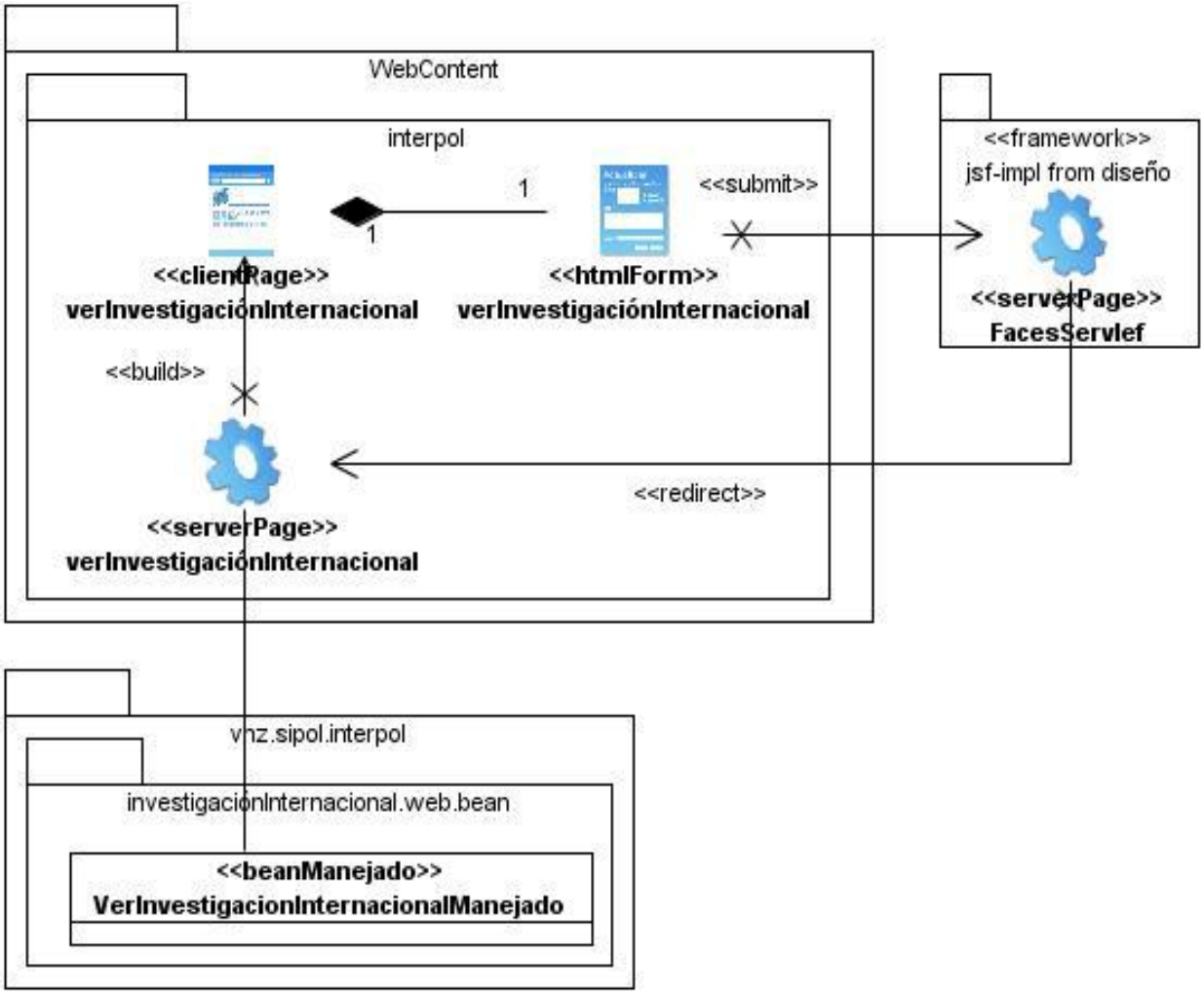


Ilustración 5. CU Ver Expediente de Interpol. Vista del diseño de la capa de presentación.

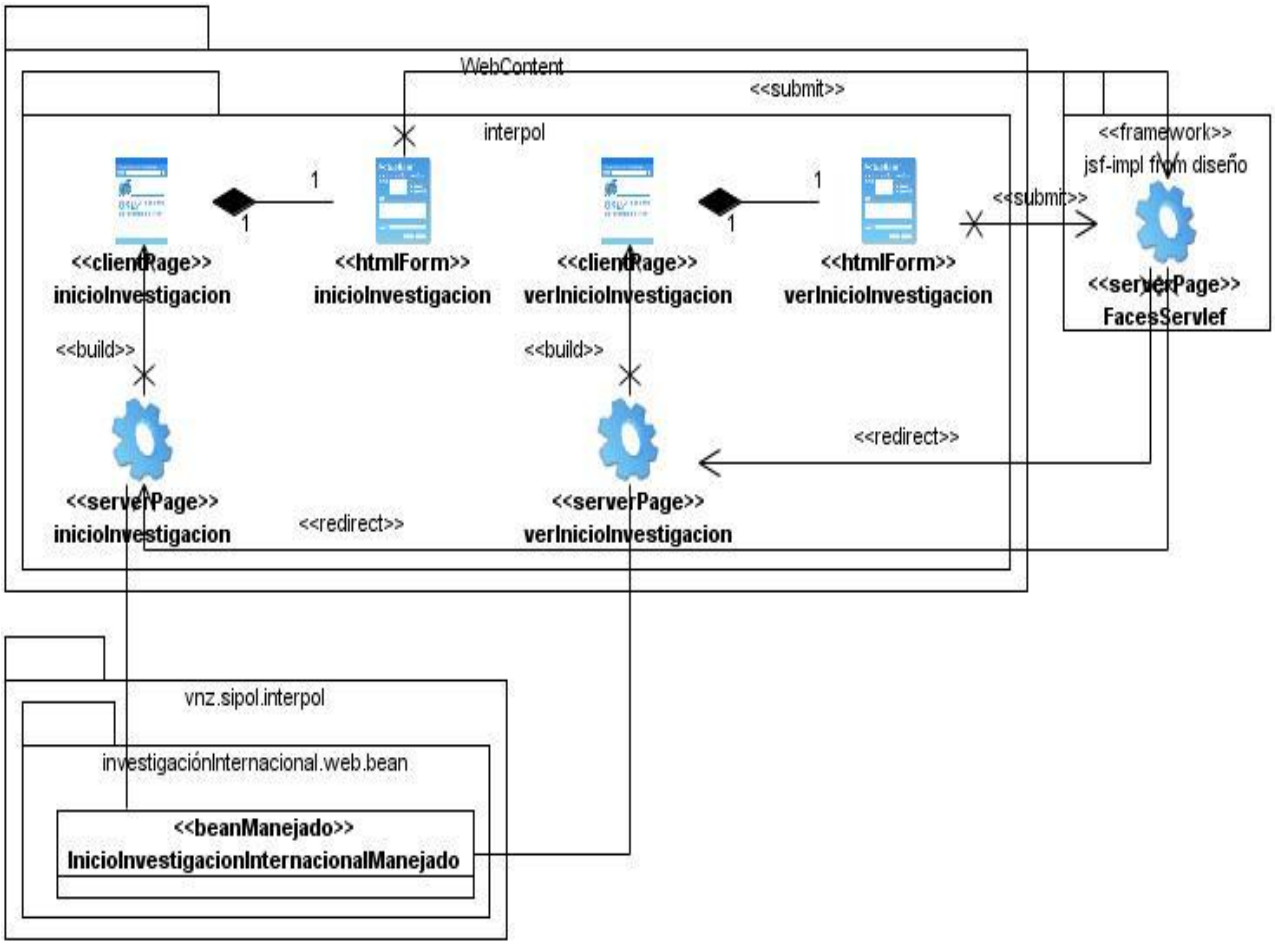


Ilustración 6. CU Gestionar Inicio de Investigación de Interpol. Vista del diseño de la capa de presentación.

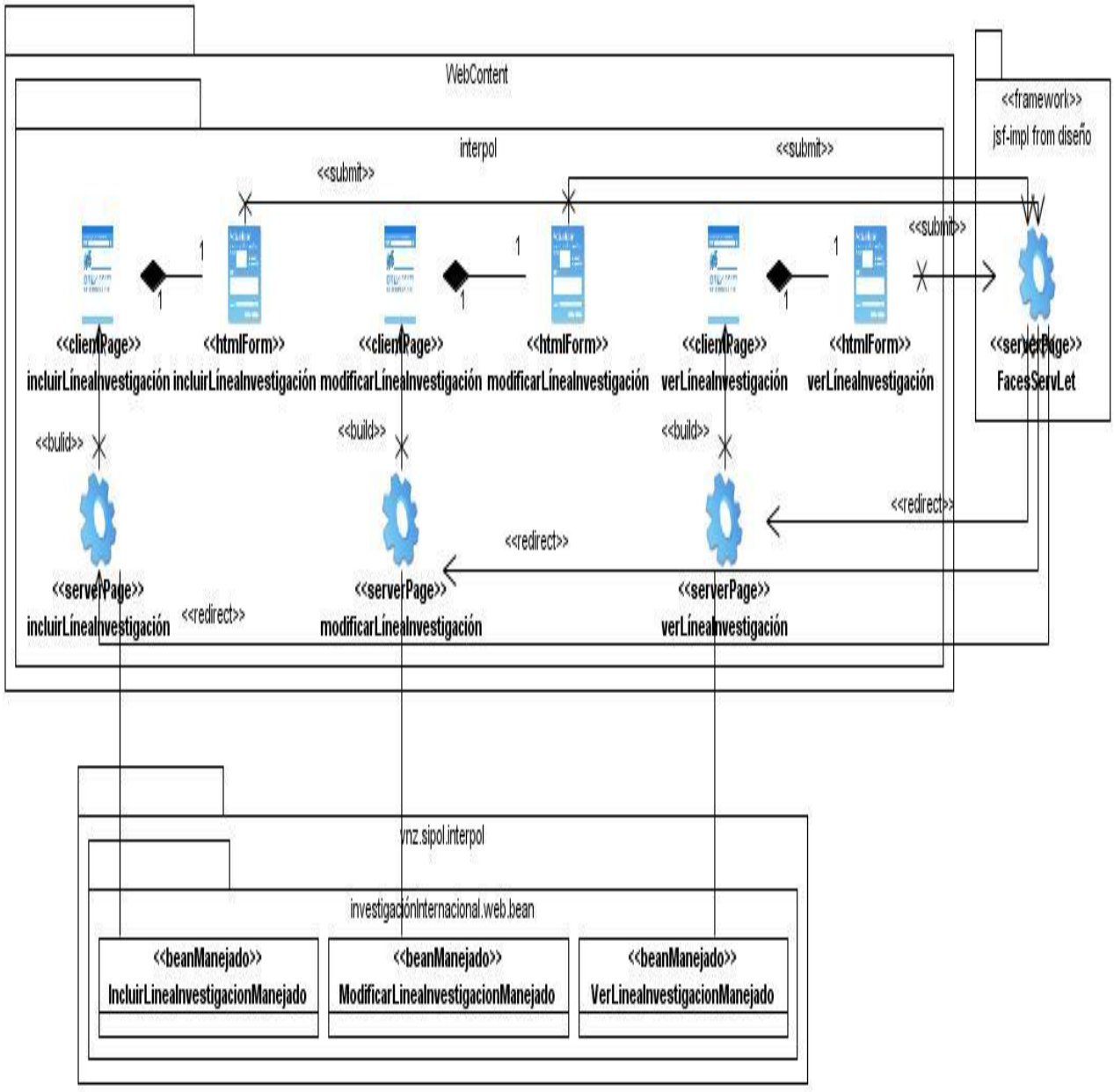


Ilustración 7. CU Gestionar Línea de Investigación. Vista del diseño de la capa de presentación.

