



UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 8

*Análisis, diseño e implementación del sub-módulo  
Dotación de Equipos Policiales perteneciente al  
módulo Registro y Control de SIIPOL*

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

*Autor: Geldri Berroa Álvarez*

*Tutor: Ing. Diana García Vicente*

*Ciudad de La Habana, 2009*

*“Año del 50 Aniversario del Triunfo de la Revolución”*

## DECLARACIÓN DE AUTORÍA

Declaramos que somos los autores de este trabajo y autorizamos a la Facultad 8 de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmamos la presente a los \_\_\_\_ días del mes de \_\_\_\_\_ del año 2009.

Geldri Berroa Álvarez

Diana García Vicente

---

**AUTOR**

---

**TUTOR**

## **AGRADECIMIENTOS**

A mis padres porque sin ellos esto no hubiera sido posible.

A mis abuelas que siempre han deseado poder ver este día.

A mi novia, Mara, por su constante apoyo e incondicionalidad, sin ella hubiera sido mucho más difícil.

A todos mis amigos.

A mi tutora Diana García Vicente que ha contribuido con la preparación del trabajo.

A los que mencioné y a los que se quedaron por mencionar, muchas gracias de todo corazón.

## DEDICATORIA

*A mis padres, los mejores del mundo.*

*Para tía, Aní, Mima y mis amigos, al cumplir mi sueño les cumplo parte de los suyos.*

*A mis hermanos, de los que espero muchas y grandes cosas.*

## RESUMEN

Estos días difíciles en los que se vive, donde cada ser social piensa en su propio bienestar sin importarle cómo y a qué precio consigue lo que necesita para subsistir, existe un profundo y prácticamente incontrolable índice de violencia. Los gobiernos que desean frenarlo, combatirlo y eliminarlo hacen todo lo que está a su alcance, como es el caso de la Hermana República Bolivariana, quien con la perspectiva de que CICPC (Cuerpo de Investigaciones Científicas, Penales y Criminalísticas) debe ser imprescindible para alcanzar la confianza y seguridad de la población realizó un convenio con el gobierno cubano, el cual consiste en desarrollar un nuevo sistema de gestión policial tomando como modelo al SIIPOL (Sistema Integrado de Información Policial. Programa Informático que utilizan los agentes del CICPC en la solución de quejas y delitos reportados), pero con la diferencia de que el nuevo sistema se desarrollará utilizando tecnologías de punta, el tiempo de respuesta a las peticiones de los usuarios será mucho más corto y brindará nuevas funcionalidades que actualmente se realizan de forma manual.

El presente trabajo con título: Análisis, Diseño e implementación del sub-módulo Dotación de Equipos Policiales perteneciente al módulo: Registro y Control de SIIPOL tiene como objetivo informatizar el área que gestiona todo lo referente a los equipos policiales. Entiéndase por todo lo referente a: registrar los equipos policiales con los que cuenta el CICPC, ya sean los que poseen los oficiales o los que se confiscan en las actividades vándalas que se llevan a cabo. Además, se controlan lo que están almacenados, deteriorados y perdidos.

Para modelar, especificar, construir y documentar la aplicación se tuvo en cuenta que los lenguajes, herramientas y tecnologías utilizadas presentaran las propiedades suficientes para obtener un programa informático con las características antes mencionadas, utilizando por ello en su construcción: el Visual Paradigm; el lenguaje de programación Java y los frameworks Java Server Faces, Spring e Hibernate.

## **ABSTRACT**

Nowadays in the world there is a deep and practically uncontrollable level of violence. The governments that wish to stop this situation do whatever they can. The Bolivarian Republic of Venezuela focuses its fight against crime using the CICPC as an indispensable tool to achieve the trust and security of the people. To this effect, it secured a contract with the Cuban government to develop a new police management system taking as a model the SIIPOL (Integrated Police Information System, that is used today by the CICPC agents to solve the crimes reported to them), but with the difference that this new system will develop using state-of-the-art technology, the wait time of the users will be shorter and the new system will bring features that are now executed manually.

The present paper entitled "Analysis, design and implementation of the sub module of Police Equipment" presents as its main objective the automation of the management of the police equipments of the CICPC, being these ones in possession of the officers or those confiscated by them. The system also manages stored, deteriorated and lost equipment.

To model, specify, build and document this application it was necessary to take into consideration that the languages, tools and technologies used presented enough properties to obtain a software product with the aforementioned characteristics, using in its build Visual Paradigm, Java programming language and Java Server Faces, Spring and Hibernate application frameworks.

# ÍNDICE DE CONTENIDOS

<b>INTRODUCCIÓN</b> .....	<b>1</b>
<b>CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA</b> .....	<b>5</b>
<b>1.1. INTRODUCCIÓN</b> .....	<b>5</b>
<b>1.2. SOFTWARE DE GESTIÓN POLICIAL</b> .....	<b>5</b>
<b>1.3. EL CUERPO DE INVESTIGACIONES CIENTÍFICAS, PENALES Y CRIMINALÍSTICAS</b> .....	<b>6</b>
<b>1.4. EL SISTEMA DE INVESTIGACIÓN E INFORMACIÓN POLICIAL</b> .....	<b>7</b>
<b>1.4.1. SUB-MÓDULO DOTACIÓN DE EQUIPOS POLICIALES</b> .....	<b>7</b>
<b>1.4.1.1. MODELO DE CASOS DE USO</b> .....	<b>8</b>
<b>1.4.1.2. DESCRIPCIÓN DE CASOS DE USO</b> .....	<b>10</b>
<b>1.5. METODOLOGÍA, LENGUAJES Y HERRAMIENTAS DE DESARROLLO</b> .....	<b>16</b>
<b>1.5.1. METODOLOGÍA DE DESARROLLO RUP</b> .....	<b>16</b>
<b>1.5.2. LENGUAJE DE MODELADO UML</b> .....	<b>17</b>
<b>1.5.3. VISUAL PARADIGM</b> .....	<b>17</b>
<b>1.5.4. JAVA</b> .....	<b>18</b>
<b>1.6. ENTORNO DE DESARROLLO</b> .....	<b>19</b>
<b>1.7. FRAMEWORKS</b> .....	<b>20</b>
<b>1.7.1. ACEGI SECURITY</b> .....	<b>20</b>
<b>1.7.2. CAPA DE PRESENTACIÓN</b> .....	<b>20</b>
<b>1.8. ARQUITECTURA TÉCNICA</b> .....	<b>21</b>
<b>1.9. CONCLUSIONES</b> .....	<b>22</b>
<b>CAPÍTULO 2. ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE LA PROPUESTA DE SOLUCIÓN.</b> .....	<b>23</b>
<b>2.1. INTRODUCCIÓN</b> .....	<b>23</b>
<b>2.2. MODELO DE ANÁLISIS</b> .....	<b>23</b>

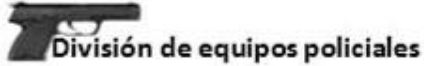
2.3.	MODELO DE DISEÑO.....	27
2.3.1.	DIAGRAMA DE PAQUETES .....	28
2.3.2.	DIAGRAMA DE CLASES DEL DISEÑO .....	30
2.3.3.	DIAGRAMA DE CONTRATO ENTRE PAQUETES .....	52
2.4.	MODELO DE DATOS.....	56
2.5.	MODELO DE IMPLEMENTACIÓN.....	59
2.5.1.	DIAGRAMA DE SUBSISTEMA DE IMPLEMENTACIÓN .....	60
2.5.2.	DIAGRAMA DE COMPONENTES .....	61
2.5.3.	ESTÁNDAR DE CODIFICACIÓN .....	65
2.6.	CONCLUSIONES.....	72
<b><i>CAPÍTULO 3. VALIDACIÓN DE LA SOLUCIÓN PROPUESTA .....</i></b>		<b>73</b>
3.1.	INTRODUCCIÓN.....	73
3.2.	PRUEBAS DEL SOFTWARE .....	73
3.3.	ESTRATEGIAS DE PRUEBAS .....	74
3.4.	NIVELES DE PRUEBAS .....	75
3.5.	CONCLUSIONES.....	79
<b><i>CONCLUSIONES.....</i></b>		<b>80</b>
<b><i>RECOMENDACIONES.....</i></b>		<b>81</b>
<b><i>BIBLIOGRAFÍA .....</i></b>		<b>82</b>
<b><i>GLOSARIO DE TÉRMINOS.....</i></b>		<b>84</b>
<b><i>ANEXOS.....</i></b>		<b>87</b>



# ÍNDICE DE FIGURAS

<i>Fig. 1 Modelo de CU Dotación de Equipos Policiales</i> .....	8
<i>Fig. 2 Modelo de CU Armas Incriminadas</i> .....	9
<i>Fig. 3 Modelo de CU Equipos Policiales</i> .....	9
<i>Fig. 4 Diagrama de Despliegue</i> .....	22
<i>Fig. 5 Modelo de Análisis - Gestionar Ingreso de Armas Incriminadas</i> .....	24
<i>Fig. 6 Modelo de Análisis - Consultar Equipos Policiales</i> .....	25
<i>Fig. 7 Modelo de Análisis - Incluir Solicitud de Munición</i> .....	26
<i>Fig. 8 Modelo de Análisis - Gestionar Dotación de Equipos Policiales a Funcionario</i> .....	27
<i>Fig. 9 Diagrama de Paquetes</i> .....	28
<i>Fig. 10 Diagrama de Clases - Equipos Policiales</i> .....	30
<i>Fig. 11 Diagrama de Clases - Paquete: comun y controller</i> .....	35
<i>Fig. 12 Diagrama de Clases - Asociar Equipos Policiales</i> .....	44
<i>Fig. 13 Diagrama de Clases - Incluir Equipo Policial</i> .....	46
<i>Fig. 14 Diagrama de Clases - Consultar Equipo Policial</i> .....	47
<i>Fig. 15 Diagrama de Clases - Solicitud de Equipos Policiales</i> .....	49
<i>Fig. 18 Contrato entre Paquetes - Equipo Policial</i> .....	52
<i>Fig. 19 Contrato entre paquetes - Armas Incriminadas</i> .....	53
<i>Fig. 20 Contrato entre Paquetes - Atender Solicitud de Equipos Policiales</i> .....	54
<i>Fig. 21 Contrato entre Paquetes - Solicitudes de Equipos Policiales</i> .....	55
<i>Fig. 22 Diagrama de Clases - Armas Incriminadas</i> .....	56
<i>Fig. 23 Diagrama de Clases - Equipos Policiales</i> .....	57
<i>Fig. 24 Diagrama de Clases - Solicitudes de Equipos Policiales</i> .....	58
<i>Fig. 25 Diagrama de Clases - Atención de Solicitudes de Equipos Policiales</i> .....	59
<i>Fig. 26 Subsistema de Implementación</i> .....	60
<i>Fig. 27 Diagrama de Componentes - Armas Incriminadas</i> .....	61
<i>Fig. 28 Diagrama de Componentes - Equipos Policiales</i> .....	62
<i>Fig. 29 Diagrama de Componentes - Solicitudes de Equipos Policiales</i> .....	63
<i>Fig. 30 Diagrama de Componentes - Atención de Solicitud de Equipos Policiales</i> .....	64

<i>Anexo 1 [Interfaz] Solicitud Equipos Policiales a Funcionario .....</i>	<i>87</i>
<i>Anexo 2 [Interfaz] Solicitud Equipos Policiales a Funcionario .....</i>	<i>87</i>
<i>Anexo 3 [Interfaz] Ver Solicitud Asignación a Funcionario .....</i>	<i>88</i>
<i>Anexo 4 [Interfaz] Asignar Dotacion Funcionario .....</i>	<i>88</i>
<i>Anexo 5 [Interfaz] Ver Asignación de Dotación a Funcionario .....</i>	<i>89</i>
<i>Anexo 6 [Interfaz] Consultar Equipo Policial .....</i>	<i>89</i>
<i>Anexo 7 [Diagrama de Secuencia] Consultar Solicitud de Equipos Policiales .....</i>	<i>90</i>
<i>Anexo 8 [Diagrama de Secuencia] Gestionar Solicitud de Equipos Policiales .....</i>	<i>90</i>
<i>Anexo 9 [diagrama de Secuencia] Solicitud de Munición .....</i>	<i>91</i>
<i>Anexo 10 [Diagrama de Clases] Reasignar Equipos Policiales .....</i>	<i>92</i>
<i>Anexo 11 [Diagrama de Clases] Registrar Envió DARFA .....</i>	<i>92</i>



## INTRODUCCIÓN

"Cuando el crimen y la corrupción reinan, y el dinero por droga pervierte la economía, los ciudadanos pierden la confianza en sus líderes e instituciones públicas"<sup>1</sup>. La violencia es un problema social que afecta a miles de personas, principalmente a países latinoamericanos como: Colombia, Honduras, El Salvador, Jamaica y Venezuela. Sin embargo, este mal que afecta a la sociedad es posible prevenirlo a través de decisiones políticas por parte del gobierno, del reforzamiento de sistemas de vigilancia y del diseño de campañas adecuadas a cada país.

Con el objetivo de reducir los índices de violencia en Venezuela, este país cuenta actualmente con varios grupos de investigación contra el crimen, los cuales tienen la tarea de dismantelar las bandas que se dedican a violar y alterar el orden social; además, se encargan de dar respuestas confiables a los hechos cometidos fuera de la ley. Como parte de estos grupos se encuentra el CICPC.

CICPC es una institución que se ocupa de garantizar la validez de las investigaciones científicas que se realizan, asegurando la justicia mediante las acciones penales que se aplican. Este grupo está integrado por funcionarios que poseen gran experiencia, vocación de servicio y una excelente calidad humana. Para garantizar el trabajo existe disponible un SIIPOL, el cual posee una tecnología obsoleta y los servicios que brinda no abarcan todos los procesos que se realizan diariamente en esta institución para combatir, frenar y resolver los crímenes que se cometen. Imposibilitando que los usuarios realicen su trabajo con certeza, facilidad y la rapidez requerida. Además, los procesos que se desarrollan manualmente cuentan con un gran volumen de información que no posee un alto nivel de confiabilidad, necesitándose para el estudio de los crímenes reportados mucho más tiempo del requerido.

Con la perspectiva de que CICPC debe ser indispensable para alcanzar la confianza y credibilidad de la población, tanto nacional como internacional en cuanto a la solución de los problemas delictivos, el gobierno de Venezuela se ha dado a la tarea de resolver, mejorar y superar las carencias que presenta

---

<sup>1</sup> **Granma.** *Preocupa a la ONU violencia y narcotráfico en Latinoamérica.* [Online] [Cited: Julio 10, 2008.] <http://www.granma.cubaweb.cu/2007/05/23/interna/artic29.html>.



esta institución en la actualidad. Para ello, los gobiernos de Venezuela y Cuba firmaron un contrato con el propósito de desarrollar un nuevo Sistema de Gestión Policial, tomando como modelo el sistema que se utiliza actualmente; pero con la diferencia de que el nuevo sistema contará con: tecnologías de punta, tiempos de respuestas más vertiginosos, la interfaz será fácil de utilizar para todos los usuarios, incluso para los que tengan un bajo nivel cultural computacional y además, poseerá nuevas funcionalidades que informatizarán los procesos que se llevan a cabo manualmente en diversas áreas, posibilitando con estas la confiabilidad, integridad y gestión adecuada de la información.

La división de equipos policiales es una de las áreas que no fue incluida en el antiguo sistema SIIPOL y los funcionarios que la integran se encargan de controlar manualmente la información relativa a equipos policiales como: armas, chalecos, municiones, esposas, entre otros. Durante todo el proceso se manejan grandes volúmenes de información que se archivan físicamente, lo que trae como consecuencia que las búsquedas de información que se realicen en los casos delictivos se tornen lentas y engorrosas. También, estos archivos con el tiempo pueden deteriorarse o perderse, provocando que no exista un control adecuado de los equipos que se almacenan en la bóveda del CICPC. Además, se incurre en atrasos para la solución de los delitos, pérdida innecesaria del tiempo de los trabajadores y gastos monetarios en la compra de material de oficina para el archivo de la información.

Debido a los problemas antes mencionados se ha decidido automatizar la División de Equipos Policiales y para ello se cuenta con la modelación del negocio y los requisitos funcionales y no funcionales con los que debe cumplir el sub-módulo; siendo el **problema a resolver**: ¿Cómo desarrollar un subsistema que garantice el cumplimiento de los requisitos funcionales y no funcionales del sub-módulo División de Equipos Policiales del SIIPOL?

Además, **se realizará un estudio (objeto de estudio)** de los requisitos funcionales y no funcionales del módulo Registro y Control de SIIPOL; **quedando enmarcado específicamente (campo de acción)** en los requisitos funcionales y no funcionales del sub-módulo Dotación de Equipos Policiales del módulo Registro y Control de SIIPOL.

Como **objetivo general** de la investigación se plantea desarrollar un subsistema para la informatización de los procesos de gestión de equipos policiales en el CICPC, definiendo como **objetivos específicos**:



- 1- Modelar los diagramas correspondientes a los flujos de trabajo: Análisis, Diseño e Implementación.
- 2- Implementar el sub-módulo Dotación de equipos policiales.

Para darle cumplimiento a estos objetivos y alcanzar los resultados requeridos se realizarán las siguientes **tareas de investigación:**

- 1- Estudiar los principales sistemas existentes que brinden servicios de gestión de información policial.
- 2- Realizar los estudios correspondientes sobre la arquitectura a utilizar para desarrollar el sistema SIIPOL.
- 3- Analizar los patrones de diseño que pueden aplicarse en el desarrollo del sub-módulo.
- 4- Describir la metodología, los lenguajes y herramientas que se utilizarán en la elaboración del subsistema.
- 5- Estudiar el modelo de casos de uso del negocio; así como la descripción detallada de los mismos.
- 6- Confección de los distintos diagramas de diseño e implementación.
- 7- Codificación.