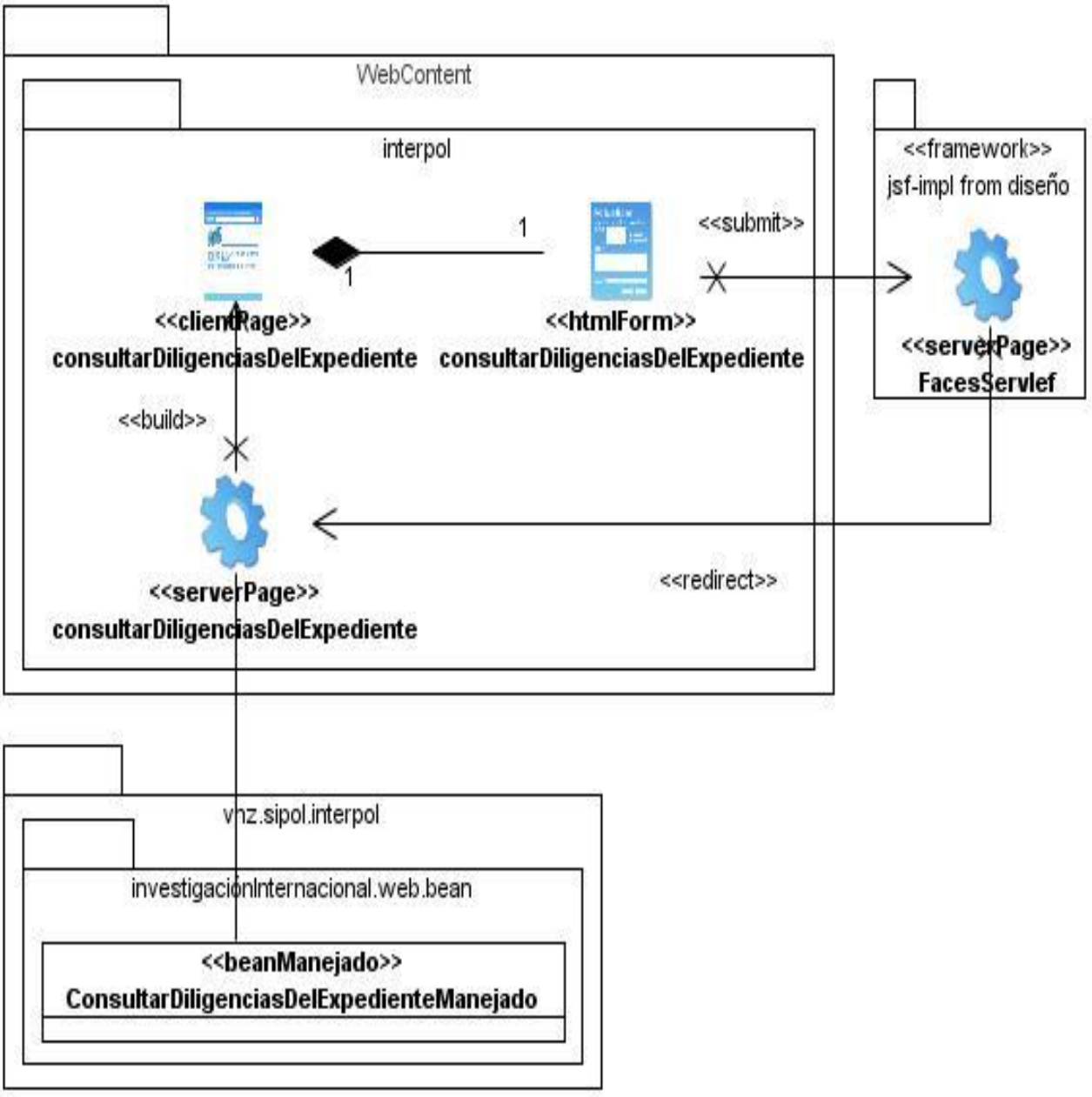


Ilustración 8. CU Consultar Diligencias de Interpol.doc. Vista del diseño de la capa de presentación.

2.2.4 Clases significativas propias de la Solución

Para la solución se tuvo en cuenta la creación de clases que agrupan funcionalidades comunes, con el fin de eliminar la repetición innecesaria de código y lograr una mayor flexibilidad en el sistema a la hora de realizar futuras refactorizaciones, pedidos de cambio solicitados por el cliente o la incorporación de nuevos componentes desarrollados por el equipo de Arquitectura. Estas clases son descritas a continuación.

Nombre: ExpedienteInternacional	
Tipo de clase: Entidad	
Atributo	**Tipo**
paisSolicitante	**Nomenclador**
Denuncia	**Denuncia**
Observaciones	**String**
Funcionarioactuante	**Funcionario**
fuenteOrigenInvestigacion	**Nomenclador**
medioUtilizado	**Nomenclador**
fiabilidadMedioUtilizado	**Nomenclador**
descripcionNovedad	**String**
Para cada responsabilidad:	
Nombre:	**getAuditClassCode()**
Descripción Nombre:	Brindar el código de identificación de esa entidad para el sistema de auditoría.
Nombre:	**clone()**
Descripción Nombre:	Hacer una copia del objeto (clonarlo)

Nombre: InformaciónInterpol	
Tipo de clase: Entidad	
Atributo	**Tipo**
Motivo	**String**
fuenteOrigen	**Nomenclador**
fuenteOrigenOtro	**String**
País	**Nomenclador**

tipoDelito	**Nomenclador**
desestimada	**Boolean**
motivoDesestimada	**String**
entidadEnvia	**Entidad**
personaEnviaExterna	**String**
funcionarioEnvia	**Funcionario**
numeroComunicacion	**String**
Para cada responsabilidad:	
Nombre:	**getPersonaEnvia()**
Descripción Nombre:	Devolver la persona emisora

Nombre: InvestigacionInternacionalFacade	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	**salvarExpedienteInternacional(ExpedienteInternacional expedienteInternacional, Funcionario funcionarioLogueado)**
Descripción Nombre:	Este método salva un expediente internacional dado
Nombre:	**salvarExpedienteComoDosierInternacional(ExpedienteInternacional expedienteInternacionalComoDosier, Funcionario funcionarioLogueado)**
Descripción Nombre:	Este método salva un expediente internacional como Dosier
Nombre:	**generarExpedienteInterpolDesdeDosier(ExpedienteInternacional expedienteInternacionalComoDosier,Funcionario funcionarioLogueado)**
Descripción Nombre:	Este método genera un expediente internacional desde el dosier
Nombre:	**salvarMinutaInformacion(MinutaInformacion minutaInformacion)**
Descripción Nombre:	Este método salva una minuta de información

Nombre:	**salvarLineaInvestigacion(LineaInvestigacionInternacional lineaInvestigacionInternacional)**
Descripción Nombre:	Este método salva la línea de investigación dada
Nombre:	**salvarActaEntrega(ActaEntrega actaEntrega)**
Descripción Nombre:	Este método salva un Acta de Entrega
Nombre:	**salvarInformacionInterpol(InformacionInterpol informacionInterpol);**
Descripción Nombre:	Este método salva una información de interpol
Nombre:	**salvarDifusionInternacional(DifusionInternacional difusionInternacional)**
Descripción Nombre:	Este método salva una difusión internacional
Nombre:	**diligenciasSustancialesExpedienteInternacional(Integer idExpedienteInternacional, Date fechaInicio, Date fechaFin, Integer tipodiligencia)**
Descripción Nombre:	Este método devuelve el listado de las diligencias internacionales de un expediente internacional.
Nombre:	**consultarInformacionInterpol(String noExpediente, Nomenclador pais, Nomenclador tipoDelito, Nomenclador fuenteOrigen, Boolean desestimada, Date fechaInicial, Date fechaFinal)**
Descripción Nombre:	Este método devuelve el lista de informaciones de interpol por uno o varios parámetros de búsqueda especificados.
Nombre:	**consultarLibroControlInvestigacion(ExpedienteInternacional expedienteInterpol, Persona denunciante, Date fechaInicio, Date fechaFin)**
Descripción Nombre:	Este método consulta el libro de control de investigaciones del despacho.

Nombre:	**concluirExpedienteInternacional(ExpedienteInternacional expedienteInternacional)**
Descripción Nombre:	Este método concluye un expediente.

Nombre: InvestigacionInternacionalDao	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	**diligenciasSustancialesExpedienteInternacional(Integer idExpedienteInternacional, Date fechaInicio, Date fechaFin, Integer tipodiligencia)**
Descripción Nombre:	Este método devuelve el listado de las diligencias internacionales de un expediente internacional.
Nombre:	**consultarInformacionInterpol(String noExpediente, Nomenclador pais, Nomenclador tipoDelito, Nomenclador fuenteOrigen, Boolean desestimada, Date fechaInicial, Date fechaFinal)**
Descripción Nombre:	Este método devuelve el lista de informaciones de interpol por uno o varios parámetros de búsqueda especificados
Nombre:	**consultarLibroControlInvestigacion(ExpedienteInternacional expedienteInterpol, Persona denunciante, Date fechaDesde, Date fechaHasta);**
Descripción Nombre:	Este método consulta el libro de control de investigaciones del despacho.

2.2.5. Diagramas de Contrato entre Paquetes

Los diagramas de interacción los cuales representan la secuencia de acciones dentro de un caso de uso, en forma de mensajes que señalan cómo interactúan los objetos entre sí, generalmente se representan mediante diagramas de colaboración o diagramas de secuencia. Siendo estos últimos los

favoritos para el diseño, ya que el centro de atención principal es encontrar secuencias de interacción detalladas y ordenadas en el tiempo.

Por la complejidad que mostraba realizar un diagrama de secuencia mediante la interacción entre objetos, la dirección del proyecto decidió realizar estos diagramas mediante la interacción entre subsistemas a lo cual denominaron “Diagrama de contrato entre paquetes”, convirtiéndose de esta forma en un artefacto generado del proyecto.