Y como preguntas científicas se plantean:

1. ¿Qué mecanismos y patrones de diseño son factibles utilizar para el diseño de la solución final?
2. ¿Cuáles serían las buenas prácticas de programación conjugadas con los patrones de diseño que permitan implementar una solución eficaz?

Con la automatización del sub-módulo: División de Equipos Policiales perteneciente al nuevo Sistema Integrado de Información Policial se esperan ofrecer las siguientes mejoras:

- 1- Seguridad y control de la información que se maneja. Ya que se llevará un control estricto de todos los equipos policiales que se encuentren almacenados; evitando así cualquier actividad delictiva (toma de los equipos sin autorización, ya sean para usos personales o para deshacerse de los que consistan pruebas de actos ilegales) que pueda realizarse.



2- Aprovechamiento fructífero del tiempo por parte de los funcionarios del CICPC. Porque cuando se reporte un crimen con armas de fuego y se haga posesión de esta, no será necesario buscar en todos los archivos a verificar si está registrada y a quien pertenece.

La estructuración del trabajo de diploma está compuesta por 3 capítulos que contienen en el siguiente orden el resultado de la investigación.

**Capítulo 1 Fundamentación Teórica:** Se refleja el análisis de las tecnologías, lenguajes y herramientas que serán empleadas en el desarrollo del sub-módulo y el estudio de los principales sistemas existentes que brinden servicios de gestión de información policial.

**Capítulo 2 Análisis, Diseño e Implementación de la propuesta de solución:** Se representa el modelo de clases del diseño. Además, se reflejan los elementos y funcionalidades que poseerá el subsistema siguiendo la metodología RUP, reconocida ampliamente en todo el mundo para el desarrollo de proyectos de calidad y se muestran los diagramas de componentes.

**Capítulo 3 Validación de la Solución Propuesta:** Se hace referencia a los tipos de pruebas que se pueden aplicar a la propuesta de solución desarrollada. Además, se hace una descripción detallada de los test realizados y, mediante los mismos se plantean las posibles mejoras a realizar en el subsistema.

## CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

### 1.1. INTRODUCCIÓN

En el presente capítulo se refleja una breve descripción de varios sistemas existentes que poseen funcionalidades para llevar un control sobre la información que se gestiona en una institución policial. Se exponen además, mediante una breve descripción, los principales conceptos que sustentan la investigación, y se muestra el estado del arte de las tecnologías, lenguajes y herramientas a emplear en el desarrollo del proyecto CICPC.

### 1.2. SOFTWARE DE GESTIÓN POLICIAL

#### ***STEGPOL:***

Es un Sistema de Información Geográfica que actualmente opera en Chile, sobre una Plataforma Nacional Común de Información, aplicada al "Sistema de Emergencias Nacionales" y al "Sistema Territorial de Gestión Policial". Se caracteriza por ser un sistema con tecnología de punta, que identifica de forma efectiva dónde geográficamente se están cometiendo o se han cometido actos delictivos a nivel territorial, conjuntamente apoyado por sistemas de información en-línea desde el lugar de los hechos; lo cual permite la acción rápida y coordinada entre los diferentes actores encargados de la seguridad ciudadana de ese país. La utilización del sistema contribuye directamente a lograr una mayor eficiencia en las gestiones de seguridad ciudadana, lo cual beneficia la calidad de vida y bienestar social de Chile.

#### ***SIGEP:***

Es uno de los proyectos surgidos a raíz de la Alternativa Bolivariana para los Pueblos de América (ALBA), desarrollado por un grupo de estudiantes y profesores de la Facultad 4 en la Universidad de las Ciencias Informáticas (UCI).

Sus funcionalidades están relacionadas con las prisiones venezolanas y aunque SIGEP enfoca sus requisitos en un área distinta a la de CICPC, su arquitectura se utilizó como punto de partida en las investigaciones del proyecto, siendo el conjunto de herramientas y tecnologías que se utilizan en ambos

sistemas similares; debido a que tanto uno como el otro brindan servicios referentes a combatir, controlar y solucionar acciones delictivas que se cometan.

### **SIIPOL:**

El Sistema Integrado de Información Policial es una aplicación informática utilizada por el Cuerpo de Investigaciones Científicas, Penales y Criminalísticas de la República Bolivariana de Venezuela para llevar a cabo las tareas fundamentales orientadas a garantizar la seguridad nacional. El SIIPOL está diseñado sobre una tecnología obsoleta: desarrollado con el lenguaje de programación Adabas-Natural y en servidores de base datos SUN 6500.

Después de realizar un estudio y teniendo en cuenta las características de los sistemas descritos anteriormente, se determinó que a pesar de que sirven como apoyo a distintas instituciones relacionadas con los temas policiales, sus objetivos y alcances no se encuentran enfocados a las necesidades del CICPC.

Por otro lado, el actual sistema SIIPOL no abarca los procesos relacionados con la División de Armamento, lo cual influye de forma negativa en el buen desempeño de los funcionarios a la hora de esclarecer hechos delictivos que involucran el manejo de los equipos policiales. Con respecto a este obsoleto sistema, la nueva aplicación se desarrollará con tecnologías y herramientas de punta, contribuyendo decisivamente en la calidad de la misma, donde toda la información que se gestione será íntegra, segura y confiable.

### **1.3. EL CUERPO DE INVESTIGACIONES CIENTÍFICAS, PENALES Y CRIMINALÍSTICAS**

La hermana República Bolivariana cuenta actualmente con diversos grupos de investigación, los cuales tienen como objetivo reducir el índice de violencia, reprender a los ciudadanos que se dedican a infringir y alterar el orden social y dar respuestas contundentes a los hechos delictivos que se cometen. Dentro de estos grupos se encuentra el (CICPC), institución que se ocupa de garantizar la validez de las investigaciones científicas que se realizan, asegurando la justicia a través las medidas penales que se aplican.



Para garantizar el trabajo del CICPC existe disponible un Sistema Integrado de Información Policial; pero posee algunas deficiencias, como son: la posesión de tecnologías obsoletas y los servicios que brinda no abarcan todos los procesos que se realizan en la institución diariamente; por lo que varias actividades que son imprescindibles en la solución de hechos delictivos se realizan manualmente; siendo esta deficiencia un grave problema para la rápida solución a las denuncias que se registran.

#### **1.4. EL SISTEMA DE INVESTIGACIÓN E INFORMACIÓN POLICIAL**

Debido a las ineficiencias que posee el actual SIIPOL se decidió realizar un nuevo sistema que corrija dichos errores y posea todas las funcionalidades que se requieren en el CICPC, posibilitando así, resolver con la calidad y rapidez necesaria los delitos cometidos. También, dicho sistema será confeccionado utilizando tecnologías de punta; trayendo como resultado compatibilidad con las nuevas tecnologías que se apliquen en el CICPC e invulnerabilidad en la integridad y confidencialidad de la información que se gestiona.

##### **1.4.1. SUB-MÓDULO DOTACIÓN DE EQUIPOS POLICIALES**

El sub-módulo: Dotación de Equipos Policiales tiene como objetivo informatizar el área: División de Equipos Policiales del CICPC; ya que el antiguo sistema no brindaba los servicios a dicha aérea. Con la incorporación de este sub-módulo al sistema se podrán registrar y gestionar adecuadamente todos los equipos policiales que se encuentren almacenados y los que son embargados en acciones vándalas. Además, el tiempo de solución de los problemas reportados será mucho más corto, no se infringirán en gastos de material de oficina y el resultado del trabajo realizado por los agentes del CICPC será más exacto y efectivo; ya que se ahorrarán el tiempo de buscar en archivos (en muchas ocasiones borrosos) los datos necesarios para resolver los casos que se les asignan.

1.4.1.1. MODELO DE CASOS DE USO

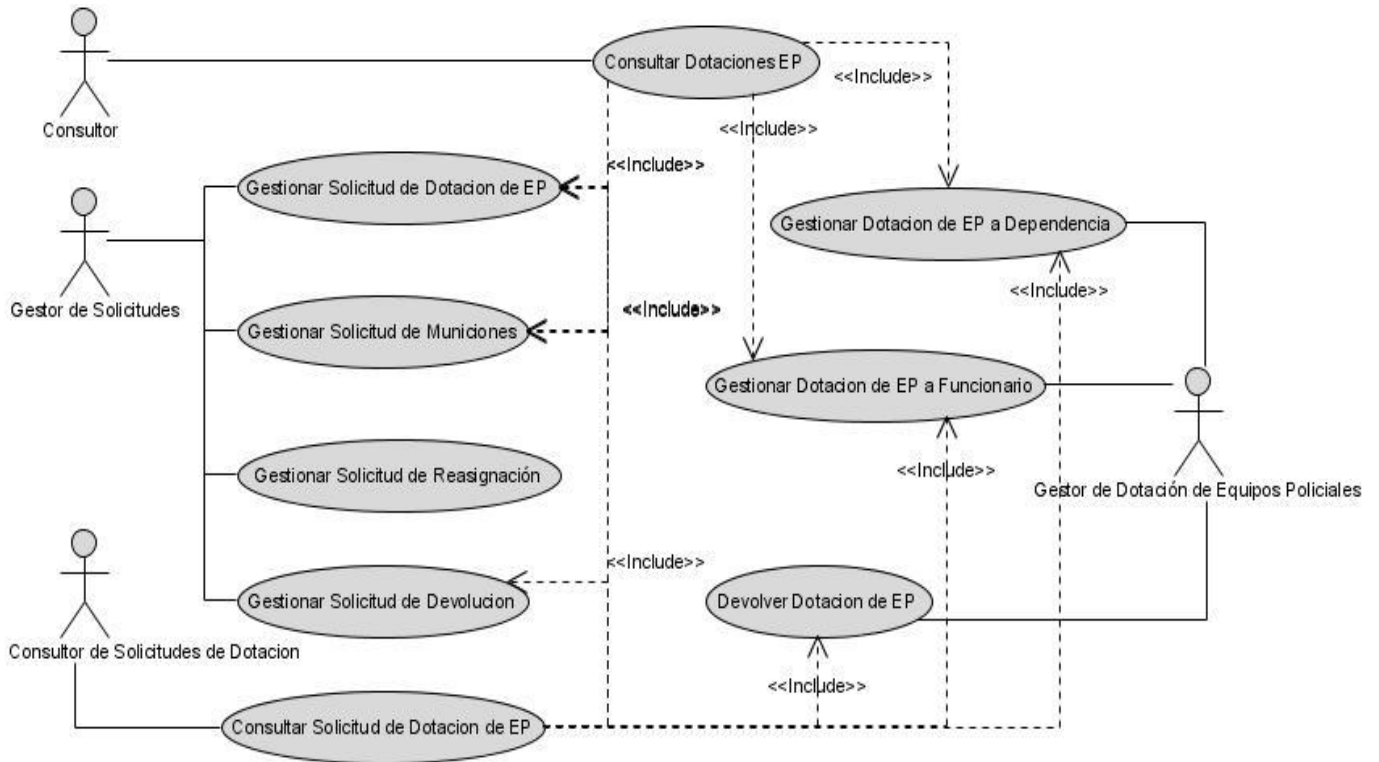


Fig. 1 Modelo de CU Dotación de Equipos Policiales

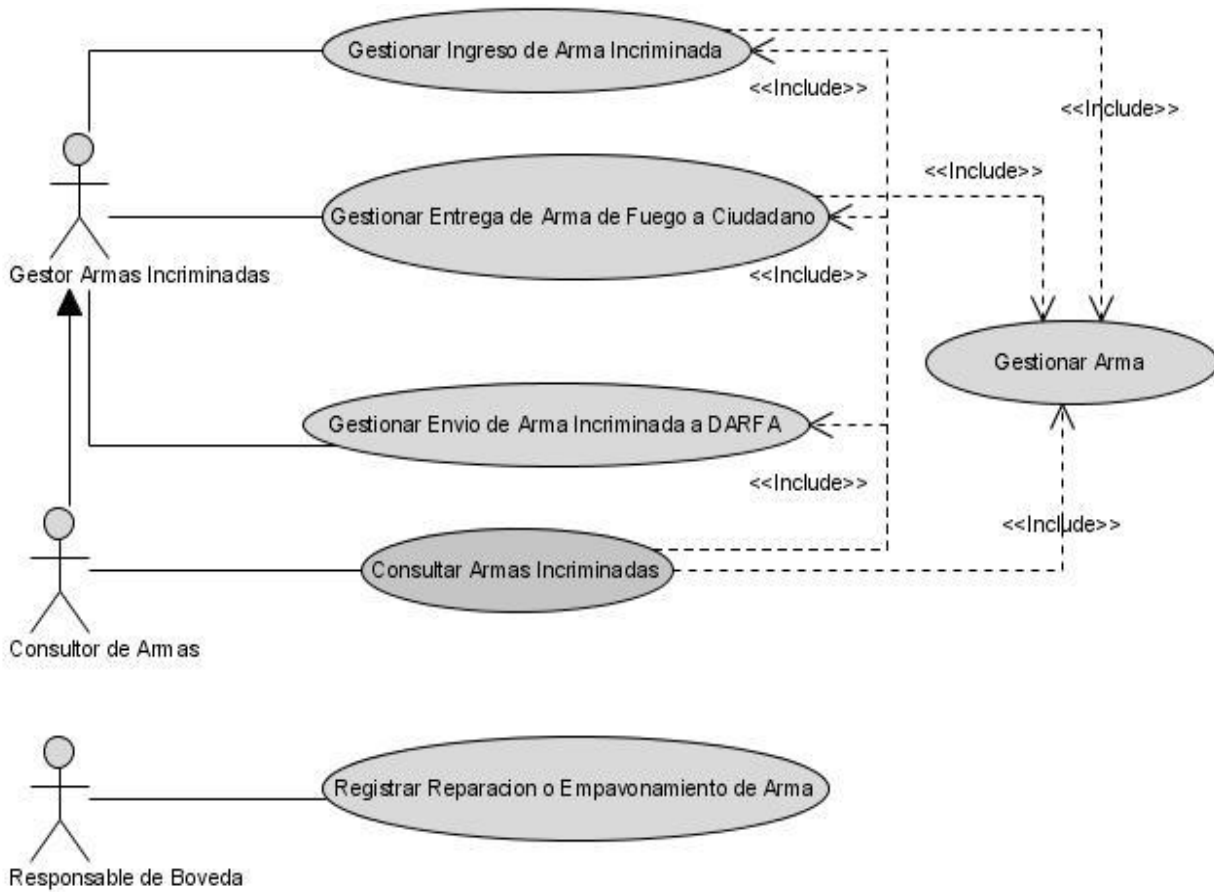


Fig. 2 Modelo de CU Armas Incriminadas

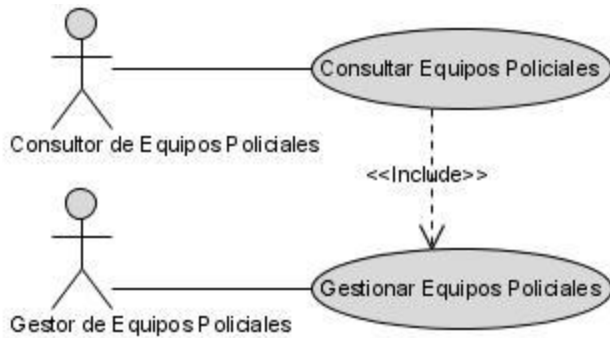


Fig. 3 Modelo de CU Equipos Policiales

### 1.4.1.2. DESCRIPCIÓN DE CASOS DE USO

Caso de Uso	
	Gestionar Solicitud de dotación de equipos policiales
propósito	Incluir, ver los datos de una solicitud de equipos policiales a una dependencia o un funcionario.
Actores: Gestor de Solicitudes	
<p>Resumen: El caso de uso se inicia cuando el Gestor de Solicitudes selecciona la opción que le permite realizar una acción sobre una Solicitud de Dotación de Equipos Policiales. El actor puede incluir una Solicitud de Dotación de Equipos Policiales a Funcionario o una Solicitud de Dotación de Equipos Policiales a Dependencia, y puede ver dichas solicitudes. En caso de que seleccione la opción de incluir una Solicitud de Dotación de Equipos Policiales, ya sea a Funcionario o a Dependencia, el sistema dará la posibilidad de insertar o seleccionar los datos que se necesitan para llenar la plantilla. Si el actor elige la opción de ver una Solicitud de Dotación de Equipos Policiales el sistema mostrará el contenido de la solicitud en cuestión. El caso de uso termina.</p>	

Caso de Uso	
	Consultar Solicitud de dotación de equipos policiales
Propósito:	Buscar y listar de manera ordenada un resumen de los datos de una Solicitud de Dotación de Equipos Policiales coincidentes con uno o varios criterios de búsqueda.
Actores: Consultor de Solicitudes de Dotación.	

Resumen: El caso de uso se inicia cuando el actor selecciona la opción de consultar una Solicitud de Dotación de Equipos Policiales desde la Agenda de Trabajo. El sistema brinda la posibilidad de introducir los datos elementales de búsqueda para realizar la consulta. El actor introduce los datos y el sistema busca los resultados y muestra una lista de posibles coincidencias. El sistema permite imprimir los resultados obtenidos terminando así el caso de uso.

Caso de Uso	
	Devolver dotación de equipos policiales.
Propósito:	Devolver una Dotación de Equipos Policiales asignada.
Actores: Gestor de Dotación de Equipos Policiales.	
<p>Resumen: El caso de uso se inicia cuando el actor accede a la opción de devolver una Dotación de Equipos Policiales en respuesta a una solicitud. El sistema muestra una vista con los datos de la Dotación de Equipos Policiales. El actor selecciona la opción registrar Devolución y automáticamente se desvincula la dotación, o sea de estado asignado pasa a estado devuelto. En caso de que el funcionario sea fallecido, el actor introduce los datos necesarios para confirmar la identidad del Portador de equipos policiales y accede a la opción de efectuar la devolución. El sistema emite la constancia de devolución individual del equipo policial y la solvencia en el caso que la devolución se efectúe por completo, permitiendo imprimir y exportar a PDF. El sistema registra una devolución dentro de las Novedades y el caso de uso termina.</p>	

Caso de Uso	
	Gestionar Ingreso de Arma Incriminada.
Propósito:	Incluir o ver el Ingreso de Armas Incriminadas.

Actores: Gestor de Arma Incriminada.
Resumen: El caso de uso se inicia cuando el actor selecciona la opción que le permite realizar una acción sobre el Ingreso de Armas Incriminadas. El actor puede incluir y ver los datos del Ingreso de Armas Incriminadas. En caso de que seleccione la opción de incluir Armas Incriminadas, el sistema brinda la posibilidad de insertar los datos que se necesitan para llenar esta información. Si el actor elige la opción de ver datos del Ingreso de las Armas Incriminadas, el sistema muestra el contenido del registro en cuestión. El sistema permite imprimir y exportar a PDF, terminando así el caso de uso.

Caso de Uso	
	Consultar Arma Incriminada.
Propósito:	Buscar y listar de manera ordenada un resumen de los datos de un Arma Incriminada coincidentes con uno o varios criterios de búsqueda.
Actores: Consultor de Armas.	
Resumen: El caso de uso se inicia cuando el actor solicita Consultar Arma Incriminada. El sistema permite introducir los datos identificativos del Arma y opcionalmente, seleccionar un rango de fecha aproximado de arribo a la bóveda, para realizar la consulta. El sistema consulta y muestra un listado de posibles coincidencias ordenadas por el serial primario, permitiendo imprimir o exportar a PDF la vista mostrada, registra la acción como una Novedad y el caso de uso termina.	

Caso de Uso	
	Registrar Reparación o Empavonamiento de arma.
Propósito:	Registrar la Reparación o el Empavonamiento de un Arma de un funcionario del CICPC o la entrega de un arma una vez reparada o empavonada.
Actores: Responsable de Bóveda.	
<p>Resumen: El caso de uso se inicia cuando el Responsable de la Bóveda selecciona la opción Registrar Reparación o Empavonamiento de un Arma de un funcionario del CICPC. El Responsable de la Bóveda consulta los datos del funcionario portador del arma y el sistema muestra un listado con las armas asignadas al funcionario. Seguidamente el Portador entrega el arma a reparar o empavonar y recibe el comprobante de entrega de parte del Responsable de la Bóveda, finalizando así el caso de uso. En caso de que sea una entrega del arma el Responsable de la Bóveda introduce el número del comprobante, el sistema consulta y muestra el comprobante. El Responsable de la Bóveda finaliza la reparación o empavonamiento, finalizando así el caso de uso.</p>	
<p>Requisitos Suplementarios relevantes:</p> <ul style="list-style-type: none"> <li>• El sistema debe generar de manera automática el número del comprobante, donde los dos primeros números son el año y los 4 siguientes son un consecutivo, por ejemplo 08-0001.</li> </ul>	

Caso de Uso	
	Gestionar Equipos Policiales.
Propósito:	Crear, ver, modificar o asociar un arma, chaleco, esposas u otro equipo policial.
Actores: Gestor de Equipo Policial	
<p>Resumen: El caso de uso se inicia cuando el actor accede a la opción de realizar una operación sobre un equipo policial. Si la opción seleccionada es crear un equipo policial el sistema brinda la posibilidad de introducir los datos conocidos del mismo. En caso de modificar los datos de uno de estos equipos policiales, el sistema muestra todos los datos conocidos del mismo y brinda la posibilidad de añadir nuevos datos o modificar los ya existentes, actualizando el registro en el sistema. En caso de ver los datos de un equipo policial, el sistema muestra todos los datos conocidos del mismo, permitiendo imprimir o exportar a PDF dicha información. En caso de asociar un equipo policial, el sistema brinda la posibilidad de realizar una búsqueda por algunos criterios básicos y seleccionar un equipo policial del listado de coincidencias quedando el mismo asociado.</p>	
<p>Requisitos Suplementarios relevantes:</p> <ul style="list-style-type: none"> <li>• Seguridad: Cuando se modifican los datos de los equipos policiales, estos no se eliminan, se ocultan para que no sean accesibles desde el sistema.</li> <li>• Interfaz: Al ver los datos de los equipos policiales el calibre (para el caso del Arma) se debe mostrar siempre en mm porque es la unidad de medida que se usará para registrar dicho dato. Ejemplo: 9 mm.</li> </ul>	



Caso de Uso	
	Consultar Equipos Policiales.
Propósito:	Buscar y listar de manera ordenada un resumen de los datos de los equipos policiales (Armas, Chalecos, Esposas y Otros Equipos) coincidentes con uno o varios criterios de búsqueda.
Actores: Consultor de Equipos Policiales.	
Resumen: El caso de uso inicia cuando el Consultor de Equipos Policiales accede a la opción de consultar equipos policiales. El sistema muestra diferentes criterios para la búsqueda de Armas, Chalecos, Esposas y Otros Equipos; el Consultor de Equipos Policiales introduce los criterios de su interés y el sistema devuelve las coincidencias encontradas. El caso de uso termina.	
Requisitos Suplementarios relevantes:	
<input type="checkbox"/> Interfaz: Los seriales del Arma se mostrarán en el listado de coincidencias concatenados, separados por “,” con el siguiente formato: O-HE34RF; donde O es el tipo de serial y HE34RF es el número del serial.	

## 1.5. METODOLOGÍA, LENGUAJES Y HERRAMIENTAS DE DESARROLLO

Para desarrollar el sistema que sustituirá al SIIPOL se utilizará la metodología RUP, lenguajes tanto para el modelado como para la codificación, herramientas de desarrollo y tecnologías de punta.

A continuación se caracterizarán detalladamente cada uno de los elementos antes mencionados.

### 1.5.1. METODOLOGÍA DE DESARROLLO RUP

“Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. Sin embargo, el Proceso Unificado es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto”<sup>2</sup>.

RUP (Rational Unified Process) es un proceso de desarrollo de software que junto al Lenguaje Unificado de Modelado (UML), constituye una metodología estándar muy utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos y es además, un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software.

- Se utiliza en proyectos que se desarrollan a largo plazo.
- Permite una mejor comunicación entre los integrantes del proyecto.
- Genera un volumen considerable de documentación, posibilitando que los cambios realizados en los miembros del equipo no resulte un problema para el avance del proyecto.
- Propone el desarrollo en ciclos e iteraciones con los artefactos que se generan, siendo esto un elemento importante para alcanzar una categoría de certificación en el desarrollo del software.

---

<sup>2</sup> Rumbaugh, James, Jacobson, Ivar and Booch, Grady. 2000. El Lenguaje Unificado de Modelado. [Online] 2000. [Cited: Diciembre 08, 2008.] <http://bibliodoc.uci.cu/pdf/reg00060.pdf>.

- Asegura la producción de software de calidad dentro de plazos y presupuestos predecibles.

### 1.5.2. LENGUAJE DE MODELADO UML

UML (Unified Modeling Lenguaje o Lenguaje de Modelación Unificado) es un lenguaje gráfico para especificar, construir, visualizar y documentar todas las disciplinas de un proyecto informático: desde el análisis con los casos de uso, hasta la implementación y configuración mediante los diagramas de despliegue.

UML permite:

- Especificar cuáles son las características de un sistema antes de su construcción.
- Construir sistemas diseñados a partir de modelos especificados.
- Visualizar gráficamente un sistema de manera que otros puedan entenderlo.
- Documentar los elementos gráficos del sistema desarrollado para futuras revisiones.
- Verificar y validar el modelo realizado.
- Generar código a partir de los modelos y viceversa.

### 1.5.3. VISUAL PARADIGM

Visual Paradigm para UML (VP-UML) es una herramienta CASE multiplataforma (Windows/Linux/Mac OS X) que soporta el ciclo de vida completo del desarrollo de software. Está diseñado para un amplio rango de usuarios, incluyendo ingenieros de software, analistas de sistema, analistas de negocio, arquitectos de sistema y quienes estén interesados en la construcción de sistemas de software confiables mediante el uso de la Orientación a Objetos. Facilita una rápida construcción de aplicaciones de calidad y a un menor coste. Visual Paradigm para UML soporta un conjunto de lenguajes (Java, C + +, PHP, Ada y Python), tanto en generación de código como ingeniería inversa. Entre sus principales características se pueden señalar:

- ❑ Facilita la comunicación de todo el equipo de desarrollo mediante el uso de un lenguaje estándar común.
- ❑ Posibilita el desarrollo de la ingeniería directa e inversa.
- ❑ Durante todo el ciclo de desarrollo el modelo y el código permanecen sincronizados, permitiendo la generación de código a partir de diagramas y viceversa.
- ❑ Permite la generación de bases de datos a partir de la transformación de diagramas de Entidad-Relación en tablas de base de datos y viceversa.

#### 1.5.4. JAVA

Es un lenguaje de programación de alto nivel, de propósito general, basado en clases y orientado a objetos; desarrollado por Sun Microsystems a mediados de los años 90. Java es el resultado de las características esenciales de otros lenguajes como C y C++.

Las características principales que ofrece son:

- ❑ Lenguaje Simple: Elimina las características menos usadas y más confusas de otros lenguajes como C++. Posee un reciclador de memoria dinámica (garbage collector) que se ocupa de liberar grandes bloques de memoria.
- ❑ Indiferente a la arquitectura: Soporta aplicaciones que serán ejecutadas en los más variados entornos de red. Para ello, el compilador de Java genera bytecodes: un formato intermedio indiferente a la arquitectura, diseñado para transportar el código eficientemente a múltiples plataformas de hardware y software.
- ❑ Distribuido: Proporciona una colección de clases para su uso en aplicaciones de red, permitiendo a los programadores acceder a la información con mucha facilidad; lo que contribuye a la creación de aplicaciones distribuidas.
- ❑ Robusto: Realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. Maneja la memoria para eliminar las preocupaciones por parte del

programador de la liberación o corrupción de memoria, reduciendo considerablemente el tiempo empleado en el desarrollo de las aplicaciones.

- ❑ Multihilo: Permite muchas actividades simultáneas en un programa, de esta forma se mejora el rendimiento interactivo y el comportamiento en tiempo real.
- ❑ Seguro: Limita cualquier aplicación del tipo Caballo de Troya a través del Cargador de Clases, separando el espacio de nombres de los ficheros locales del sistema y el de los recursos procedentes de la red. Además, elimina los punteros para prevenir el acceso ilegal a la memoria.

## 1.6. ENTORNO DE DESARROLLO

### Eclipse

Eclipse es una plataforma universal para integrar herramientas de desarrollo, posee una arquitectura abierta y basada en plug-ins. La principal característica de Eclipse es la extensibilidad, porque permite integrar diversos lenguajes sobre un mismo IDE e introducir otras aplicaciones accesorias que resultan útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces y ayuda en línea para librerías.

De manera general, presenta las siguientes características:

- ❑ Es multiplataforma (GNU/Linux, Solaris, Mac OSX).
- ❑ Es soportado para distintas arquitecturas (x86, 64 bits,...).
- ❑ Posee una estructura de plug-in que hace sencillo añadir nuevas características y funcionalidades.
- ❑ Dispone de un Control de versiones integrado a Subversión.
- ❑ Incluye muchas utilidades de edición que ayudan al programador a desarrollar el producto con rapidez, algunas son: resaltado de sintaxis, autocompletado de código, tabulador de un bloque de código seleccionado y formateado automático de código, tributando a la obtención de código de mayor calidad.

- ❑ Posee asistentes para la creación, exportación e importación de proyectos y para generar esqueletos de códigos.

## 1.7. FRAMEWORKS

### 1.7.1. ACEGI SECURITY

*Acegi Security* es un framework de código abierto, sin coste de licencias y con la seguridad añadida que proporciona el respaldo de un enorme grupo de usuarios que lo utilizan. La arquitectura de Acegi está fuertemente basada en interfaces y en patrones de diseño, proporcionando las implementaciones más utilizadas y numerosos puntos de extensión donde nuevas funcionalidades pueden ser añadidas. También provee la funcionalidad necesaria para adoptar mecanismos de seguridad en aplicaciones Java utilizando características de programación orientada a aspectos, de forma transparente para el desarrollador y sin necesidad de desarrollar código.

Acegi proporciona cuatro opciones principales de seguridad:

- ❑ Listas de control de acceso web basadas en esquemas URL.
- ❑ Protección de métodos y clases Java usando la Programación Orientada a Aspectos.
- ❑ Un procedimiento de autenticación denominado Single sign-on (SSO) que habilita al usuario para acceder a varios sistemas o recursos con una sola instancia de identificación.
- ❑ Seguridad proporcionada por el contenedor Web.

### 1.7.2. CAPA DE PRESENTACIÓN

#### Java Server Faces (JSF):

Es una tecnología para aplicaciones Web basadas en Java, que simplifica notablemente la construcción de interfaces de usuario. Posee una arquitectura fácil de usar, que define claramente una separación entre la lógica de la aplicación y la presentación, permitiendo a cada miembro de un equipo enfocarse en una parte del proceso de desarrollo; de esta manera brinda consistencia al código y seguridad a la aplicación.

Los principales componentes de JSF son:

- ❑ Un API y una implementación de referencia para: representar los componentes de Interfaz de Usuario (UI) y administrar su estado, manejar eventos, introducir validaciones y definir navegación entre las páginas.
- ❑ Una librería de etiquetas Java Server Pages (JSP) personalizadas.

JSF permite la integración con librerías como: Ajax4JSF que posibilita extender la funcionalidad de sus etiquetas dotándolas con tecnología AJAX de forma limpia y sin añadir código JavaScript. El empleo de Ajax4JSF conjuntamente al framework posibilita: cargar determinados componentes de la página sin necesidad de recargarla por completo, realizar peticiones automáticas al servidor, controlar los eventos de usuario y variar el ciclo de vida de una petición JSF.

## 1.8. ARQUITECTURA TÉCNICA

Se definió una arquitectura de **n-capas de tres niveles**, ofreciendo (ayudado por los frameworks utilizados) gran simplicidad y modularidad en la aplicación. Para el sub-módulo: Dotación de Equipos Policiales se definió un paquete común donde se encuentran clases que ofrecen servicios habituales a cada uno de los subsistemas que forman parte de este. La mayoría de estos servicios se encuentran encapsulados en las clases **ConsultarManejado** y **GestionarManejado**, para dar soporte a las funcionalidades de consultar y gestionar respectivamente.

El patrón arquitectónico **MVC** se encuentra también dentro de la arquitectura de la aplicación y está de forma implícita en el framework de presentación Java Sever Face (JSF). Es un patrón que permite que los cambios en la vista no afecten las restantes capas que componen el sistema; además, muestra varias vistas de un mismo modelo. Los participantes del framework para este patrón son las páginas .jsp como vistas, los beans de respaldo como controladores y las entidades del dominio como modelo.

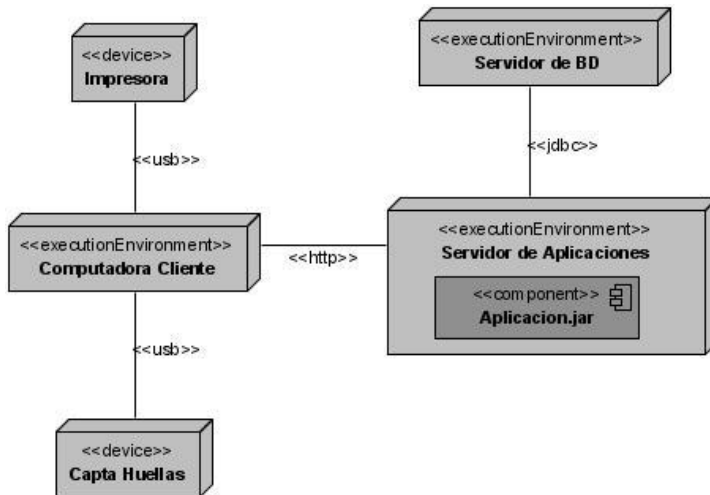


Fig. 4 Diagrama de Despliegue

## 1.9. CONCLUSIONES

En este capítulo se realizó una descripción panorámica general de algunos sistemas que actualmente se utilizan en el mundo para el control de la información relacionada con las instituciones policiales; con el estudio efectuado se determinó que las aplicaciones analizadas reúnen características muy novedosas, sin embargo no cubren los alcances del nuevo sistema diseñado para El Cuerpo de Investigaciones Científicas, Penales y Criminalísticas de la República Bolivariana de Venezuela. También ha quedado reflejado el estudio realizado a la metodología de desarrollo RUP y el análisis efectuado al conjunto de herramientas y tecnologías que serán utilizadas en el desarrollo del sistema. Como resultado se determinó el uso de: Eclipse como entorno de desarrollo, Java como lenguaje de programación y los frameworks que se utilizarán son Acegi para la seguridad y Java Server Faces para crear la Interfaz de usuario.



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

### **2.1. INTRODUCCIÓN**

En el desarrollo de este capítulo se reflejan las clases de análisis y diseño de los casos de uso, donde se especifican las funcionalidades del sistema y las necesidades de cada una para obtener el resultado esperado. Además, se detalla la secuencia de acciones entre las clases, paquetes e interfaces representadas. También se muestran los modelos de datos e implementación con sus respectivos diagramas de clases persistentes, de componentes y estándares de codificación a emplear en la solución de la propuesta.

A continuación se mostrarán los diagramas de clases del análisis de los principales casos de uso descritos en el capítulo anterior.

### **2.2. MODELO DE ANÁLISIS**

El desarrollo de sistemas y aplicaciones figuran una secuencia de acciones que comienzan con la identificación de los procesos del negocio, sigue hacia una descripción de los objetivos, define las funcionalidades y finalmente se obtienen los requisitos que conducen hacia un modelo de análisis. El propósito del análisis es transformar los requerimientos del sistema en un formato que acote perfectamente en el área concerniente a los diseñadores de software. El análisis fija su atención en garantizar que los requerimientos funcionales sean manejados; por pura simplicidad ignora muchos de los requisitos no funcionales del sistema además, de las restricciones del ambiente de implementación y como resultado expresa una idea vaga de lo que será el sistema.