Para entender mejor en que se basan estos diagramas, se cita textualmente del libro Proceso Unificado de Desarrollo el siguiente fragmento En los diagramas de secuencia, mostramos las interacciones entre objetos mediante transferencia de mensajes entre objetos o subsistemas. Cuando decimos que un subsistema “recibe” un mensaje, queremos decir en realidad que es un objeto de una clase del subsistema el que recibe el mensaje. Cuando un subsistema envía un mensaje, realmente es un objeto de una clase del subsistema el que envía el mensaje. (12)

Es decir, son diagramas de secuencia que contienen las instancias de actores, subsistemas, y transmisiones de mensajes entre estos y que se diferencian en los siguientes aspectos. (12)

- La línea de vida en los diagramas de secuencia denota subsistemas en lugar de objetos del diseño.
- Cada subsistema identificado debería tener al menos una línea de vida que lo denote en un diagrama de secuencia.
- Si asignamos un mensaje a una operación de una interfaz, puede resultar apropiado cualificar el mensaje con la interfaz que proporciona la operación. Esto es necesario cuando un subsistema proporciona varias interfaces, y debemos distinguir que interfaz se utiliza en cada mensaje.

A continuación se muestra el “diagrama de contrato entre paquetes” de los casos de usos seleccionados anteriormente.

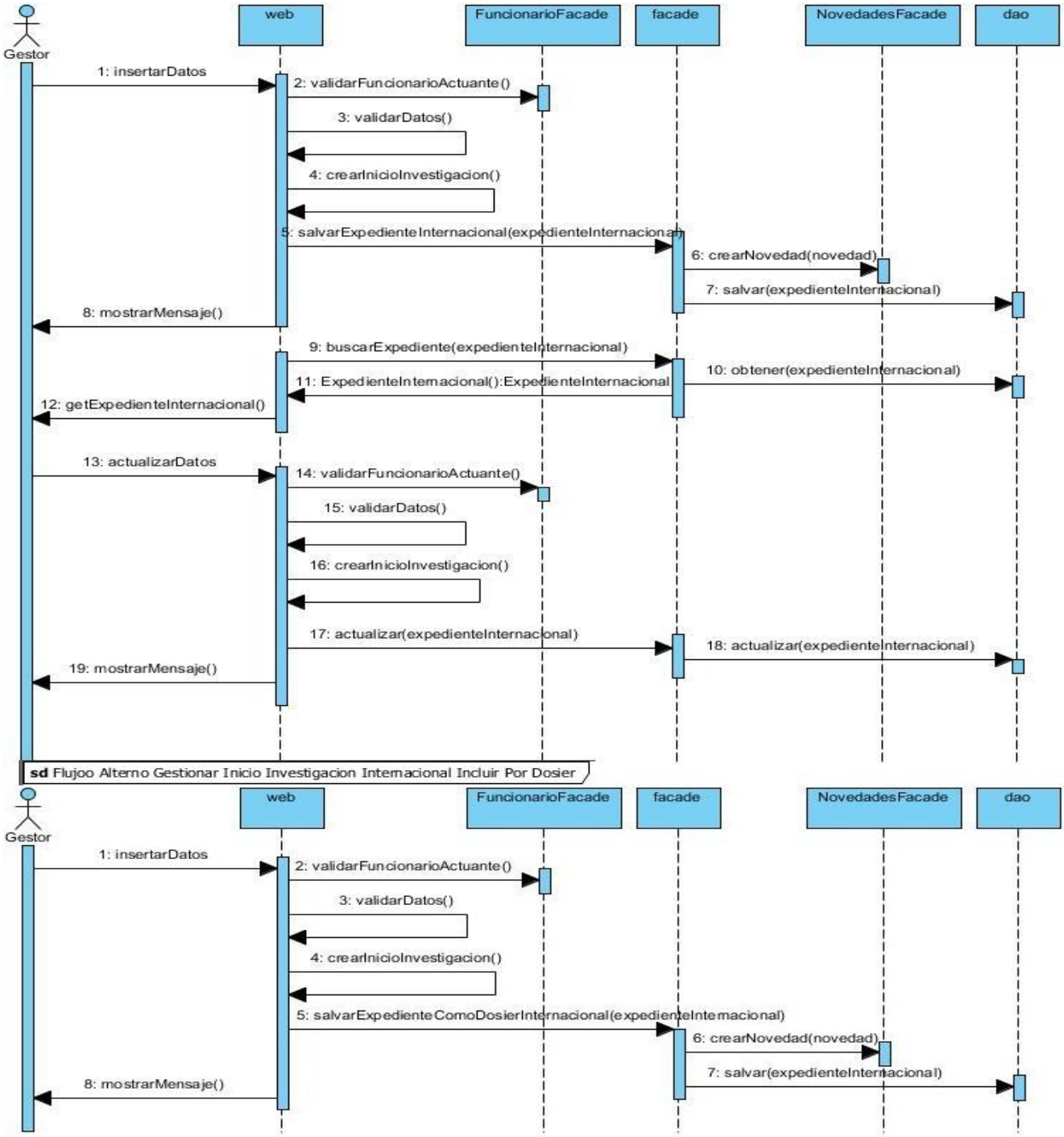


Ilustración 9. CU Gestionar Inicio de Investigación de INTERPOL

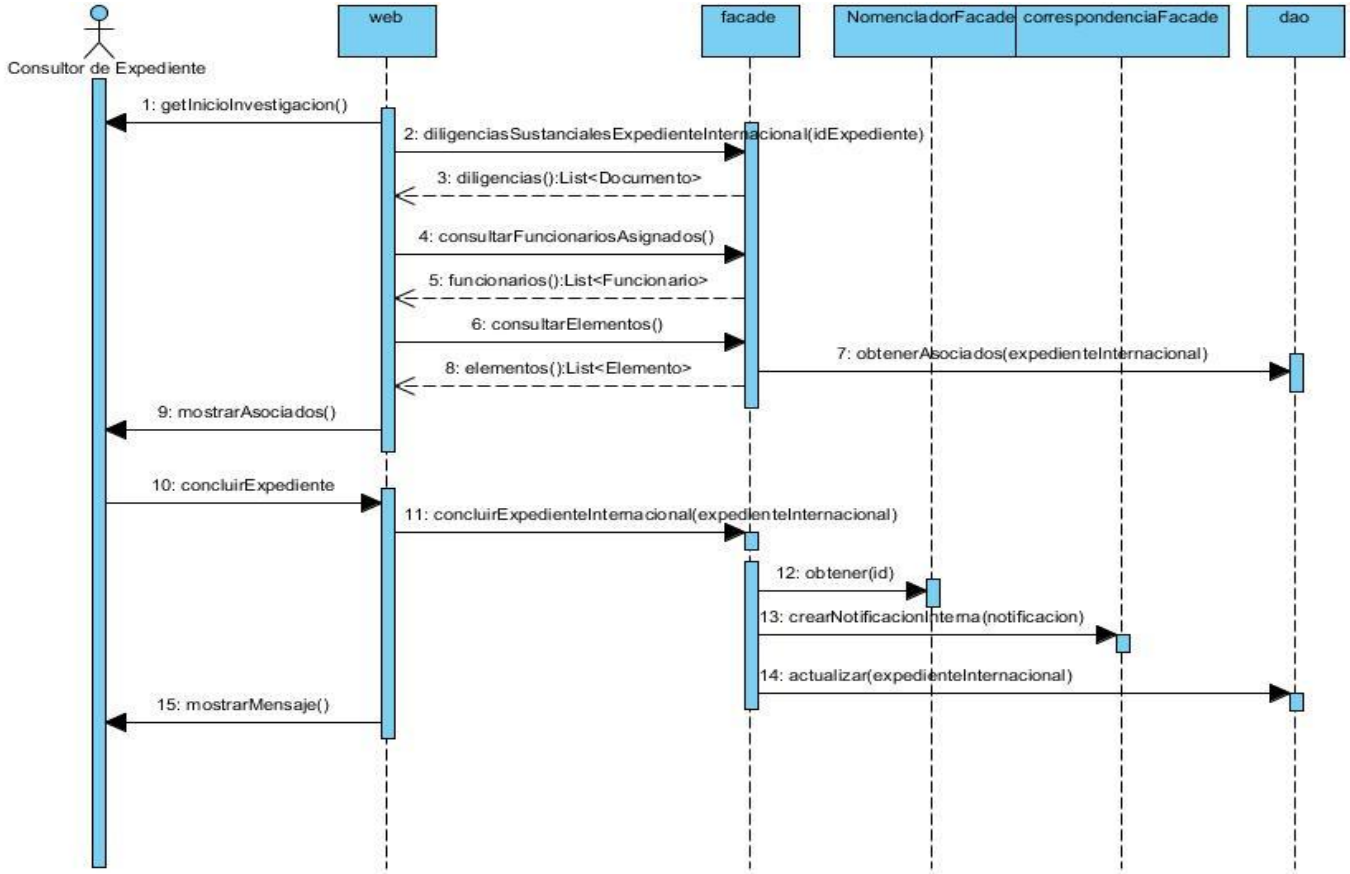


Ilustración 10. CU Ver Expediente de INTERPOL.

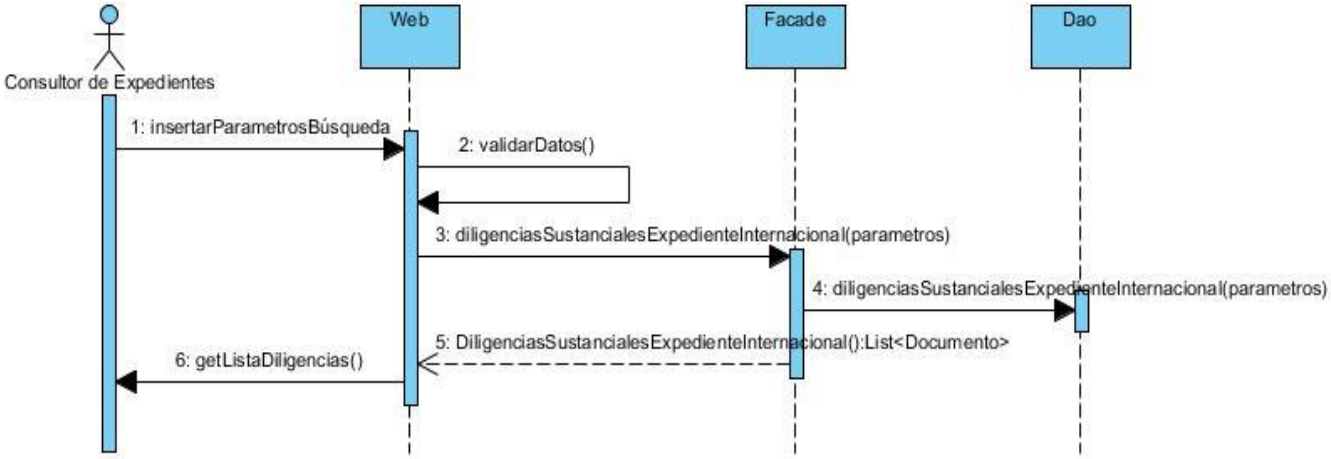


Ilustración 11.CU Consultar Documentos.