A continuación se mostrarán los diagramas de clases del análisis de los principales casos de uso descritos anteriormente, principalmente aquellos relacionados con los equipos policiales, las armas incriminadas y las solicitudes de equipos policiales.



## Caso de Uso: Gestionar Ingreso de Arma Incriminada

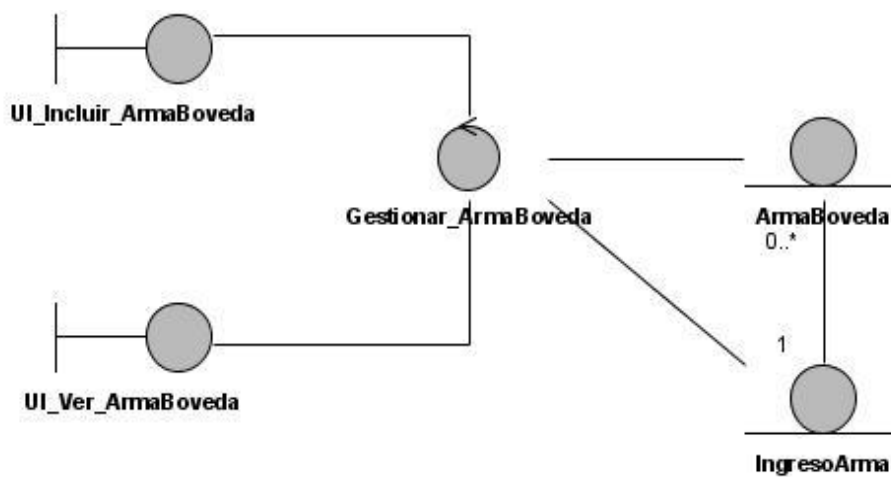


Fig. 5 Modelo de Análisis - Gestionar Ingreso de Armas Incriminadas



## Caso de Uso: Consultar Equipos Policiales

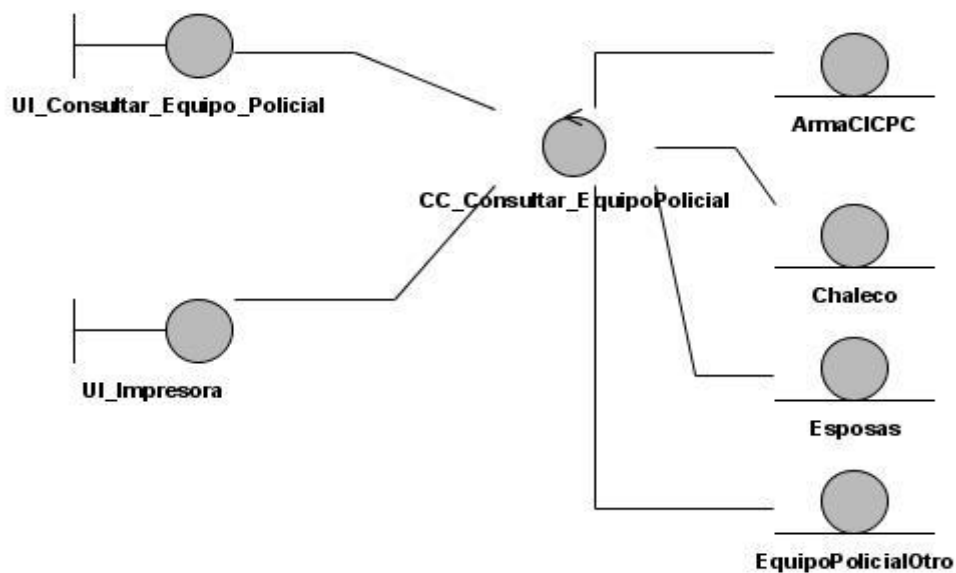
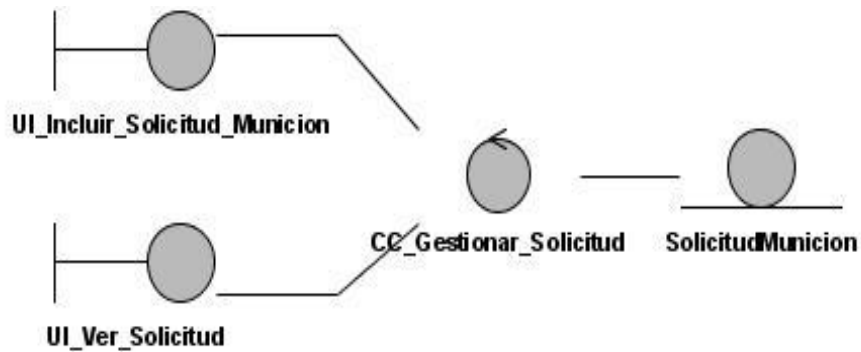
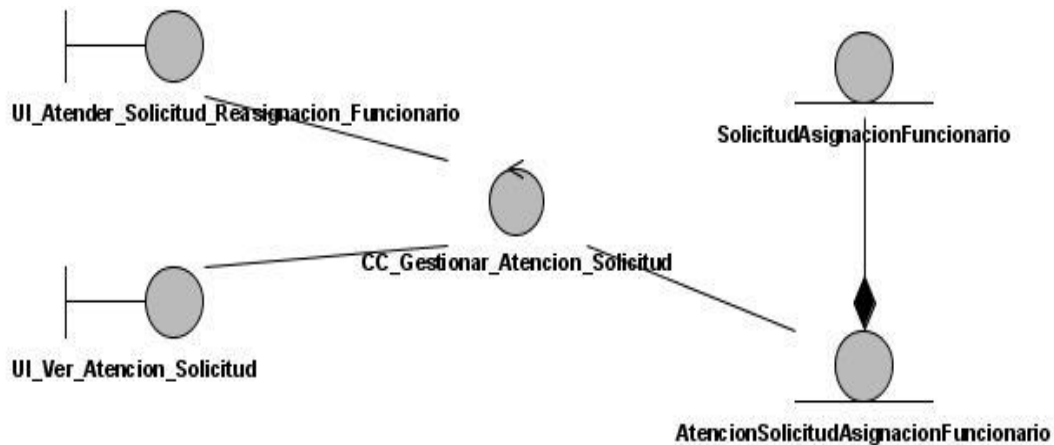


Fig. 6 Modelo de Análisis - Consultar Equipos Policiales

**Caso de Uso: Incluir Solicitud de Munición****Fig. 7 Modelo de Análisis - Incluir Solicitud de Munición**

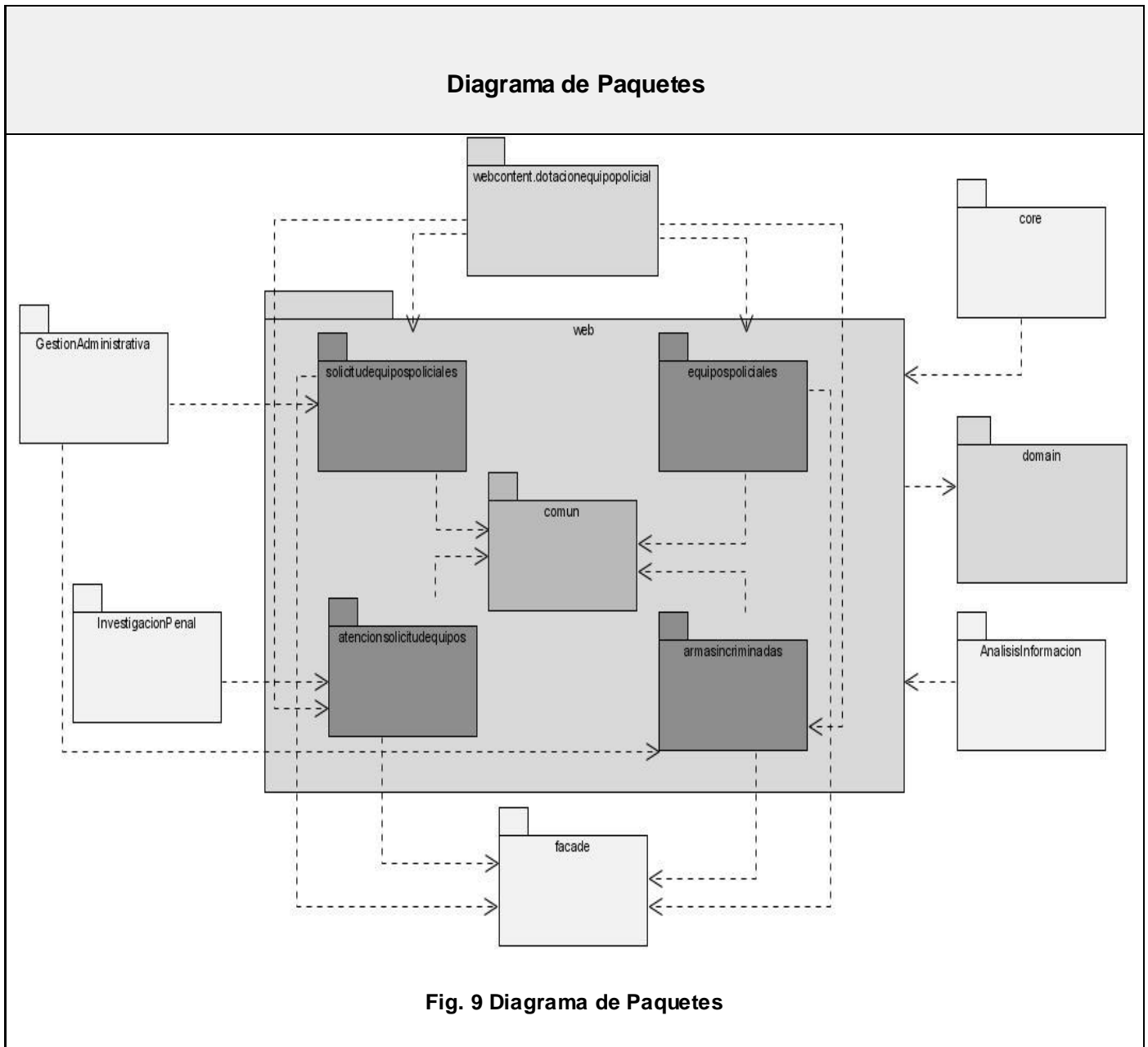
**Caso de Uso: Gestionar Dotación de Equipos Policiales a Funcionario****Fig. 8 Modelo de Análisis - Gestionar Dotación de Equipos Policiales a Funcionario**

o

**2.3. MODELO DE DISEÑO**

El modelo de diseño es el refinamiento del modelo de análisis y se centra en optimizar la concepción del diseño del sistema cuando asegura la total cobertura de los requerimientos. Transforma el modelo de la información en las estructuras de datos necesarias para implementar el sistema y su propósito es adaptar los resultados del análisis a las restricciones impuestas por los requisitos no funcionales, el entorno de implementación, entre otros elementos.

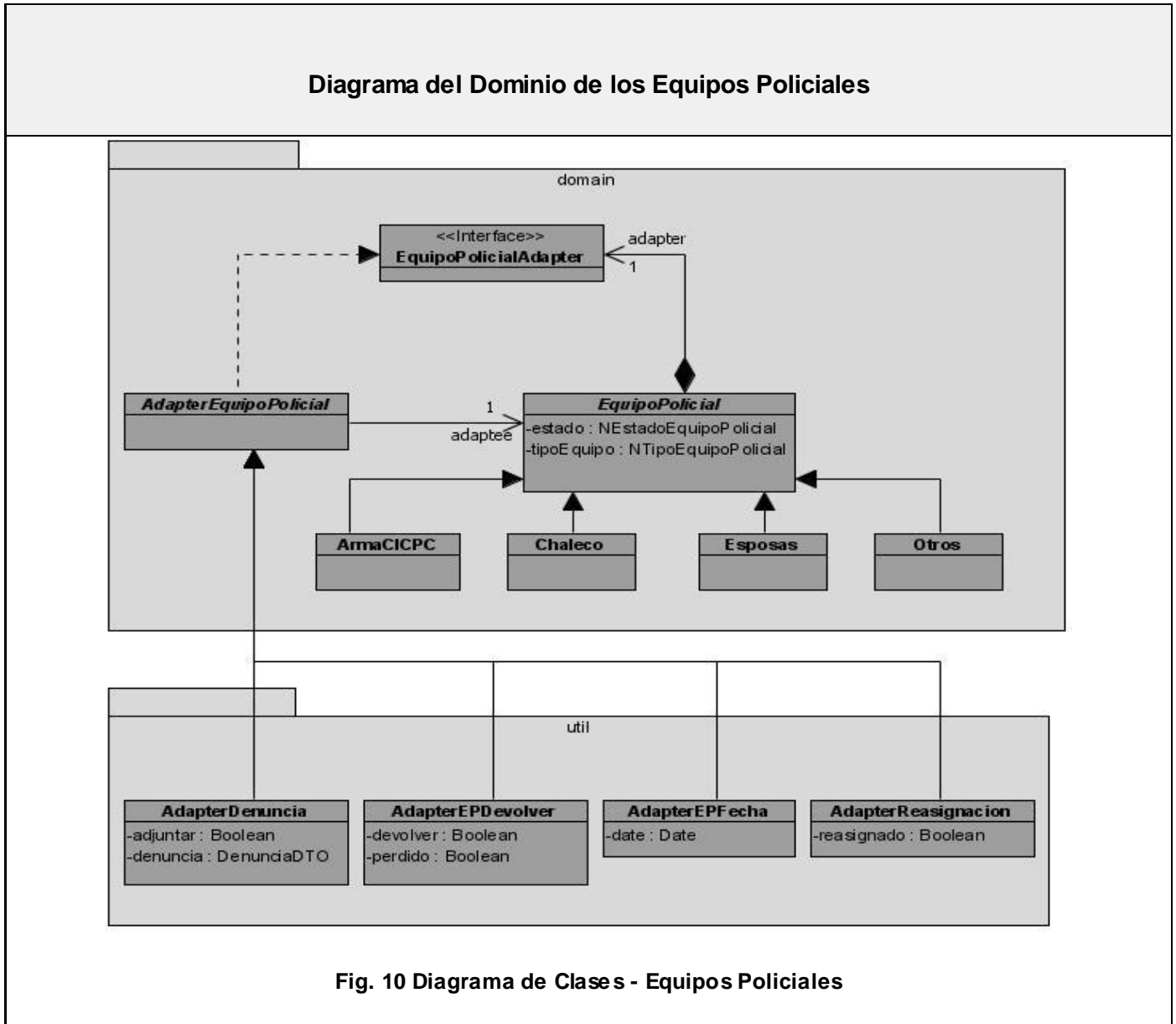
2.3.1. DIAGRAMA DE PAQUETES



En la Fig. 9 Diagrama de Paquetes, se muestra el sub-módulo Dotación de Equipos Policiales de la capa de presentación, conjuntamente a las interacciones que tiene este con el resto del sistema. La relación con el resto de la aplicación es a través de las interfaces brindadas por la capa media o de negocio; esta capa está implementada según el patrón estructural “Facade”, el cual proporciona una interfaz unificada de alto nivel que, representando todo un subsistema, facilita de esta forma su uso; permitiendo que los beans de respaldo localizados en la capa de presentación sólo conozcan el qué y no el cómo están implementados los servicios en los diferentes subsistemas. Esto promueve un bajo acoplamiento entre los subsistemas y los beans de respaldo, eliminándose o reduciéndose las dependencias; proporcionando portabilidad a los subsistemas.

Si los beans de respaldo sólo conocen la interfaz brindada por los distintos subsistemas, entonces: ¿Cómo saben estos que implementación exponer? Existen dos formas para darle a la interfaz la implementación correspondiente: Mediante una de las funcionalidades principales del framework usado en la capa media, Spring, que son las inyecciones de dependencias, traducido a que, el propio contenedor es el encargado de “inyectar” a cada interfaz la implementación correspondiente. Y la segunda es mediante el patrón j2ee “ServiceLocator”; el cual se abstrae del uso del JNDI (Java Naming and Directory Interface), consistiendo esta vía en que diferentes beans de respaldo acceden al objeto encargado de localizar los servicios, reduciendo la complejidad del código, proveyendo un sólo punto de control y mejorando el rendimiento.

2.3.2. DIAGRAMA DE CLASES DEL DISEÑO





En la Fig. 10 Diagrama de Clases - Equipos Policiales, se refleja el dominio de los equipos policiales, donde se ha utilizado el patrón de diseño estructural “**Adapter**”<sup>3</sup>

Nombre: **Adapter**.

Motivación: Incompatibilidad de la interfaz de los equipos policiales con lo que se necesitaba en la capa de presentación, así como poder añadir funcionalidades que la interfaz de los equipos policiales no proporciona.

Intención: Convertir la interfaz de la clase **EquipoPolicial** para que se adapte a lo que necesita la interfaz de usuario.

Aplicación: Al diseñar las clases **Esposas**, **Chaleco**, **ArmaCICPC**, **EquipoPolicialOtro**, cada una tiene incompatibilidad en los tipos de datos de los modelos y las marcas que exponen, en este caso se crea un adaptador de objeto enchufado junto a una interfaz **EquipoPolicialAdapter** que oculta esta problemática a los beans de respaldo que las utilizan. Para añadir funcionalidad se crea la clase Abstracta **AdapterEquipoPolicial**, de esta forma se agregan servicios a los equipos policiales encapsulados en las clases **AdapterDenuncia**, **AdapterEPDevolver**, **AdapterEPFecha**, **AdapterReasignacion**.

Consecuencias:

- Dificulta la redefinición de la interfaz de los equipos policiales.
- Agrega flexibilidad al dominio de los equipos policiales; ya que un sólo adaptador (**AdapterEquipoPolicial**) trabaja con muchas clases a adaptar.
- Agrega extensibilidad, puesto que se pueden añadir a todas las clases adaptadas a la vez.

---

<sup>3</sup> Adaptador: Patrón de diseño documentado en el libro “Design Patterns: Elements of Reusable Object-Oriented Software”



Comentario:

- ❑ El método **getAdapter ()** localizado en la clase EquipoPolicial tiene visibilidad a nivel de paquete, un nivel que sólo java puede proporcionar; permitiendo que el resto del sistema no tenga conocimiento que los equipos policiales han sido adaptados.



Nombre: EquipoPolicialAdapter	
Tipo de clase: Interfaz	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	getMarca()
Descripción:	Marca del equipo policial en formato String.
Nombre:	getModelo()
Descripción:	Modelo del equipo policial en formato String.
Nombre:	getSerial()
Descripción:	Serial del equipo policial en formato String.

Nombre: AdapterEquipoPolicial	
Tipo de clase: clase abstracta	
Atributo	Tipo
adaptee	EquipoPolicial
Para cada responsabilidad:	
Nombre:	getMarca()
Descripción:	Implementación de la interfaz, delega la responsabilidad en el adaptee.
Nombre:	getModelo()
Descripción:	Implementación de la interfaz, delega la responsabilidad en el adaptee.



Nombre:	getSerial()
Descripción:	Implementación de la interfaz, delega la responsabilidad en el adaptee.
Nombre:	getTipoEquipo()
Descripción:	Delega la responsabilidad en el adaptee de devolver el tipo de equipo policial.
Nombre:	getAdaptee()
Descripción:	Devuelve el equipo policial adaptado

Nombre: EquipoPolicial	
Tipo de clase: clase	
Atributo	Tipo
estado	NEstadoEquipoPolicial
tipoEquipo	NTipoEquipoPolicial
adapter	EquipoPolicialAdapter
Para cada responsabilidad:	
Nombre:	getEstado()
Descripción:	Devuelve el estado del equipo policial, este puede ser: extraviado, en bóveda y asignado.
Nombre:	getTipoEquipo()
Descripción:	Devuelve el tipo de equipo policial, puede ser: arma, chaleco, esposas u otro equipo policial.
Nombre:	getAdapter()
Descripción:	Método con visibilidad de paquete para ocultar el adaptador al sistema, devuelve un adaptador para cada equipo policial.



Diagrama de los Paquetes: comun y controller

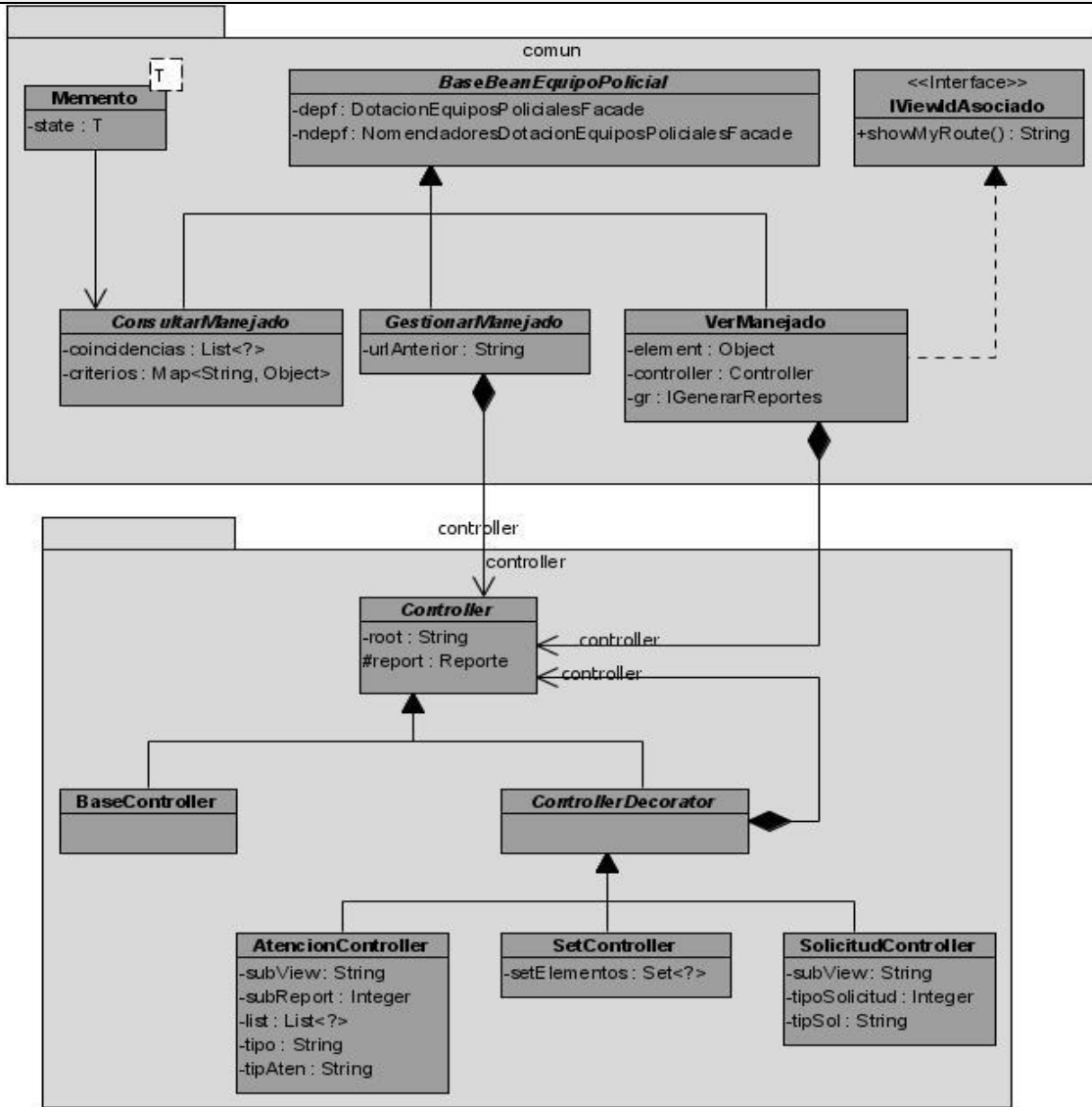


Fig. 11 Diagrama de Clases - Paquete: comun y controller



En el paquete común Fig. 11 se encuentran las clases que contienen las funcionalidades más generales y comunes. En lo más alto de la jerarquía se encuentra la clase abstracta **BaseBeanEquipoPolicial**, todas las clases pertenecientes al sub-módulo: Dotación de Equipos Policiales en la capa de presentación forman parte de su jerarquía, desde ahí se tiene acceso a la fachada del propio sub-módulo. Luego se encuentran las clases **ConsultarManejado**, **GestionarManejado** y **VerManejado**, que a continuación se explicarán detalladamente.

La clase abstracta **ConsultarManejado** posee las funcionalidades para consultar cualquier entidad persistente en el sistema; diseñada según los patrones de comportamiento “**Memento**<sup>4</sup>” y “**Template Method**<sup>5</sup>” (los cuales se detallan a continuación) e implementada utilizando los APIs de *reflection* y *anotations* de Java.

Nombre: Memento.

Motivación: Se necesitaba guardar el estado interno de un objeto para su futura utilización.

Intención: Capturar y exteriorizar el estado interno del bean de respaldo sin violar el encapsulamiento.

Participantes:

- Memento: El objeto que mantiene el estado.
- Originador: El objeto al que pertenece originalmente el estado conservado por el Memento.
- Cuidador: Objeto que maneja los Mementos en una estructura de datos.

---

<sup>4</sup> Memento: Patrón de diseño documentado en el libro “Design Patterns: Elements of Reusable Object-Oriented Software”

<sup>5</sup> Método Plantilla: Patrón de diseño documentado en el libro “Design Patterns: Elements of Reusable Object-Oriented Software”



#### Consecuencias:

- ❑ El uso de “Memento” puede ser problemático por el excesivo consumo de recursos, en caso de que la información que el bean de respaldo deba almacenar sea considerable o las peticiones sean muy frecuentes.
- ❑ Se salvaguarda el encapsulamiento, impidiendo la exposición pública de información que sólo el bean de respaldo debe manejar.

#### Comentarios:

- ❑ El bean de respaldo participa como “Originador” y “Cuidador”, dando lugar a que no esté simplificado y se haga cargo del mantenimiento de todas las versiones de su estado interno.
- ❑ Sólo el “Originador” puede gestionar la información del memento, que resulta por tanto, una caja negra para el resto del sistema.

Nombre: Template Method.

Motivación: Sacar un factor común del comportamiento compartido por varios beans de respaldo para localizarlo en una única clase (**ConsultarManejado**).

Intención: Definir el esqueleto de un algoritmo para una operación, dejando para sus subclases la capacidad de redefinir el funcionamiento de los pasos de este algoritmo.

Aplicación: Consultar entidades persistentes es la funcionalidad principal de todo bean de respaldo en la jerarquía de la clase **ConsultarManejado**. El método correspondiente a hacer las consultas: **lookForCoincidence()** con una firma preparada para devolver un listado de objetos persistentes según una búsqueda por ciertos criterios (debe ser redefinido).

Consecuencias:

- ❑ Favorece el bajo acoplamiento y la alta cohesión debido a su estructura de control invertido (IoC) basado en el “Principio de Hollywood”.

Comentario:

- ❑ Constituyendo una técnica fundamental para la reutilización de código.
- ❑ Evita la obligación de las subclasses llamadas a la superclase, degenerando en el frecuente anti patrón “**Call Super**”<sup>6</sup>.

La clase abstracta **GestionarManejado** Fig. 11, está diseñada para incluir (utilizando el patrón “**Template Method**”) y ver (apoyándose de la clase **VerManejado** y las clases incluidas en el paquete controller) cualquier entidad persistente en el sistema. La filosofía con el patrón “**Template Method**” utilizada en la clase **GestionarManejado** es igual que la utilizada en el **ConsultarManejado** explicado anteriormente. Esta define el método abstracto **incluir()** para que las subclasses implementen el algoritmo que permite incluir satisfactoriamente un elemento y el método abstracto **getViewElement()** para comunicarle al **VerManejado** a través de la clase **Controller** el objeto que se desea mostrar.

La clase **VerManejado** Fig. 11 es el bean de respaldo que apoya todas las interfaces de usuario destinadas a mostrar un elemento al usuario. Este bean expone el método **showMyRoute()** como contrato con la interfaz **IViewIdAsociado**, dicho método le permite variar entre las distintas interfaces de usuario a las cuales respalda.

El paquete **Controller** Fig. 11, contiene una jerarquía de clases hecha específicamente para mediar entre el **GestionarManejado** y el **VerManejado**. Los controladores dentro de este paquete se diseñaron

---

<sup>6</sup> Llamada a la Súper Clase: Anti patrón de diseño que obliga a las clases sobrescribir los métodos de estas y llamar dentro del método sobrescrito el método de la súper clase.





siguiendo el patrón estructural “**Decorator**”<sup>7</sup>(detallado a continuación) y son los encargados de llevar el objeto que se gestiona dentro del **GestionarManejado** para mostrarlo con ayuda del **VerManejado**.

Nombre: Decorator.

Motivación: La extensión de los controladores por herencia no es factible por ser esta imprevista en tipo y número.

Intención: Añadir responsabilidades a un controlador de manera dinámica y transparente (herencia dinámica).

Aplicación: El controlador con las funcionalidades para mediar entre el **GestionarManejado** y el **VerManejado** es el **BaseController**, sin embargo, a este se le añaden funcionalidades decorándolo con el **AtencionController**, **SetController**, **SolicitudController** o una composición de estos.

Consecuencias:

- Evita que en lo alto de la jerarquía, clases “guiadas por las responsabilidades” procuren satisfacer todas las posibilidades. De esta forma las nuevas funcionalidades se componen de piezas simples que se crean y se combinan con facilidad.
- Gran fragmentación, el sistema se llena de gran cantidad de objetos pequeños, lo cual puede dificultar el aprendizaje de su funcionamiento y su depuración.

---

<sup>7</sup> Decorador: Patrón de diseño documentado en el libro “Design Patterns: Elements of Reusable Object-Oriented Software”



Nombre: Memento	
Tipo de clase: clase	
Atributo	Tipo
state	T
Para cada responsabilidad:	
Nombre:	getState()
Descripción:	Devuelve el estado guardado de un objeto.
Nombre:	setState(T state)
Descripción:	Salva el estado de un objeto.

Nombre: ConsultarManejado	
Tipo de clase: abstracta	
Atributo	Tipo
coincidencias	List<?>
critérios	Map<String, Object>
memento	Memento<ConsultarManejado>
Para cada responsabilidad:	
Nombre:	consultar(ActionEvent e)



Descripción:	Busca las coincidencias según la consulta que se realice además, de guardar el estado de sí mismo según el patrón memento.
Nombre:	lookForCoincidence()
Descripción:	Delega la responsabilidad para hacer las consultas.
Nombre:	nuevaBusqueda(ActionEvent e)
Descripción:	Delega la responsabilidad de preparar la interfaz de usuario para una nueva búsqueda.
Nombre:	cancelar(ActionEvent e)
Descripción:	Cancela la operación de hacer una consulta específica.



Nombre: GestionarManejado	
Tipo de clase: abstracta.	
Atributo	Tipo
urlAnterior	String
controller	Controller
Para cada responsabilidad:	
Nombre:	incluir()
Descripción:	Delega la responsabilidad para incluir un elemento (Basado en el patrón de diseño "Template Method").
Nombre:	finalizar(ActionEvent e)
Descripción:	Finaliza la gestión de un elemento.
Nombre:	getViewElement()
Descripción:	Elemento que será procesado por el controller y redireccionado a la interfaz de ver elementos.
Nombre:	cancelar(ActionEvent e)
Descripción:	Cancela la operación en curso.



Nombre: Controller	
Tipo de clase: abstracta	
Atributo	Tipo
root	String
report	Reporte
Para cada responsabilidad:	
Nombre:	verBean(String url, Object obj)
Descripción:	Redirecciona para la interfaz de usuario “ver elemento” usando el patrón de diseño “Decorator”.

Diagrama del Clases asociar de los Equipos Policiales

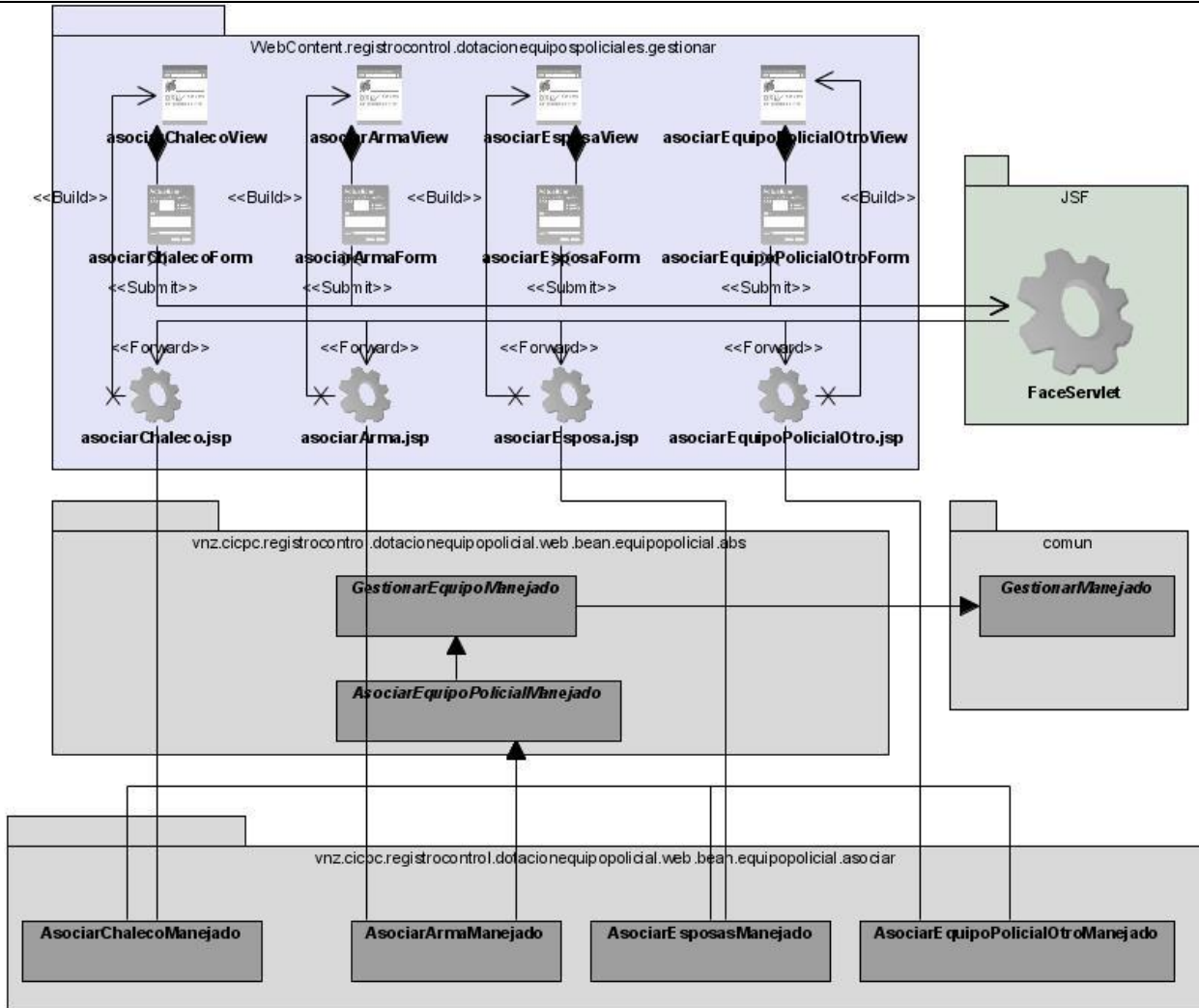


Fig. 12 Diagrama de Clases - Asociar Equipos Policiales



Nombre: AsociarEquipoPolicialManejado	
Tipo de clase: clase abstracta.	
Atributo	Tipo
tabla	UIData
Para cada responsabilidad:	
Nombre:	buscarCoincidencias()
Descripción:	Delega en las subclases la responsabilidad de buscar las coincidencias.
Nombre:	getCoincidencias()
Descripción:	Retorna las coincidencias de la búsqueda.
Nombre:	cancelarAsociacion(ActionEvent e)
Descripción:	Cancela la asociación de un equipo policial.
Nombre:	seleccionarEquipoPolicial(ActionEvent e)
Descripción:	Selecciona un equipo policial.