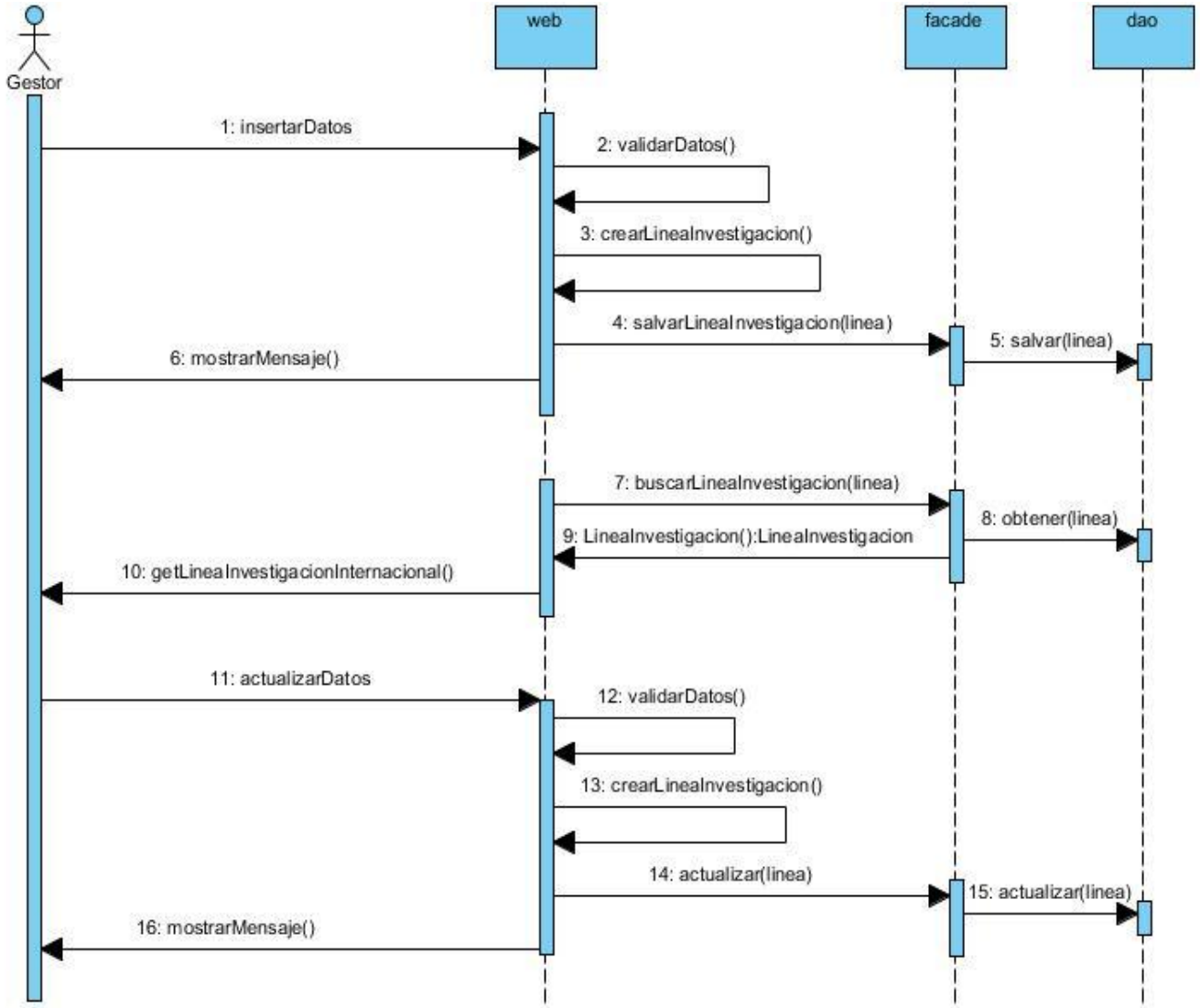


Ilustración 12.CU Gestionar Línea de Investigación

2.3 Modelo de Datos.

Un modelo de datos es un lenguaje utilizado para la descripción de una base de datos**.** Por lo general, un modelo de datos permite describir las estructuras de datos de la base (el tipo de los datos que incluye la base y la forma en que se relacionan), las restricciones de integridad (las condiciones que los datos deben cumplir para reflejar correctamente la realidad deseada) y las operaciones de manipulación de los datos (agregado, borrado, modificación y recuperación de los datos de la base). (39)

2.3.1 Diagrama de Clases Persistentes

El diagrama de clases persistentes ilustra de forma gráfica la organización estructural de las entidades que almacenan los datos asociados al submódulo Investigación Internacional.