<i>Nombre: AsociarChalecoManejado</i>	
Tipo de clase: clase	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	buscarCoincidencias()
Descripción:	Busca las coincidencias de los chalecos.

Diagrama de Clases incluir de los Equipos Policiales

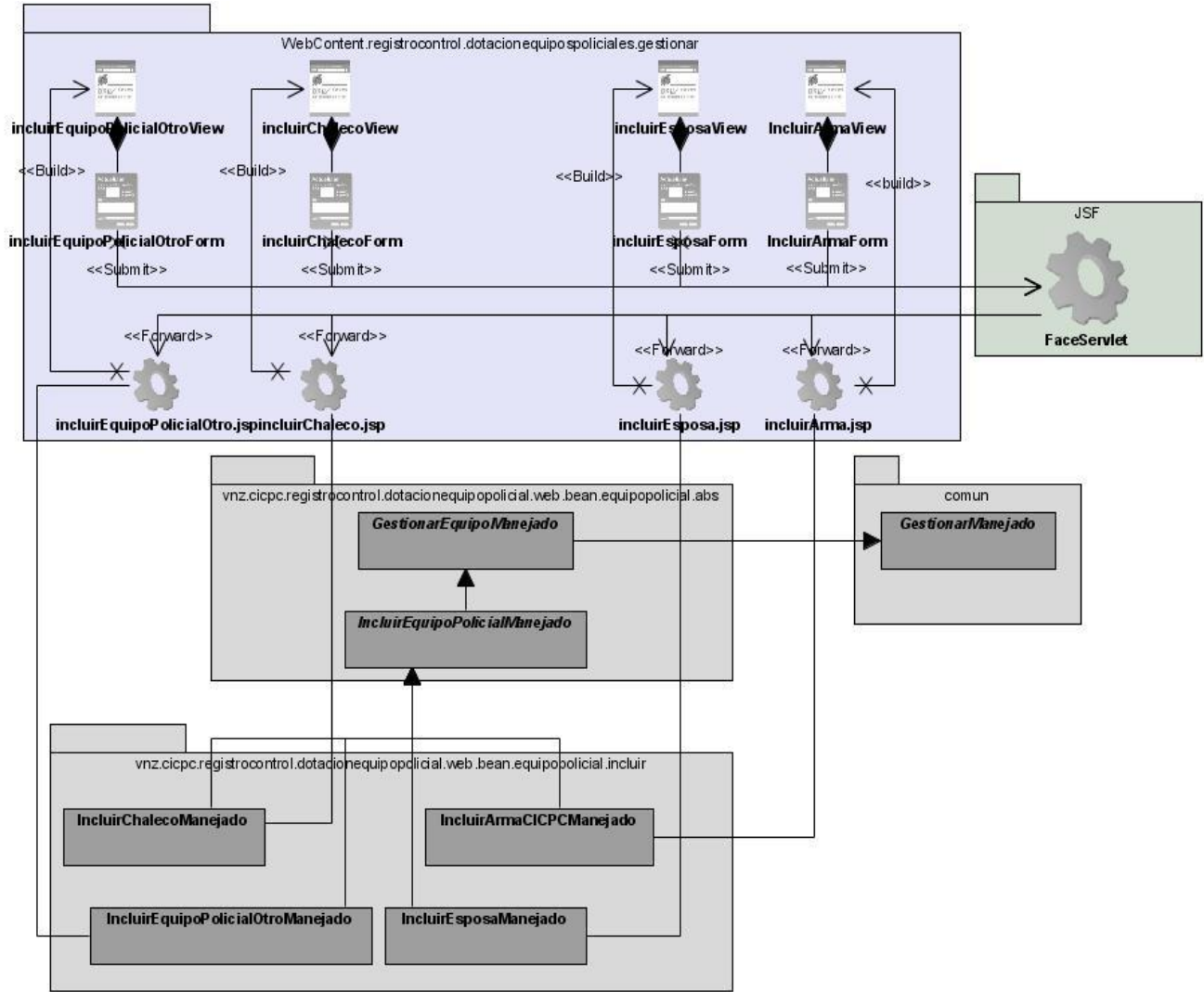
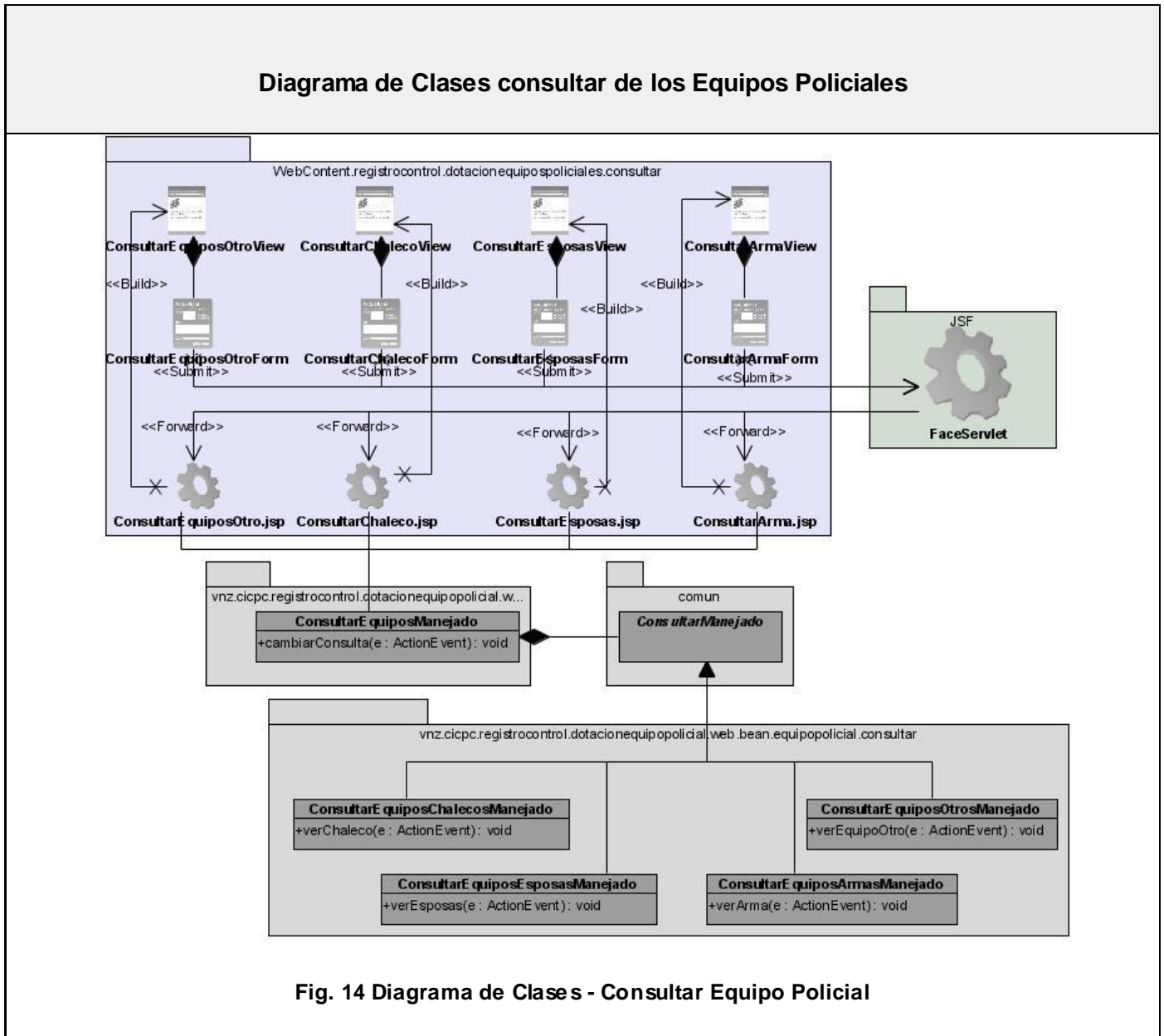


Fig. 13 Diagrama de Clase - Incluir Equipo Policial



En el paquete consultar de los equipos policiales Fig. 14 se muestra un diseño siguiendo el patrón estructural “Bridge”<sup>8</sup>.



<sup>8</sup> Puente: Patrón de diseño documentado en el libro “Design Patterns: Elements of Reusable Object-Oriented Software”



Nombre: Bridge

Motivación: Proliferación de clases, tal que, el uso de la herencia no es recomendable por su inflexibilidad. Ocultar la implementación de la abstracción del **ConsultarManejado**.

Intención: Separar la abstracción de su implementación para permitir que ambos puedan variar independientemente.

Aplicación: La clase **ConsultarEquiposPoliciales** oculta completamente la implementación de la clase **ConsultarManejado**, de esta forma las implementaciones pueden variar sin afectar la abstracción.

Consecuencias:

- La separación entre interfaz e implementación permite configurar esta relación dinámicamente, en tiempo de ejecución. Además, esta independencia favorece la estructuración del sistema.
- Se favorece el encapsulamiento al ocultarle a los clientes detalles referentes a la implementación.

Nombre: ConsultarEquipoManejado	
Tipo de clase: clase	
Atributo	Tipo
tipoEquipo	String
urlConsultar	String
consultar	ConsultarManejado
include	Include
Para cada responsabilidad:	
Nombre:	cambiarConsulta(ActionEvent e)
Descripción:	Utilizando el patrón de diseño "Bridge" posibilita cambiar en tiempo de ejecución

la implementación para consultar equipos policiales.

En la Fig. 15 se representa una de las solicitudes de equipos policiales que se pueden realizar en el sistema, estas solicitudes están diseñadas según el patrón de diseño creacional “**Factory Method**”<sup>9</sup>, el cual se detallara a continuación.

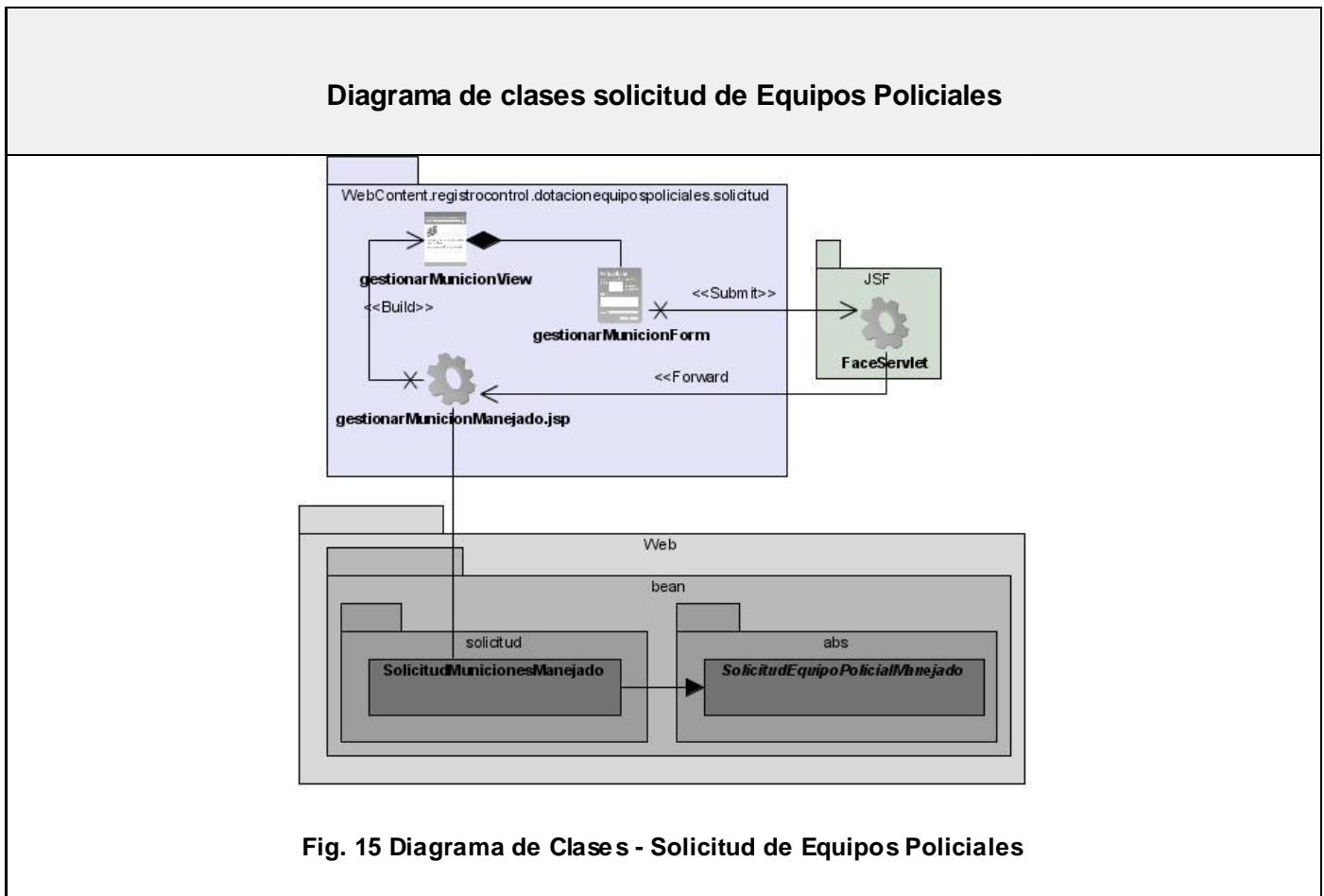


Fig. 15 Diagrama de Clases - Solicitud de Equipos Policiales

<sup>9</sup> Método Fábrica: Patrón de diseño documentado en el libro “Design Patterns: Elements of Reusable Object-Oriented Software”



Nombre: Factory Method.

Motivación: No se puede anticipar el tipo de solicitud que se va a crear.

Intención: Definir un método para la creación de una solicitud, pero permitiendo a las subclases decidir de qué clase instanciarlo. Permitir a la clase **SolicitudEquipoPolicialManejado** diferir la instanciación de las solicitudes a favor de sus subclases.

Aplicación: La clase **SolicitudEquipoPolicialManejado** maneja datos generales de las solicitudes sobre los equipos policiales, la aplicación de este patrón le permite manejar estos datos sin preocuparse por el tipo de solicitud, delegando en las subclases la creación de la solicitud con el método **newInstance()**.

Consecuencias:

- Se gana en flexibilidad; pues crear las solicitudes dentro de una clase con un Factory Method es siempre más flexible que hacerlo directamente, debido a que se elimina la necesidad de atar al sistema solicitudes concretas.
- Se obliga a definir subclases de la clase **SolicitudEquipoPolicialManejado** sólo para crear un producto concreto y esto no siempre puede ser apropiado.

Nombre: SolicitudEquipoPolicialManejado	
Tipo de clase: clase abstracta	
Atributo	Tipo
funcionarioGestor	Funcionario
credencialFuncionario	String
funcionarioReceptor	Funcionario
mostrarDatosFuncionario	boolean
solicitud	SolicitudEquiposPoliciales



numeroComunicacion	String
dependenciaReceptora	DependenciaInterna
importancia	NImportanciaComunicacion
Para cada responsabilidad:	
Nombre:	buscarFuncionarioCredencial(ActionEvent e)
Descripción:	Busca un funcionario según la credencial.
Nombre:	autenticarFuncionario(ValidatorParams param)
Descripción:	Valida la autenticidad de un funcionario.
Nombre:	newInstance()
Descripción:	Devuelve una nueva instancia de un tipo de solicitud y delega la responsabilidad de crearla según el patrón de diseño "Factory Method".
Nombre:	incluir()
Descripción:	Realiza algunas configuraciones necesarias para guardar una solicitud.