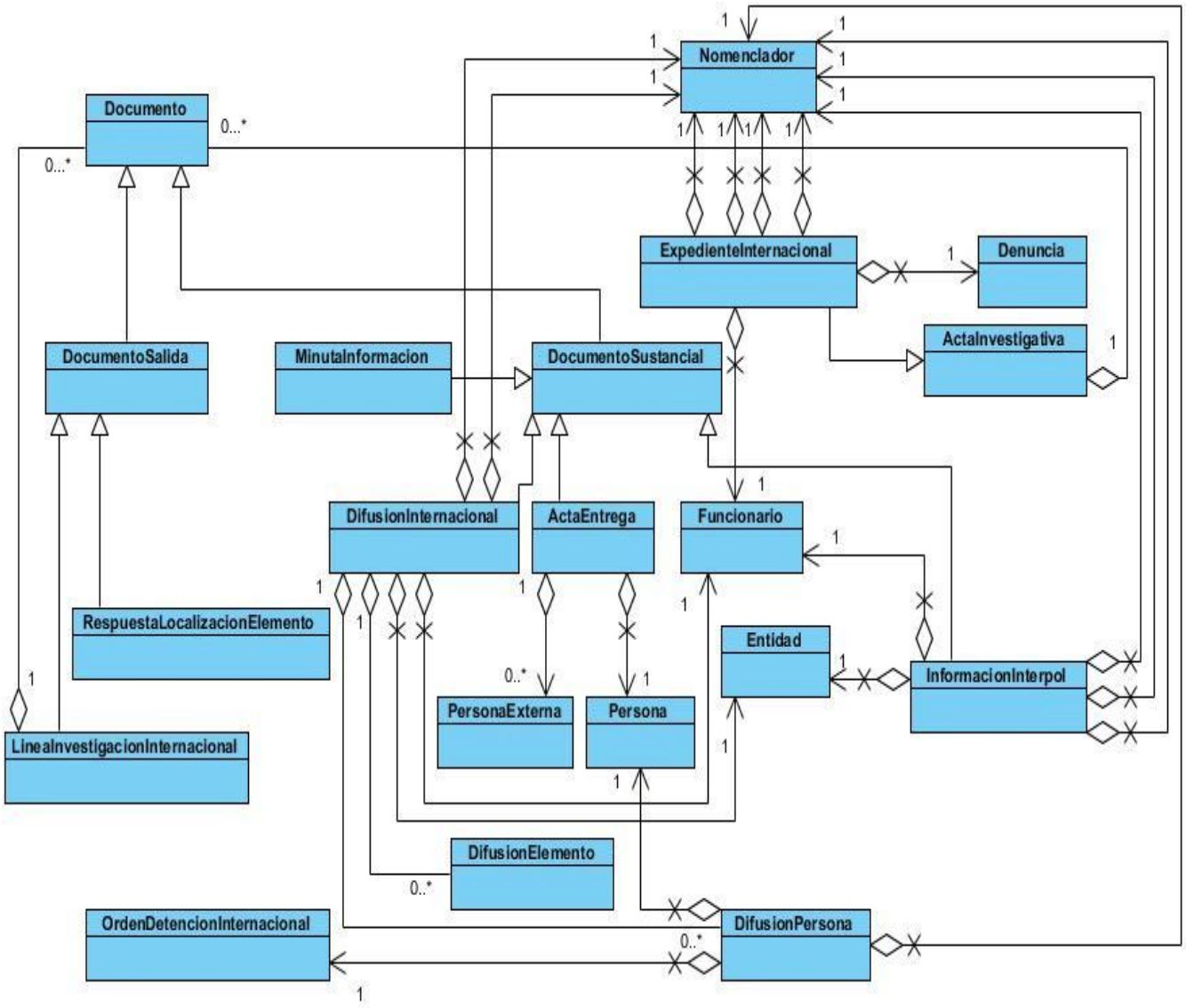


Ilustración 13. Diagrama de clases persistentes

2.3.2 Diagramas de Tablas del Modelo Relacional.

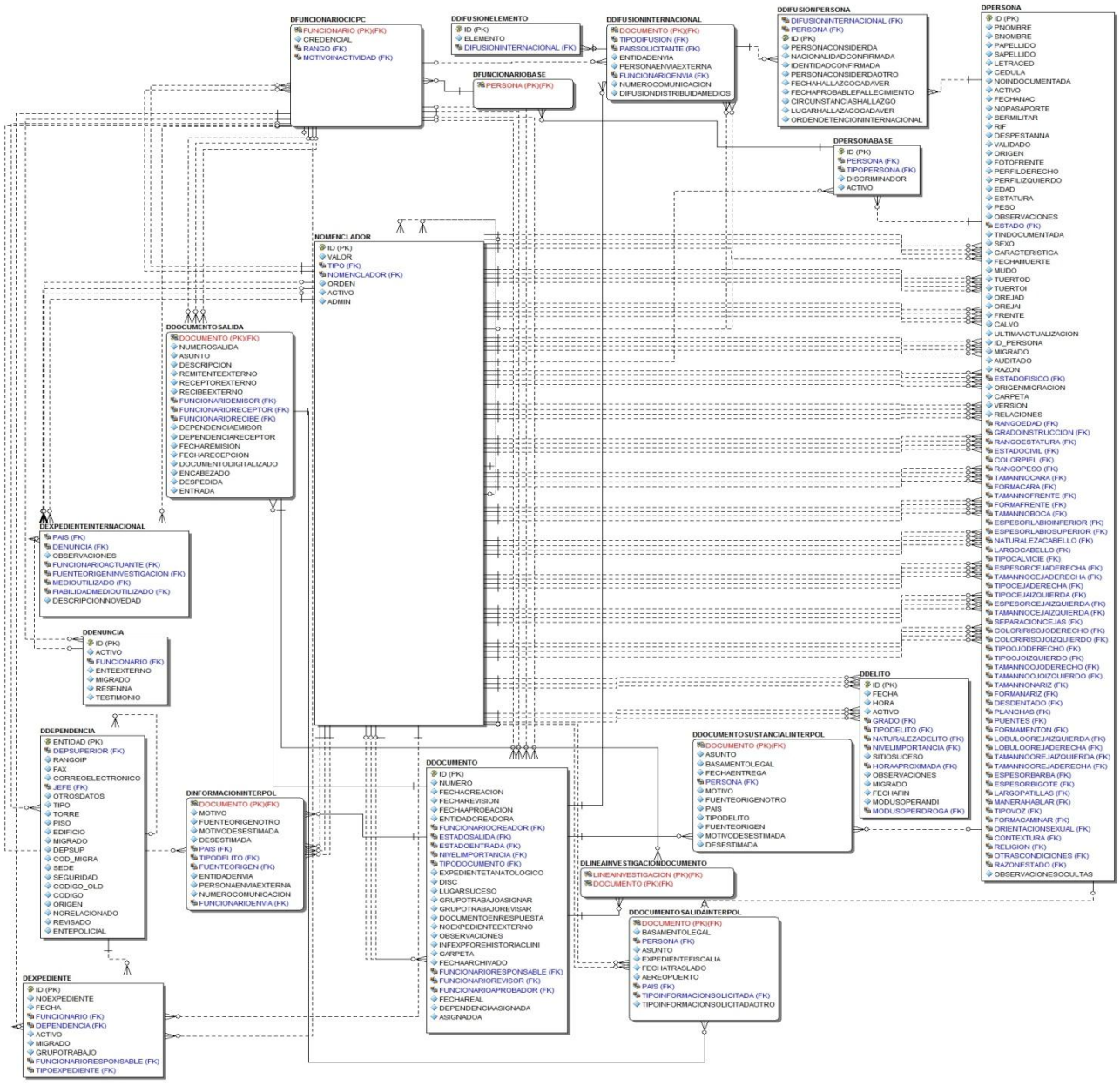


Ilustración 14. Diagrama de tablas del modelo relacional

2.4 Modelo de Implementación.

El modelo de implementación describe como los elementos del modelo de diseño, como las clases, se implementan en términos de componentes, como ficheros de código fuente, ejecutables, etc. El modelo de implementación describe también como se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y como dependen los componentes unos de otros. (12)

2.4.1 Diagramas de subsistemas de implementación.

Los subsistemas de implementación proporcionan una forma de organizar los artefactos del modelo de implementación en trozos más manejables. Un subsistema puede estar formado por componentes, interfaces y otros subsistemas (recursivamente). Además, un subsistema puede implementar -y así proporcionar - las interfaces que representan la funcionalidad que exportan en forma de operaciones. (12)

Los subsistemas de implementación están muy relacionados con los subsistemas de diseño en el modelo de diseño. De hecho, los subsistemas de implementación deberían seguir la traza de uno a uno de sus subsistemas de diseño correspondientes. (12)

Siguiendo esta filosofía una vista de los subsistemas de implementación relacionados al submódulo Investigación Internacional quedaría de la siguiente manera.

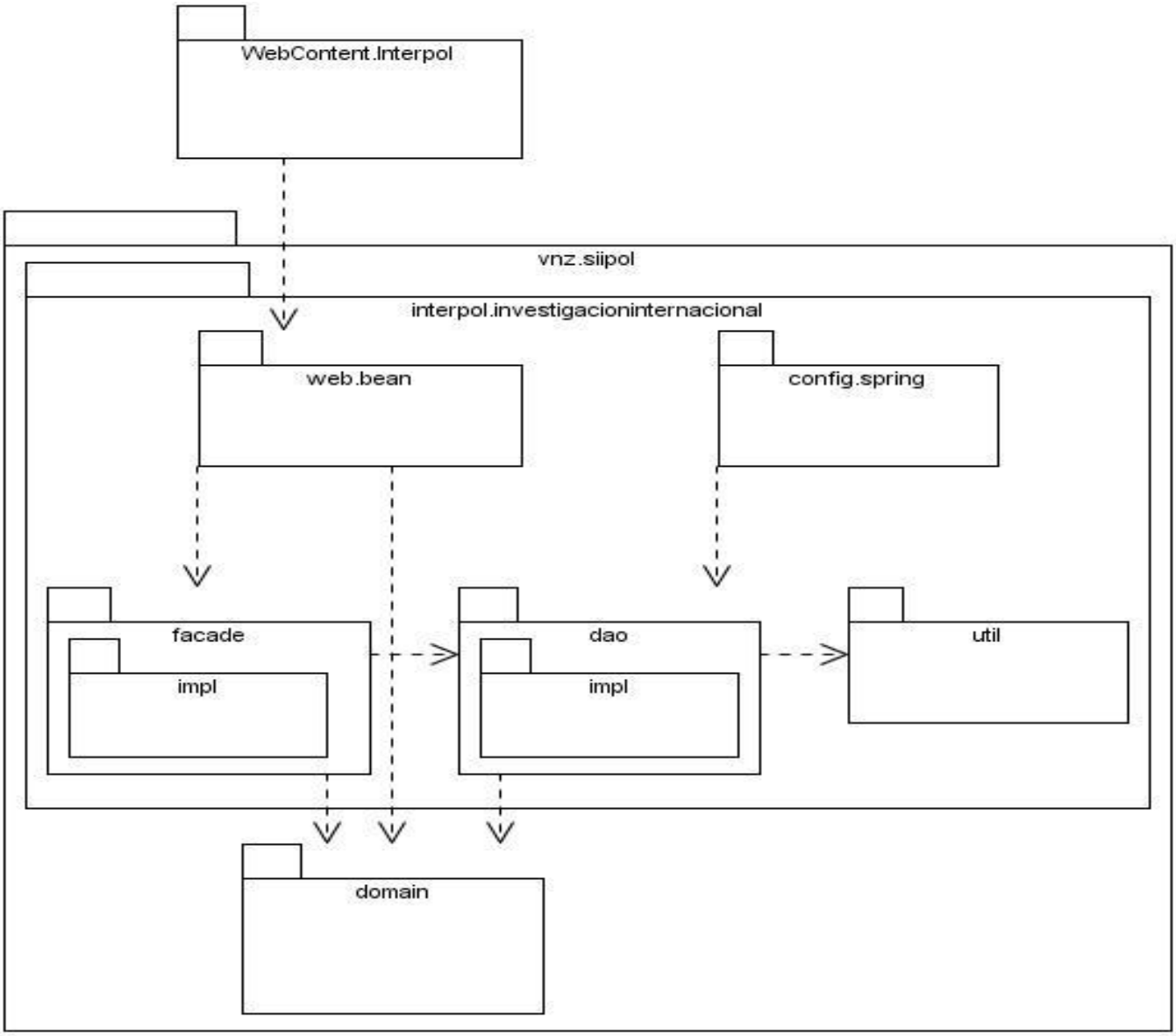


Ilustración 15. Diagrama de subsistemas de implementación

2.4.2 Diagramas de Componentes

Los Diagramas de Componentes ilustran las piezas del software que conformarán un sistema. Un diagrama de Componentes tiene un nivel más alto de abstracción que un diagrama de clase, usualmente un componente se implementa por una o más clases en tiempo de ejecución. Estos son bloques de construcción, como eventualmente un componente puede comprender una gran porción de un sistema. (12)

Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño. De ahí que los componentes tengan relaciones de trazas con los elementos del modelo que implementan (12). El diagrama de componentes que representa las relaciones entre los diferentes componentes como realización del modelo de diseño, se dividirá por problemas de espacio en vistas.

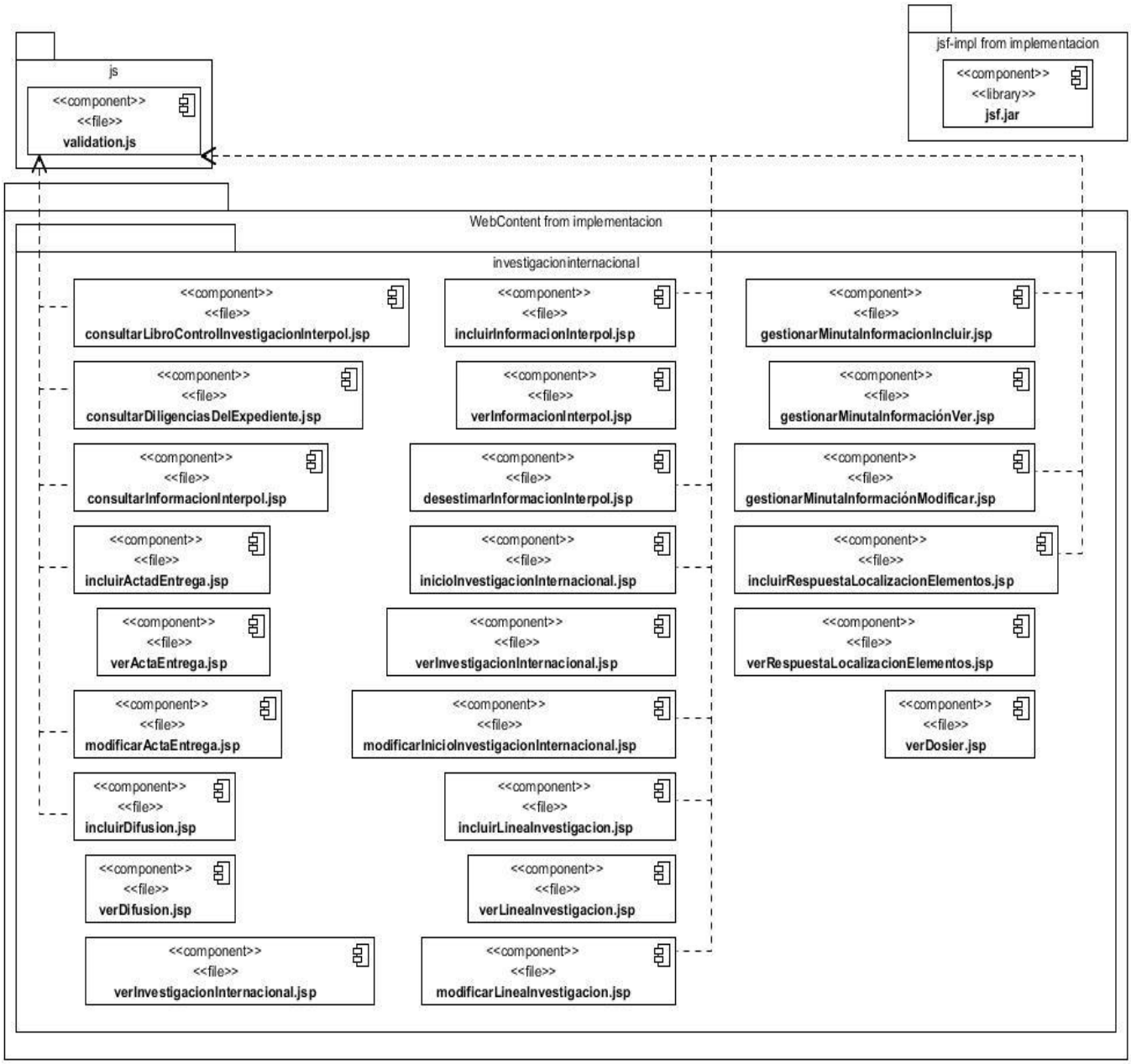


Ilustración 16. Diagrama de componentes. Vista de la capa de presentación. Paginas JSP

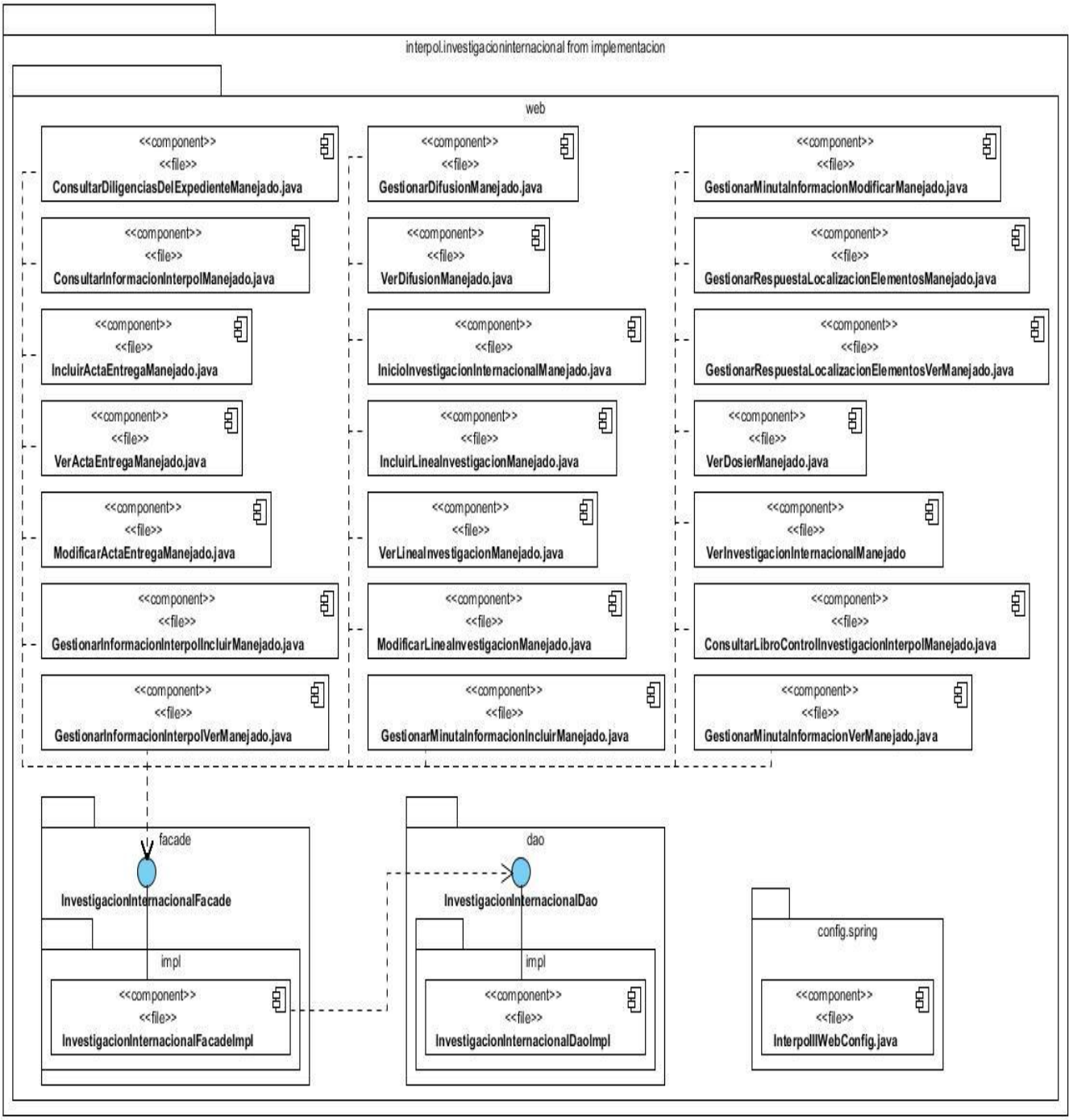


Ilustración 17. Diagrama de componentes. Vista de la capa de presentación, Vista de fachada, y daos

2.6 CONCLUSIONES

En el presente capítulo se desarrolló la solución, que se basa en realizar el diseño y la implementación de la propuesta. Según la metodología seleccionada, fueron generados el modelo de diseño, modelo de datos y modelo de implementación, para la representación de estos modelos, se realizó un estudio de la arquitectura establecida para un mayor entendimiento de las pautas que se deben cumplir. Una vez comprendida la arquitectura se prosiguió a diseñar sobre esta línea, no se consideró necesario realizar el modelo de análisis ya que esta propuesta pertenece a una iteración avanzada de la fase de elaboración donde ya se tenía la visión de las clases de análisis, constituyendo de esta forma la realización de los casos de uso directamente a partir de su descripción textual y no como una traza de las realizaciones del análisis. La utilización de los patrones de diseño como Facade, Domain Model, Front Controller, Singleton, Composite View, Iterator y MVC en el caso de Jsf para el desarrollo de la solución propuesta, formalizó un vocabulario común entre el equipo de desarrollo y proporcionó elementos reusables en el diseño del sistema de software.

CAPITULO 3. VALIDACIÓN DE LA SOLUCIÓN PROPUESTA.

3.1 Introducción

Este capítulo va orientado como su nombre lo indica a demostrar la validez de la solución propuesta para la resolución del problema, para llevar a cabo esta tarea y siguiendo con la metodología establecida para el desarrollo, se hará uso de los artefactos generados durante el flujo de trabajo de pruebas, exponiendo resultados obtenidos por diferentes tipos de pruebas realizadas al submódulo.

La validación de software es de vital importancia ya que proporciona un alto grado de confianza y seguridad no solo en el producto sino en los resultados que se obtienen en implantarlo. Validar el software que se produce es un requisito que debe ser cumplido con rigor.

Las pruebas del software se integran dentro de las diferentes fases del ciclo de RUP, estas se centran principalmente en la evaluación o la valoración de la calidad del producto. A diferencia de otros flujos como requisitos, análisis, diseño e implementación. Su principal objetivo es sacar a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y de esfuerzo. (33)De una manera más desglosada se pueden ver los objetivos de las pruebas como (33)

- La prueba es el proceso de ejecución de un programa con la intención de descubrir un error.
- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
- Una prueba tiene éxito si descubre un error no detectado hasta entonces.

Sin embargo, las pruebas no pueden asegurar ausencia de defectos en el software, solo puede como ventaja secundaria demostrar hasta qué punto las funciones del software parecen funcionar de acuerdo con las especificaciones y de alguna manera, indican la calidad del software en conjunto.

Entre los principales conceptos relacionados a las pruebas se distinguen:

- Métodos de Pruebas.
- Niveles de Pruebas.

3.2 Métodos de Pruebas.

Existen dos métodos básicos para diseñar casos de pruebas, las pruebas de caja negra y las de caja blanca, las cuales han tomado un lugar muy importante en el desarrollo de sistemas de cualquier tipo, tanto que sin dichas pruebas un sistema desarrollado carecería de garantías y credibilidad en sus resultados.

3.2.1 Métodos de Pruebas de Caja Negra.

Las pruebas de caja negra se llevan a cabo sobre la interfaz del software, obviando el comportamiento interno y la estructura del programa.

Se denomina **caja negra** a aquel elemento que es estudiado desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno. En otras palabras, de una *caja negra* nos interesará su forma de interactuar con el medio que le rodea (en ocasiones, otros elementos que también podrían ser *cajas negras*) entendiendo **qué es lo que hace**, pero sin dar importancia a **cómo lo hace**. Por tanto, de una *caja negra* deben estar muy bien definidas sus entradas y salidas, es decir, su interfaz; en cambio, no se precisa definir ni conocer los detalles internos de su funcionamiento. (40)

Los casos de prueba de la caja negra pretenden demostrar que: (40)

- Las funciones del software son operativas.
- La entrada se acepta de forma correcta.
- Se produce una salida correcta.
- La integridad de la información externa se mantiene.

3.3 Niveles de Pruebas.

Los niveles de prueba están relacionados básicamente con la profundidad que cada papel de un proyecto le puede aportar.

Los métodos de pruebas mencionados anteriormente son muy eficientes para lograr la validez de la solución en cuanto a lo especificado en los casos de uso, pero no demuestran totalmente la ausencia de defectos, ni la correcta integración de la solución con el sistema y la arquitectura ya implementada, por lo que a la solución propuesta se le realizaron varios niveles de pruebas las cuales se describen a continuación:

3.3.1 Pruebas de Sistema

El software ensamblado totalmente con cualquier componente hardware que requiera, se prueba para comprobar que se cumplen los requisitos funcionales. Cualquier pieza de software completo, desarrollado o adquirido, puede verse como un sistema que debe probarse, ya sea para decidir acerca de su aceptación, para analizar defectos globales o para estudiar aspectos específicos de su comportamiento, tales como seguridad o rendimiento.

Las pruebas de sistema se dividieron en dos etapas: Pruebas Cruzadas y Pruebas de Calidad Interna.

3.3.1.1 Pruebas Cruzadas

Pruebas realizadas al sistema por el resto de los equipos de desarrollo del proyecto. Se realizaron con el fin de encontrar la mayor cantidad de errores posibles en término de validaciones, pautas definidas por arquitectura de información, formato de los campos, entre otras. Estas pruebas, al ser realizadas por los desarrolladores en papel de probadores, tienen un beneficio extra, y es que el conocimiento del funcionamiento interno del software que tienen estas personas les permite aportar valiosa información al desarrollo y detectar errores altamente complejos antes de que se conviertan en un riesgo.