## 2.3.3. DIAGRAMA DE CONTRATO ENTRE PAQUETES

Diagrama: Contrato entre Paquetes - Equipo Policial

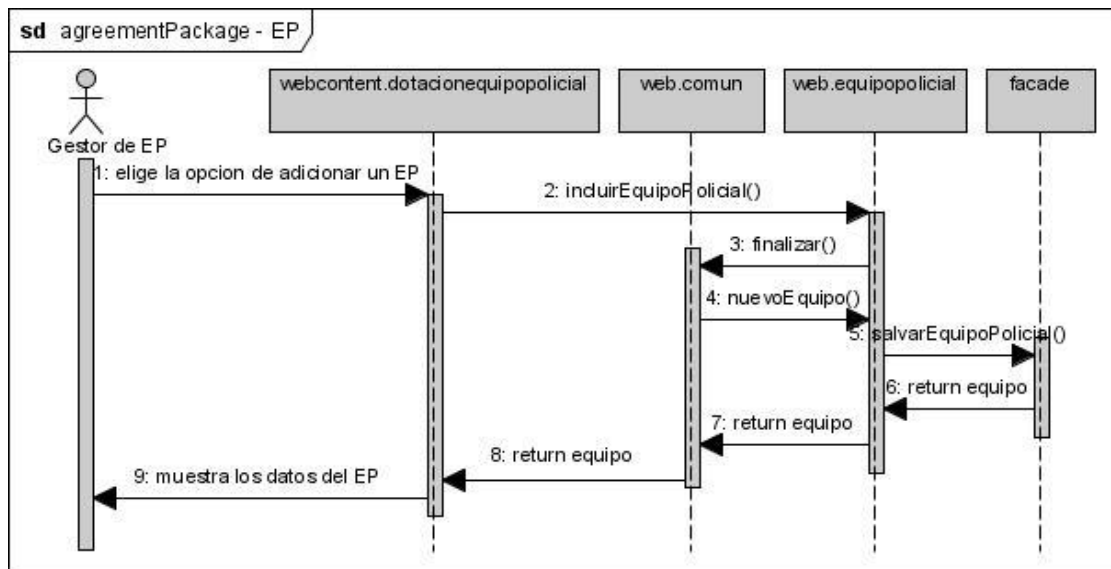


Fig. 16 Contrato entre Paquetes - Equipo Policial

## Diagrama: Contrato entre Paquetes - Armas Incriminadas.

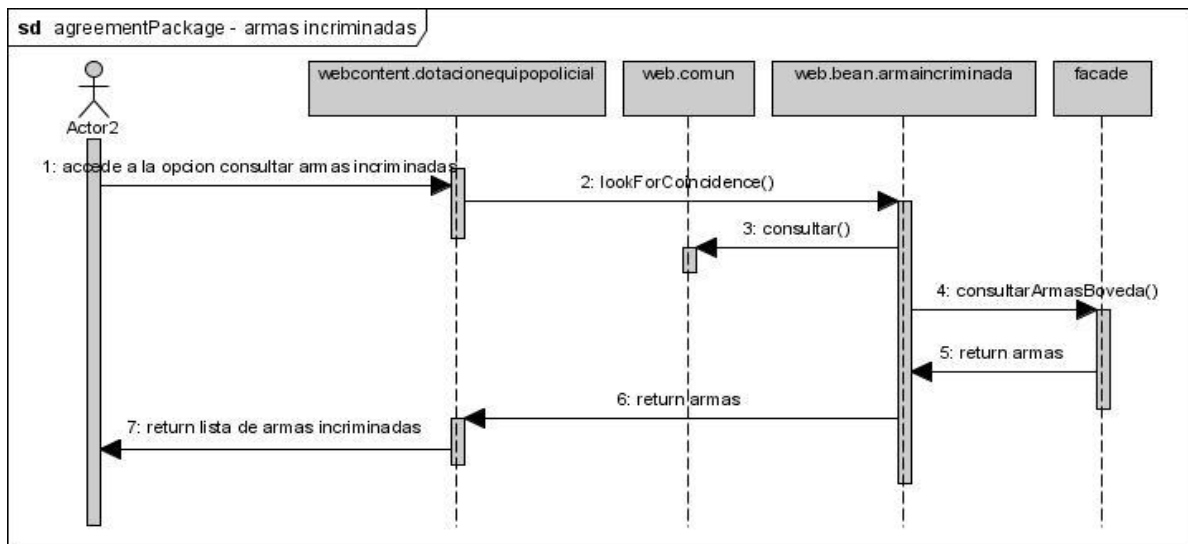
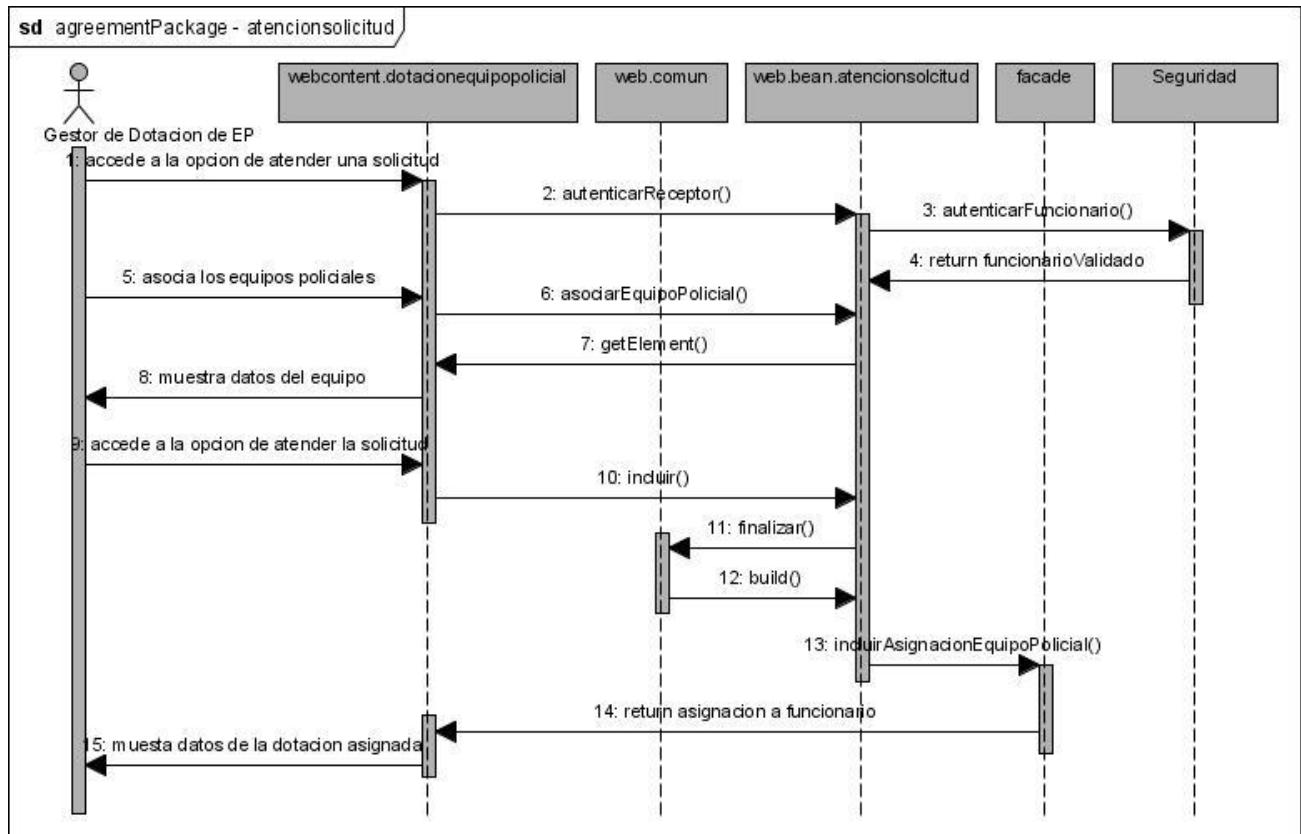