A continuación se muestra el resumen grafico de cada una de las iteraciones realizadas:

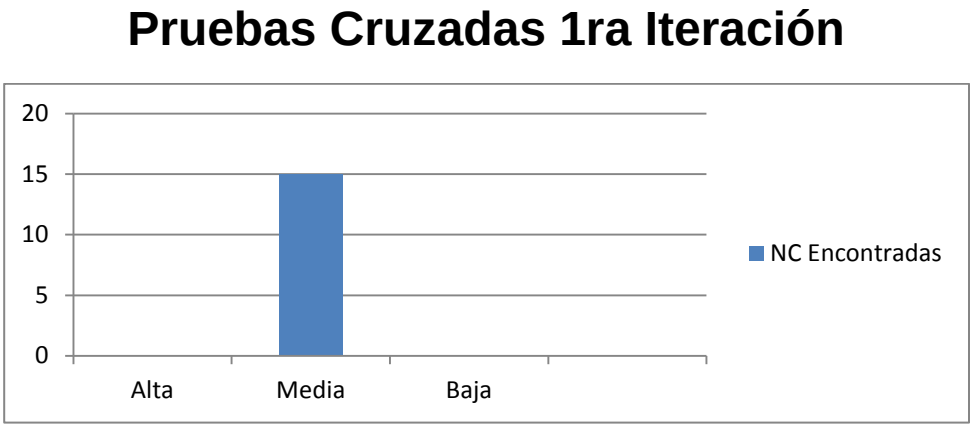


Ilustración 18. NC detectadas durante las Pruebas Cruzadas 1ra Iteración

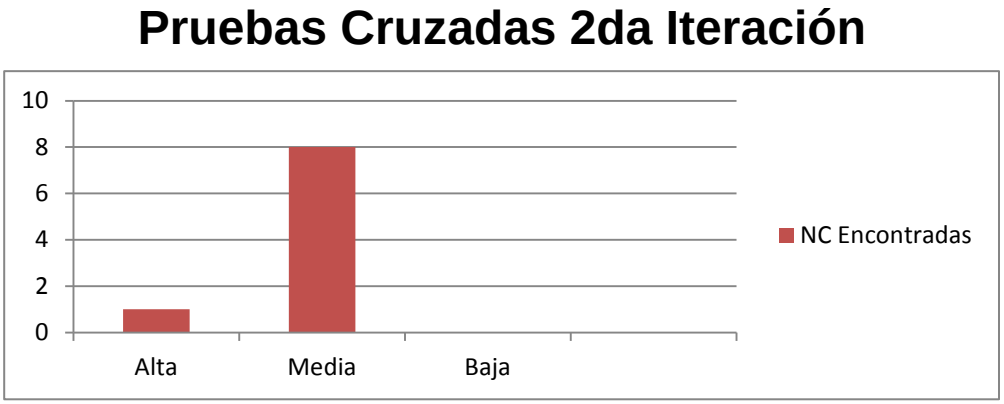


Ilustración 19. NC detectadas durante las Pruebas Cruzadas 2da Iteración

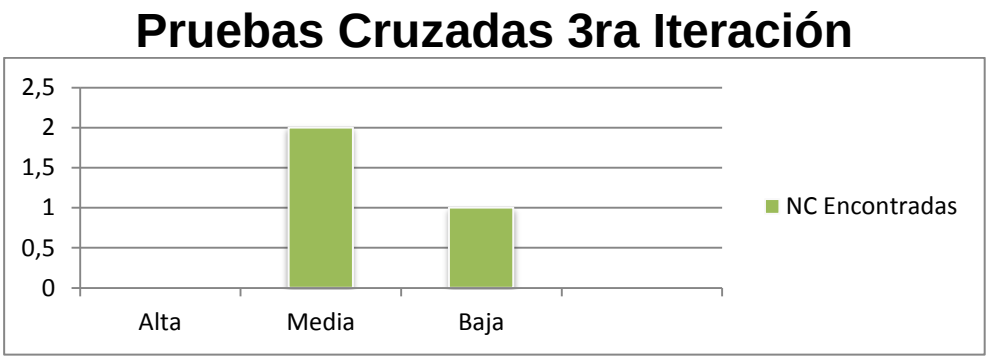


Ilustración 20.NC detectadas durante las Pruebas Cruzadas 3ra Iteración

En una simple comparación de las iteraciones se puede observar como la cantidad de no conformidades van disminuyendo, demostrando que lo implementado corresponde cada vez más a lo especificado

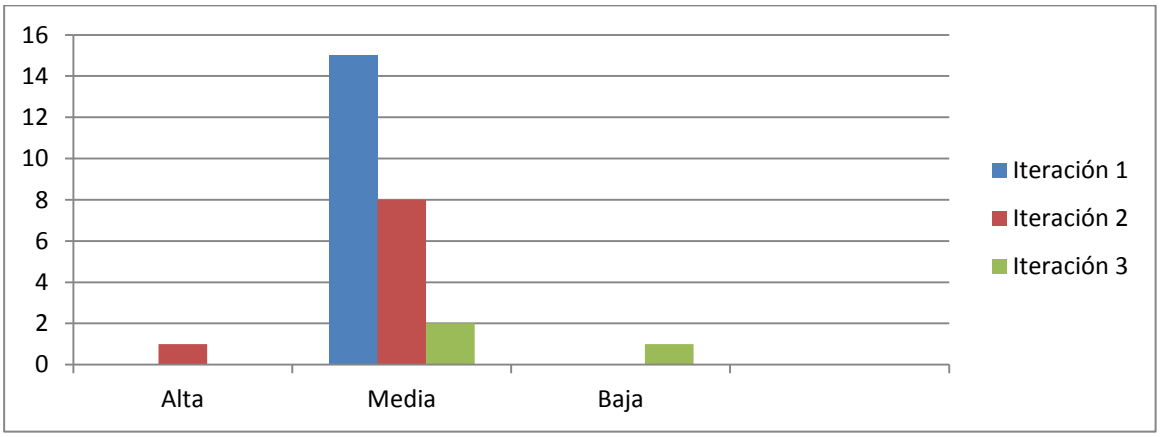


Ilustración 21.Resumen Gráfico de la Pruebas Cruzadas realizadas

3.3.1.2 Pruebas de Calidad Interna

Son las pruebas realizadas al sistema por el equipo de calidad interna del proyecto, con el fin de entregar un producto lo más libre de errores posibles al tercero, encargado de validar la factibilidad del sistema. Se centraron en el cumplimiento de las funcionalidades descritas en el listado de requisitos y de casos de uso.

Se realizaron dos iteraciones de pruebas internas en el proyecto, de las cuales se muestra a continuación el resumen grafico del resultado de las mismas:

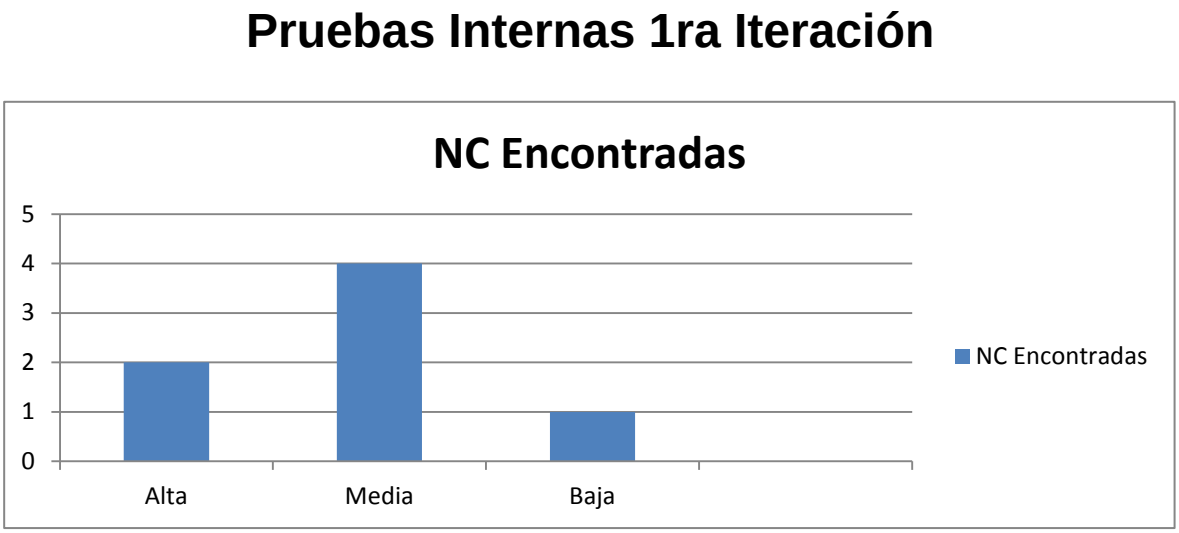


Ilustración 22.NC detectadas durante las Pruebas internas 1ra Iteración

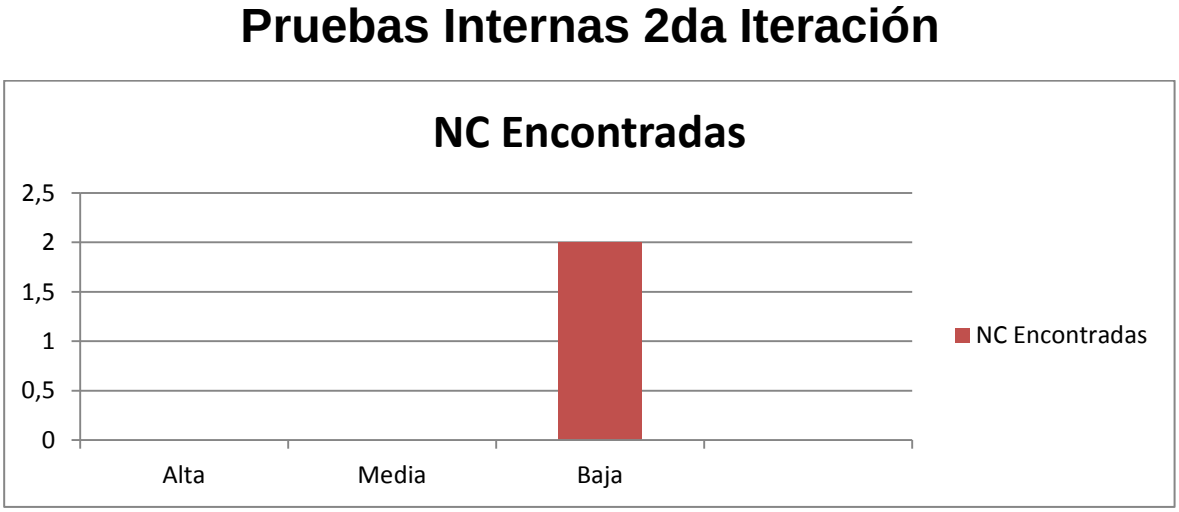


Ilustración 23.NC detectadas durante las Pruebas internas 2da Iteración

Se muestra una comparación de las iteraciones de las pruebas internas que se desarrollaron:

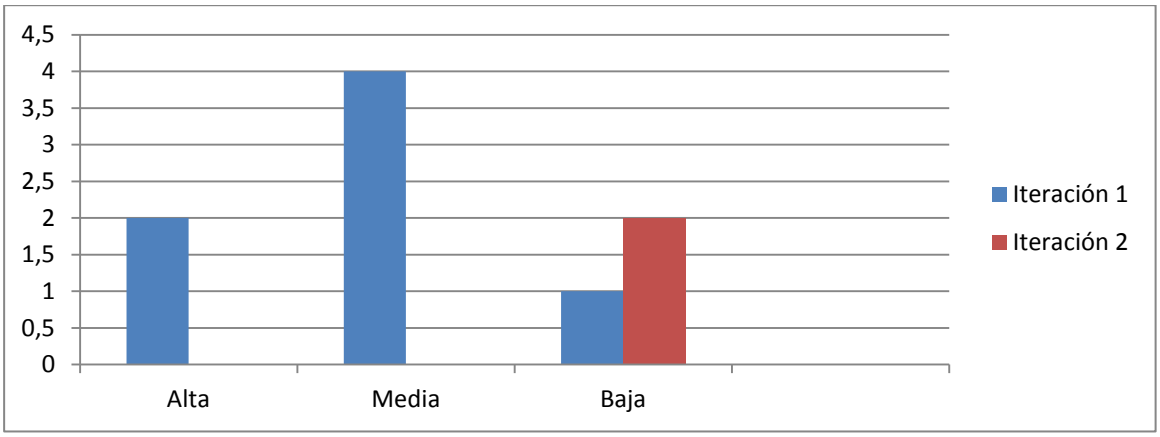


Ilustración 24.Resumen Gráfico de la Pruebas Internas realizadas

3.3.2 Pruebas de Integración

Las pruebas de integración se llevan a cabo durante la construcción del sistema, involucran a un número creciente de módulos y terminan probando el sistema como conjunto, estas pruebas son necesarias por el hecho de que los módulos que forman un software suelen fallar cuando trabajan de forma conjunta, aunque previamente se haya demostrado que funcionan correctamente de manera individual.

Estas pruebas se pueden plantear desde un punto de vista estructural o funcional (41).

Las pruebas estructurales de integración son similares a las pruebas de caja blanca; pero trabajan a un nivel conceptual superior. En lugar de referirnos a sentencias del lenguaje, nos referiremos a llamadas entre módulos. Se trata pues de identificar todos los posibles esquemas de llamadas y ejercitarlos para lograr una buena cobertura de segmentos o de ramas.

Las pruebas funcionales de integración son similares a las pruebas de caja negra. Aquí trataremos de encontrar fallos en la respuesta de un módulo cuando su operación depende de los servicios

prestados por otro módulo. Según nos vamos acercando al sistema total, estas pruebas se van basando más y más en la especificación de requisitos del usuario.

Estas pruebas fueron realizadas por parte del propio equipo de desarrollo del submódulo en cuestión y fueron guiadas a probar las dependencias del submódulo Investigación Internacional con otros módulos como Análisis de Información e Investigación Penal, también se probaron la integración con algunos componentes definidos por el equipo de arquitectura, durante estas pruebas no se generó documentación oficial sin embargo se pueden mencionar algunos de los principales errores detectados y corregidos durante esta etapa.

En el CU Gestionar Inicio de Investigación de Interpol se permite asociar una persona como denunciante llamando al CU Gestionar Persona del módulo Análisis de Información, cuando se seleccionaba la opción de regresar a la vista anterior el sistema no respondía y no ejecuta la operación.

Solución al Problema: Al realizar la llamada al CU Gestionar Persona se pasaba la implementación de la interfaz IGestionarPersonaManejado donde se especificaba a esa funcionalidad como debía llevar a cabo varias operaciones una de ellas la de regresar a la vista anterior mediante el método cancelar donde siempre se regresaba un valor falso y se cambió a devolver un valor verdadero.

En el CU Ver Expediente de Interpol se permite crear una nueva diligencia penal llamando a los CU del módulo Investigación Penal según el tipo de documento que se desea incluir, en el caso particular de incluir una Acta de Investigación Penal mediante el CU Gestionar Acta de Investigación Penal, al seleccionar la opción de incluir una nueva Acta de investigación Penal el sistema lanzaba un error diciendo Ha Ocurrido un error interno, luego de un análisis del mismo se identificó que el Bean Manejado que controlaba la creación de dicha acta no estaba implementando la interfaz EntitySustanciate definida para la sustanciación de expedientes como es el caso, lo cual fue notificado al responsable del módulo correspondiente.

Luego de finalizadas las pruebas realizadas al software y analizando los resultados obtenidos se puede ver como la calidad del producto aumenta con respecto a las especificaciones del mismo, demostrando que el sistema se encuentra listo para ser entregado al cliente o pasar a otra etapa de pruebas superior.

Es importante resaltar que una vez concluida cada iteración de prueba se realizó un análisis por parte del equipo de desarrollo de cada no conformidad detectada, con el fin de verificar que realmente fueron respondidas correctamente y garantizar que en próximas iteraciones no vuelvan a salir las mismas no conformidades detectadas.

El resumen de los errores detectados en cada iteración de pruebas, muestra la correcta evolución del software en cuanto a la reducción de las no conformidades.

3.4 CONCLUSIONES

Tras el análisis de los resultados expuestos en este capítulo, se puede demostrar la solidez de la propuesta basado en el método de pruebas de caja negra y los niveles de pruebas de integración y pruebas de sistema en el paso por varios ciclos. De estas pruebas se obtuvo además de una aplicación con menos errores, una vasta experiencia en el tratamiento de los problemas encontrados, en el modo de trabajar y organizarse. En fin mediante las pruebas se valora cuan correcto y útil es lo que se implementó, y en este caso los resultados lo demuestran. Luego de analizar los resultados satisfactorios obtenidos y habiendo corregido cada no conformidad detectada, se puede decir que el producto final obtenido es un producto con elevada calidad y que da cumplimiento a todos los requisitos funcionales y no funcionales asociados al submódulo Investigación Internacional lo que demuestra la validez de la solución.

CONCLUSIONES GENERALES

El presente documento contiene toda la documentación del proceso de desarrollo del submódulo Investigación Internacional perteneciente al software SIIPOL. Primeramente se realizó un estudio teórico de las características que debía poseer un software de gestión policial y se analizaron algunos de los ya existentes, lo cual arrojó la primera conclusión de la investigación, implementar un sistema nuevo y no utilizar uno de los existentes. Las herramientas y definiciones arquitectónicas puestas en práctica permitieron el desarrollo claro y fluido de un sistema construido sobre bases sólidas y un entorno bien definido. Como propuesta de solución se generan una serie de artefactos, basados en la metodología utilizada (RUP) orientados a elaborar el diseño e implementación del subsistema. Estos artefactos son la prueba del cumplimiento del objetivo general de esta investigación. Los patrones de diseño seleccionados para el desarrollo de la solución propuesta, formalizaron un vocabulario común entre el equipo de desarrollo. La validación de software constituyó un requisito de obligatorio cumplimiento en el proceso de desarrollo de la solución propuesta. Las pruebas realizadas posibilitaron la revisión detallada de cada uno de los componentes desarrollados así como el análisis del comportamiento de la aplicación tras su integración con el resto del sistema. Con esta investigación se obtuvo el submódulo Investigación Internacional integrado al software SIIPOL que da cumplimiento a los requisitos especificados.

RECOMENDACIONES

Identificar componentes conceptuales que sean de utilidad para la creación de uno o varios componentes genéricos enfocados a las funcionalidades provistas por el módulo actual; de forma que dichos componentes puedan ser reutilizados en arquitecturas y/o sistemas de gestión semejantes que tengan necesidades de negocio similares. Estos componentes deben integrarse dentro de la arquitectura como parte del sistema, no como un sistema separado.

Realizar las pruebas de liberación y aceptación del software con el cliente.

Estudiar la posibilidad de realizar un sistema similar para los órganos del MININT de Cuba.

Bibliografía

1.[Online]http://www.acn.com.ve/actualidad/sucesos/item/13203-venezuela-es-el-pa%C3%ADs-m%C3%A1s-violento-del-mundo.html .

2. [Online] http://www.vtv.gob.ve/noticias-nacionales/63444.

3. [Online] http://www.cicpc.gov.ve/objetivos.

4. [Online] http://www.cicpc.gov.ve/mision-vision.

5. [Online] http://www.cicpc.gov.ve/policiainternacional.

6. **Olivia, Maité.** *Ingenieria de Requisitos para el Módulo Interpol del SIIPOL.* 2010.

7. [Online] http://www.hpchile.cl/forense/index.php?option=com_content&view=article&id=382:ique-es-un-sistema-afis&catid=17&Itemid=41.

8. [Online] http://www.dnsp.gob.ve/?q=node/159.

9. [Online] http://www.pol-es.com/pol.html?*session*id*key*=*session*id*val*.

10. [Online] http://www.interpol.int/Public/ICPO/FactSheets/GI03Es.pdf.

11.[Online]http://eva.uci.cu/file.php/102/Curso_20102011/Clases/Semana_02/Seminario_1/Materiales_complementarios/04.Metodologias_de_desarrollo_de_software.pdf.

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

13.[Online]http://eva.uci.cu/file.php/102/Curso_20102011/Clases/Semana_02/Conferencia_3/Materiales_complementarios/Introduccion_a_RUP_y_UML.pdf.

14. [Online] http://www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.

15. [Online] http://www.visual-paradigm.com/product/vpuml/..

16. [Online] http://www.definicion.org/lenguaje-de-programacion.

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

18.[Online]http://java.sun.com/j2ee/overview.html Java 2 Platform, Enterprise Edition (J2EE) Overview.

19. [Online] http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

20. [Online] http://www.sicuma.uma.es/sicuma/Formacion/documentacion/JSF.pdf.

21. [Online] http://www.juntadeandalucia.es/xwiki/bin/view/MADEJA/Ajax4JSF [En línea] 2011.

22. [Online] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=richFacesJsfIntro.

23. [Online] http://pablolg.wikispaces.com/file/view/spring_tutorial_v0.271.pdf.

24. *Spring in Action. Manning, 2005.*

25. *Community Spring Source.* [Online] http://www.springframework.org/documentation.
26. **Rosés Albiol, Francesc.** *Introduccion a Hibernate. 2003.*
27. [Online] http://www.ecured.cu/index.php/IDE_de_Programaci%C3%B3n.
28. **Rodríguez, Msc. Yadiel Ramos.** *DESCRIPCION DE LA ARQUITECTURA DE SOFTWARE.*
29. [Online] http://www.desarrolloweb.com/articulos/840.php.
30. [Online] http://www.rhernando.net/modules/tutorials/doc/bd/oracle.html.
31. [Online] http://www.slideshare.net/Decimo/arquitectura-3-capas.
32. **Larman, Craig.** *UML y Patrones. s.l.: PEARSON, 2003. Segunda Edición.*
33. **Pressman, Roger S.** *Ingeniería de Software un enfoque práctico. Quinta Edición.*
34. [Online] http://www.info-ab.uclm.es/asignaturas/42579/pdf/01-Capitulo1.pdf.
35. [Online] http://www2.elo.utfsm.cl/~elo325/presentaciones/R2_P08.pdf.
36. [Online] http://modelosprogramacion.blogspot.com/2009/05/bridge.html.
37. [Online] http://msdn.microsoft.com/es-es/library/bb972272.aspx#EDAA.
38. [Online] http://www.scribd.com/doc/27613388/Patron-de-DiseNo-Builder.
39. [Online] http://definicion.de/modelo-de-datos/ .
40. [Online] http://indalog.ual.es/mtorres/LP/Prueba.pdf.
41. [Online] http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing.htm#s3.

GLOSARIO DE TÉRMINOS

Ajax4JSF: Es un framework libre y Open Source que adiciona capacidades a las páginas JSF, a través de componentes implementados.

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

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.

CASE: Sigla que corresponde a las iniciales de Computer Aided Software Engineering; y en su traducción al español significa Ingeniería de Software Asistida por Computación.

Rich-Faces: Conjunto de librerías para el framework JSF.

Faces Servlet: Es el Servlet para las componentes visuales del framework JSF.

Framework: Es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

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

Skins: Imagen, conjunto de imágenes o archivo que permite cambiar la apariencia de un programa.