Fig. 17 Contrato entre paquetes - Armas Incriminadas



**Diagrama: Contrato entre Paquetes - Atender Solicitud de Equipos Policiales**



**Fig. 18 Contrato entre Paquetes - Atender Solicitud de Equipos Policiales**





Diagrama: Contrato entre Paquetes - Solicitudes de Equipos Policiales

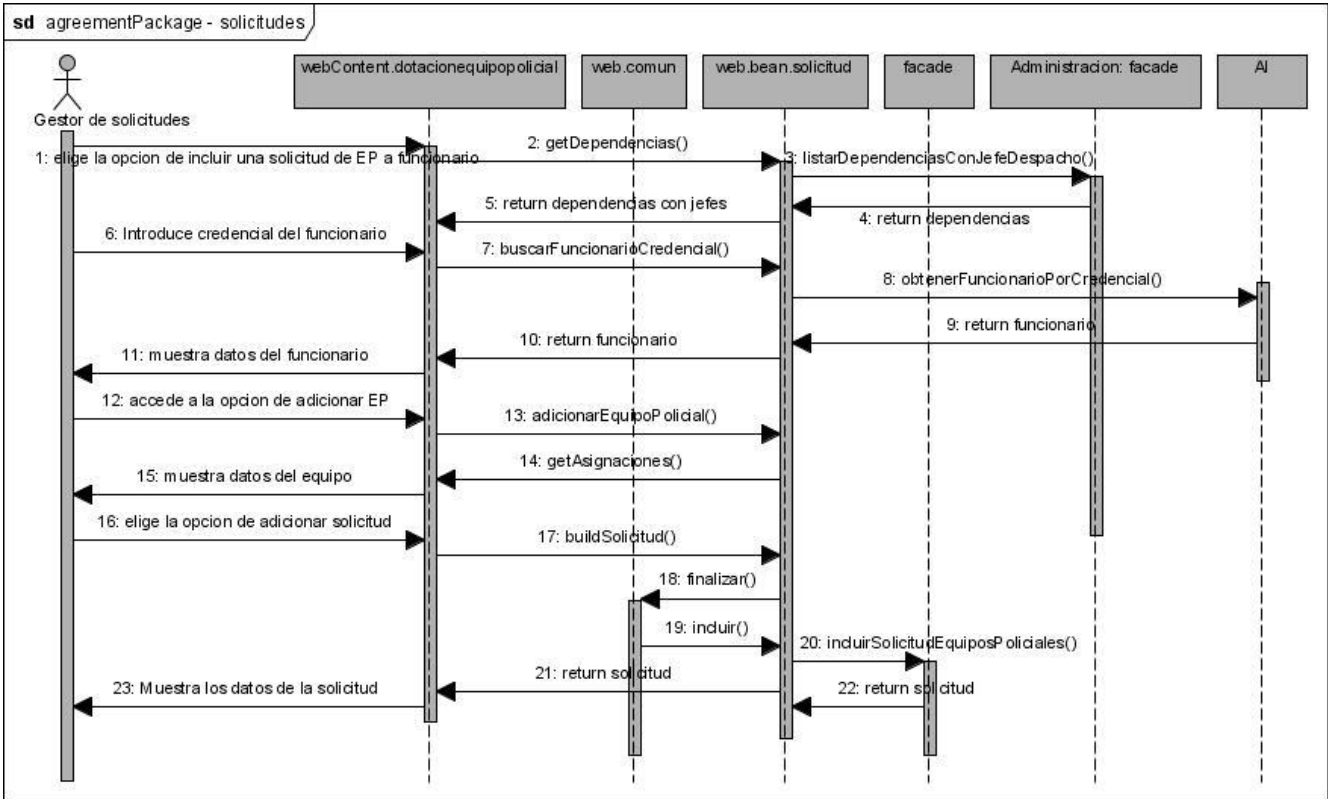


Fig. 19 Contrato entre Paquetes - Solicitudes de Equipos Policiales

## 2.4. MODELO DE DATOS

Diagrama de Clases Persistentes para las Armas Incriminadas

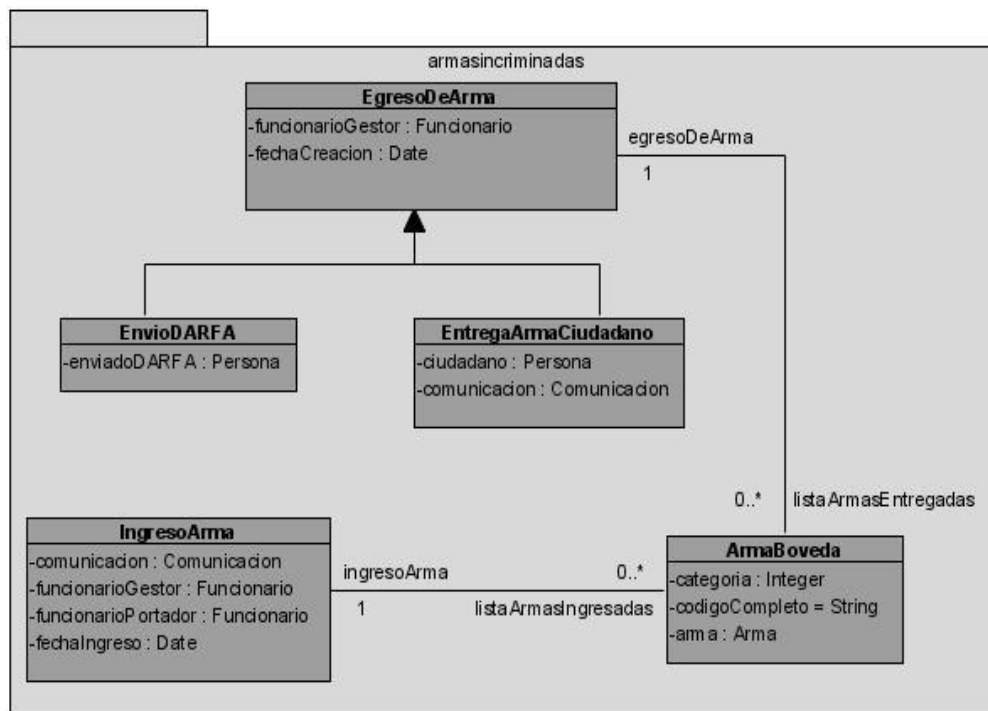


Fig. 20 Diagrama de Clases - Armas Incriminadas



Diagrama de Clases Persistentes para los Equipos Policiales

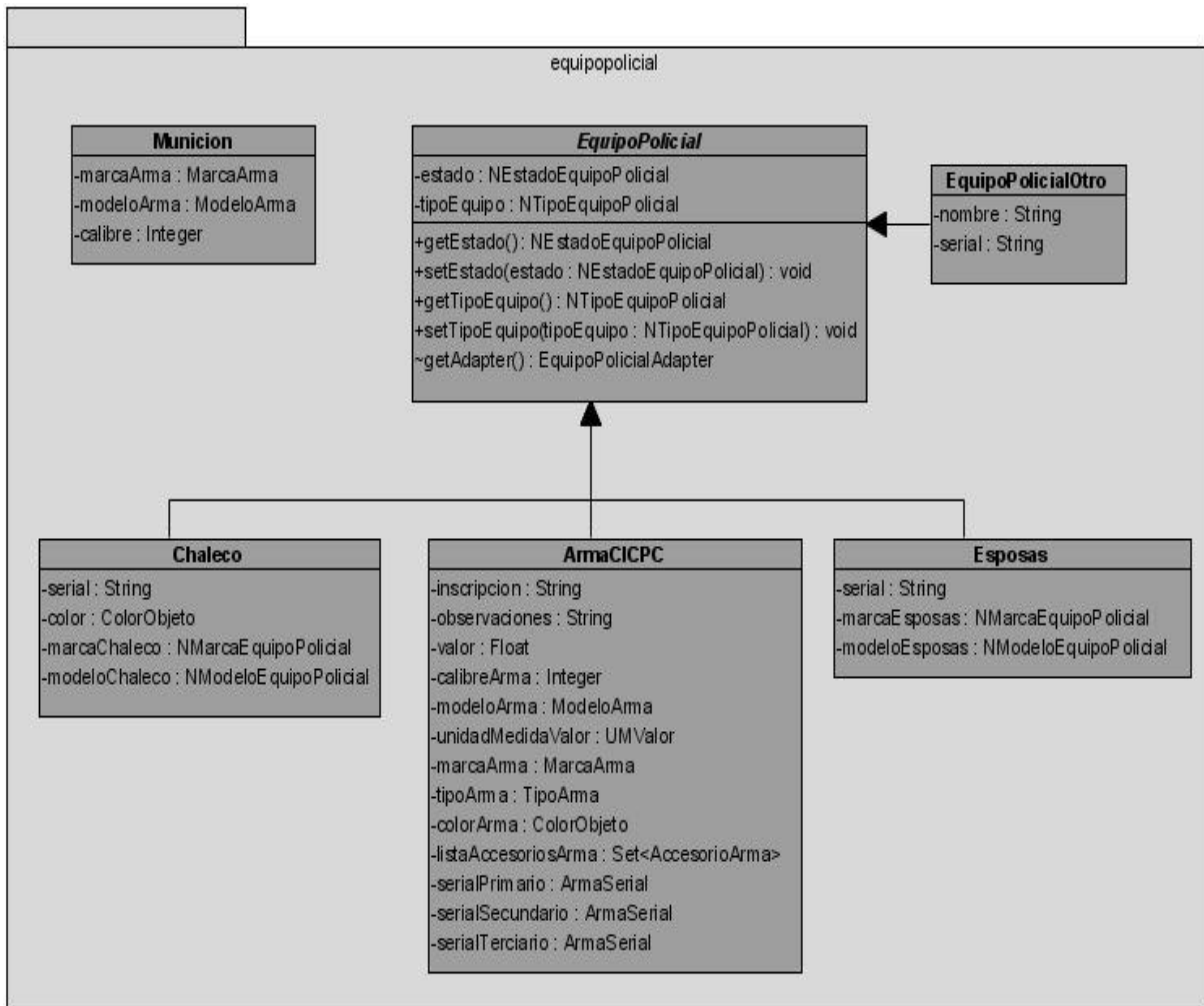


Fig. 21 Diagrama de Clases - Equipos Policiales

Diagrama de Clases Persistentes Solicitudes de Equipos Policiales

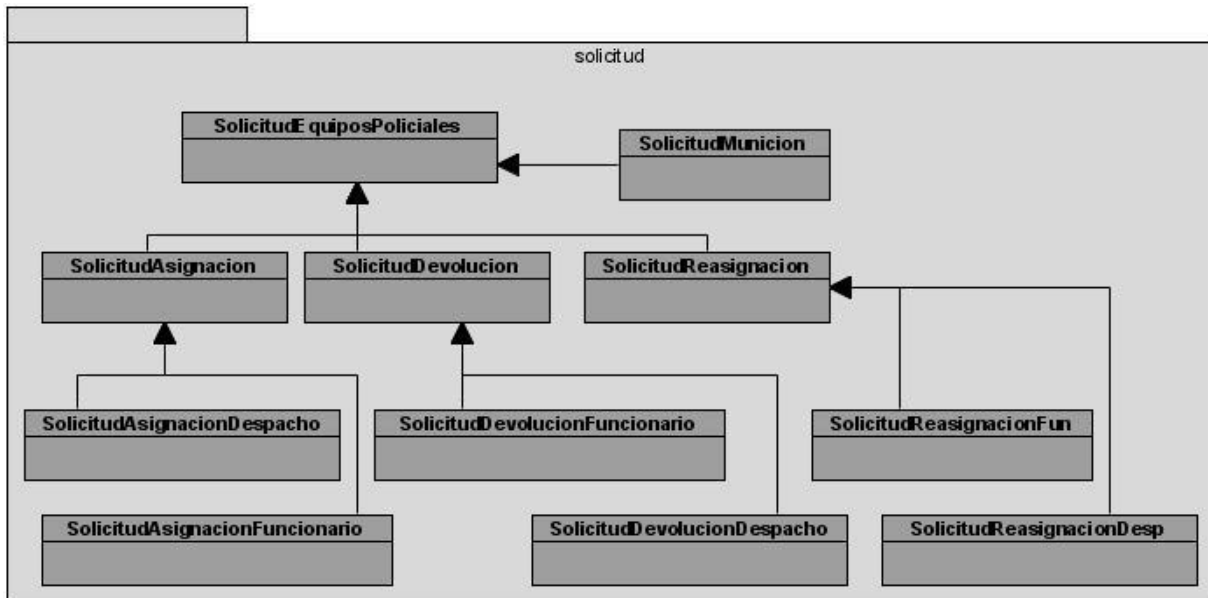


Fig. 22 Diagrama de Clases - Solicitudes de Equipos Policiales

## Diagrama de Clases Persistentes Atención de Solicitudes de Equipos Policiales

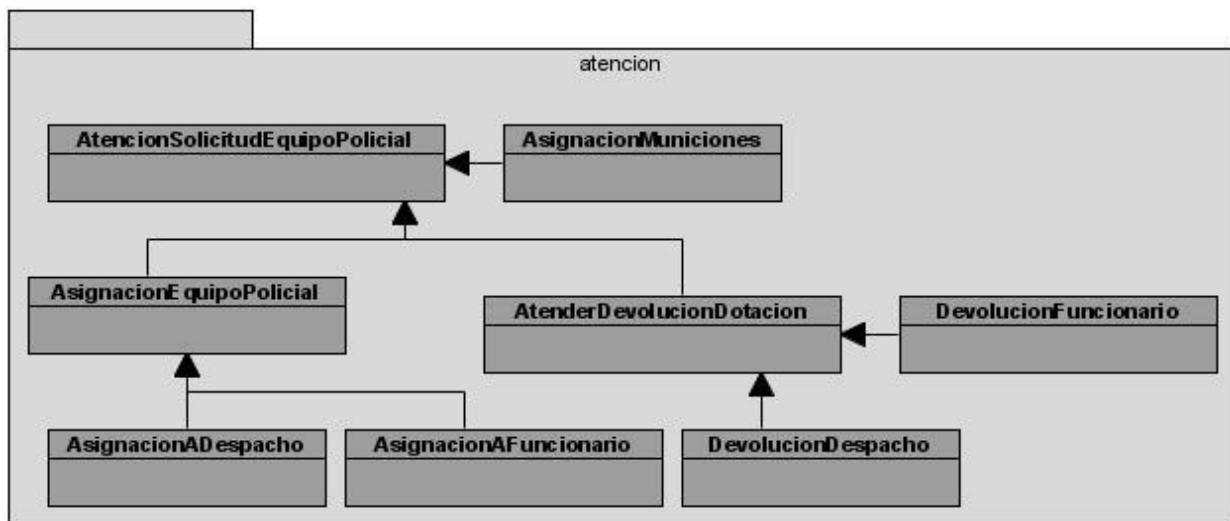
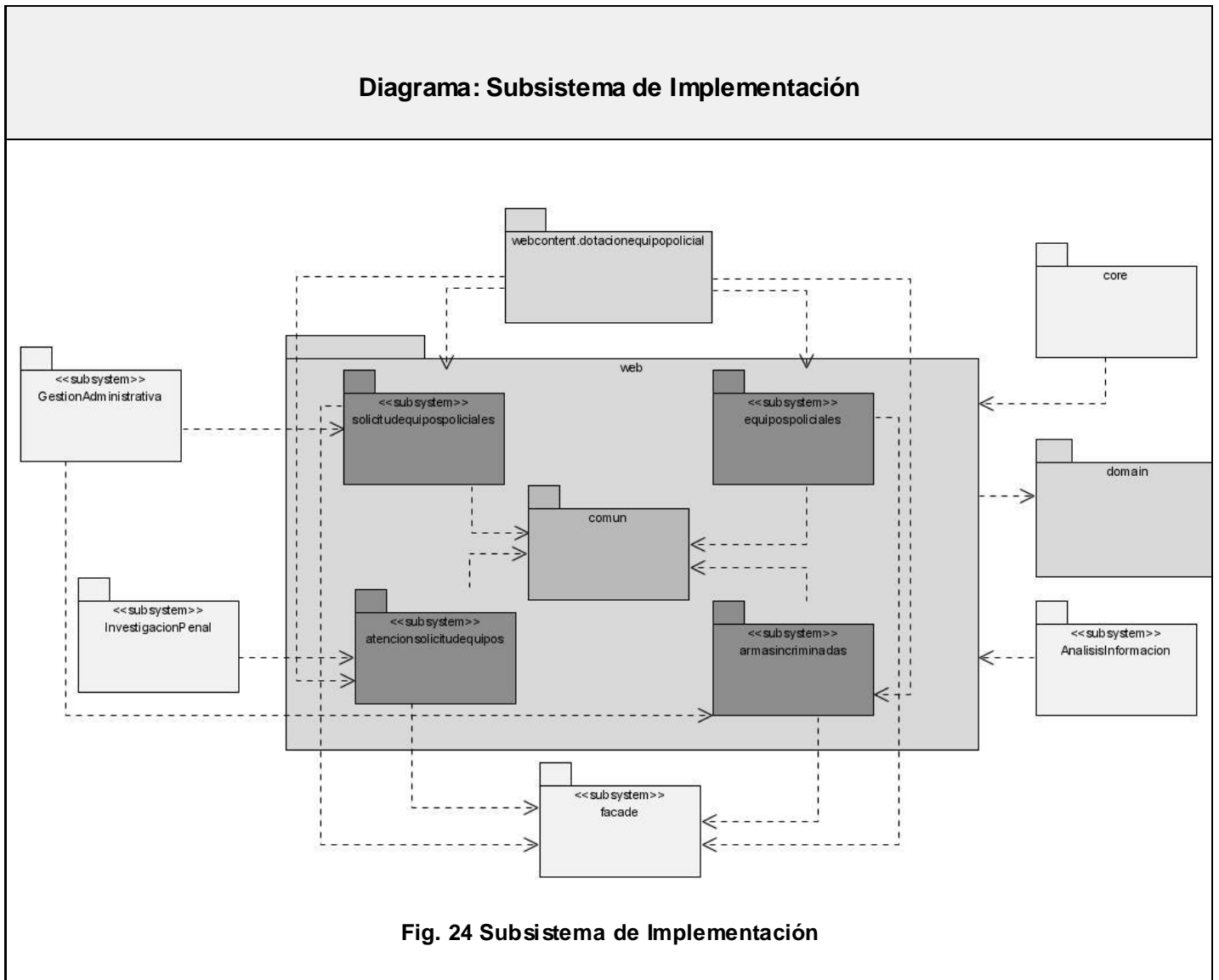


Fig. 23 Diagrama de Clases - Atención de Solicitudes de Equipos Policiales

## 2.5. MODELO DE IMPLEMENTACIÓN

A continuación se representa la modelación de los diagramas de la etapa de implementación; la cual tiene como propósito dejar listo un software en su versión operativa inicial que cumpla con los requerimientos tanto, funcionales como no funcionales y posea la calidad, robustez, portabilidad, y flexibilidad adecuada para su aplicación.

2.5.1. DIAGRAMA DE SUBSISTEMA DE IMPLEMENTACIÓN



2.5.2. DIAGRAMA DE COMPONENTES

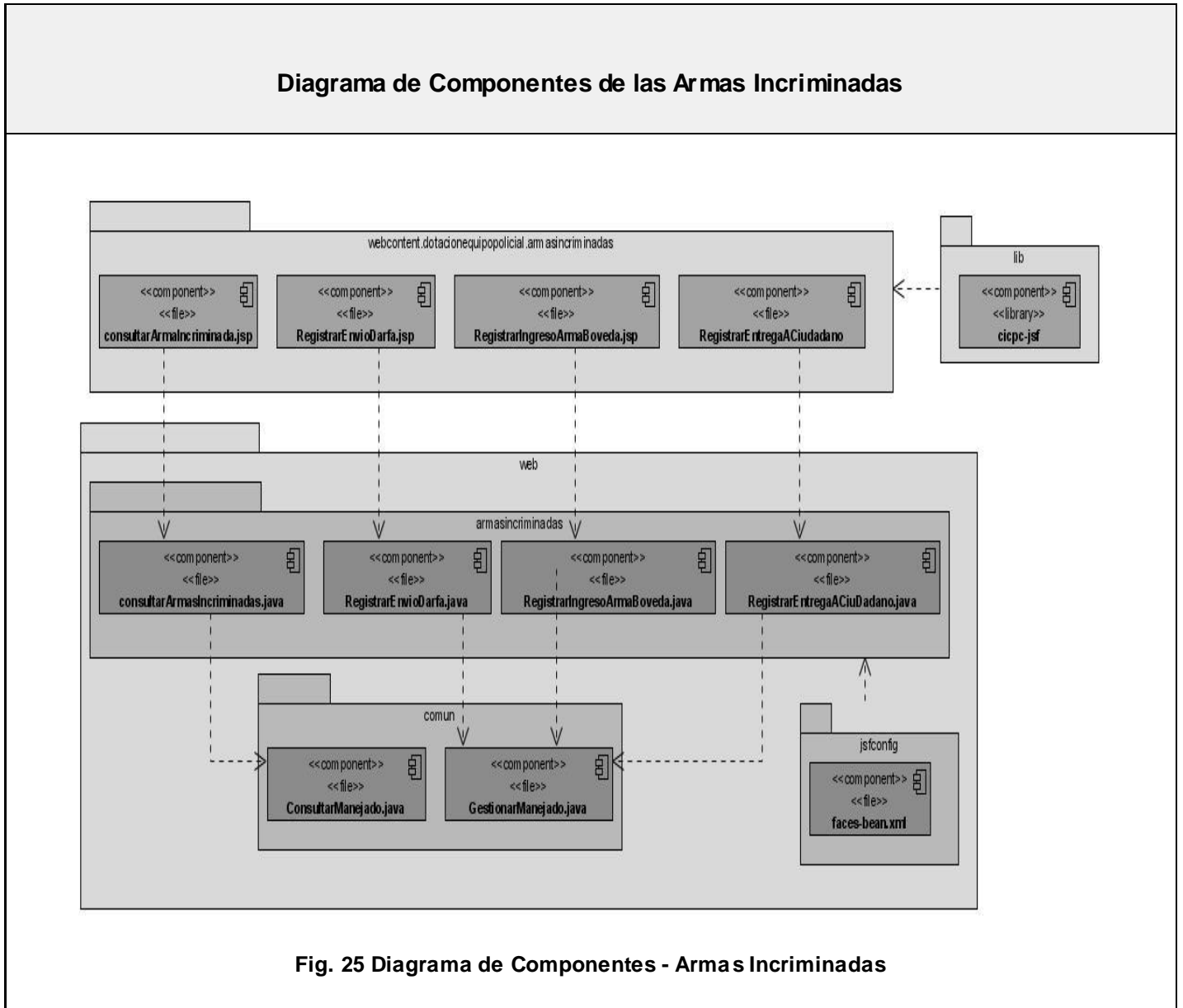




Diagrama de Componentes de los Equipos Policiales

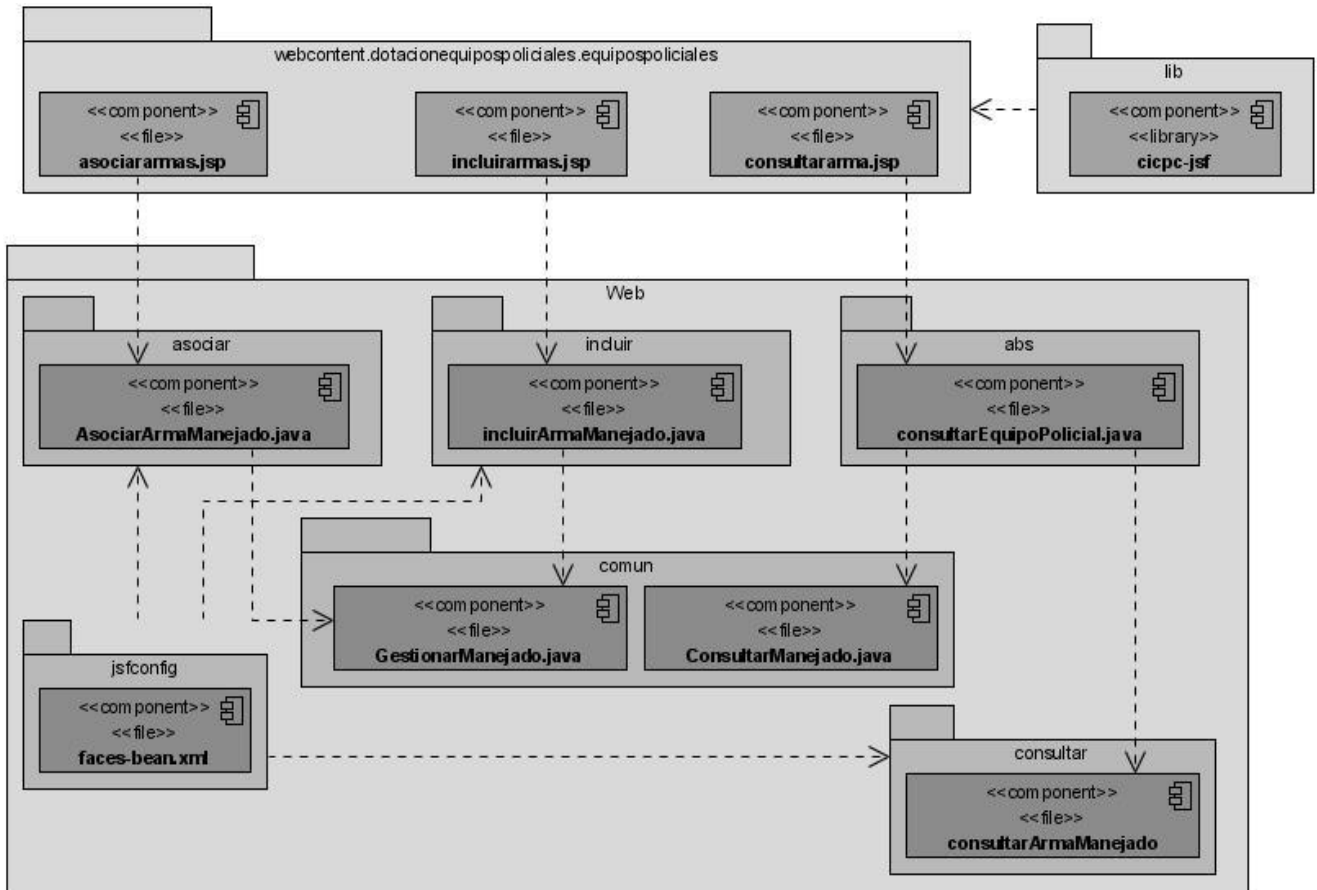


Fig. 26 Diagrama de Componentes – Equipos Policiales



Diagrama de Componentes de las Solicitudes de Equipos Policiales

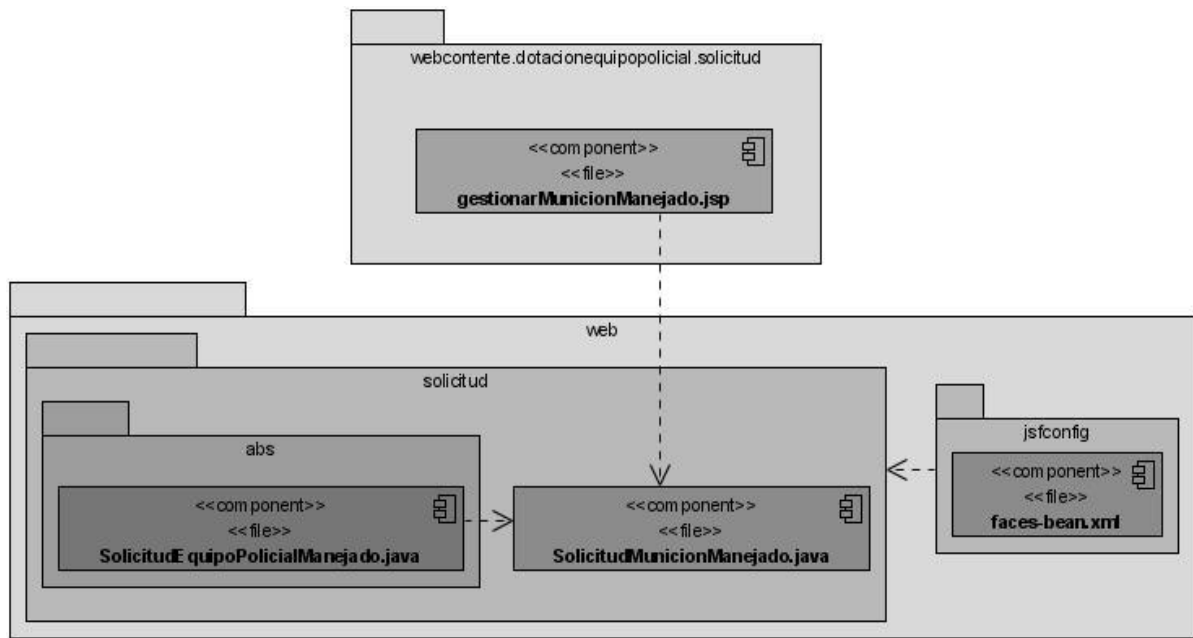
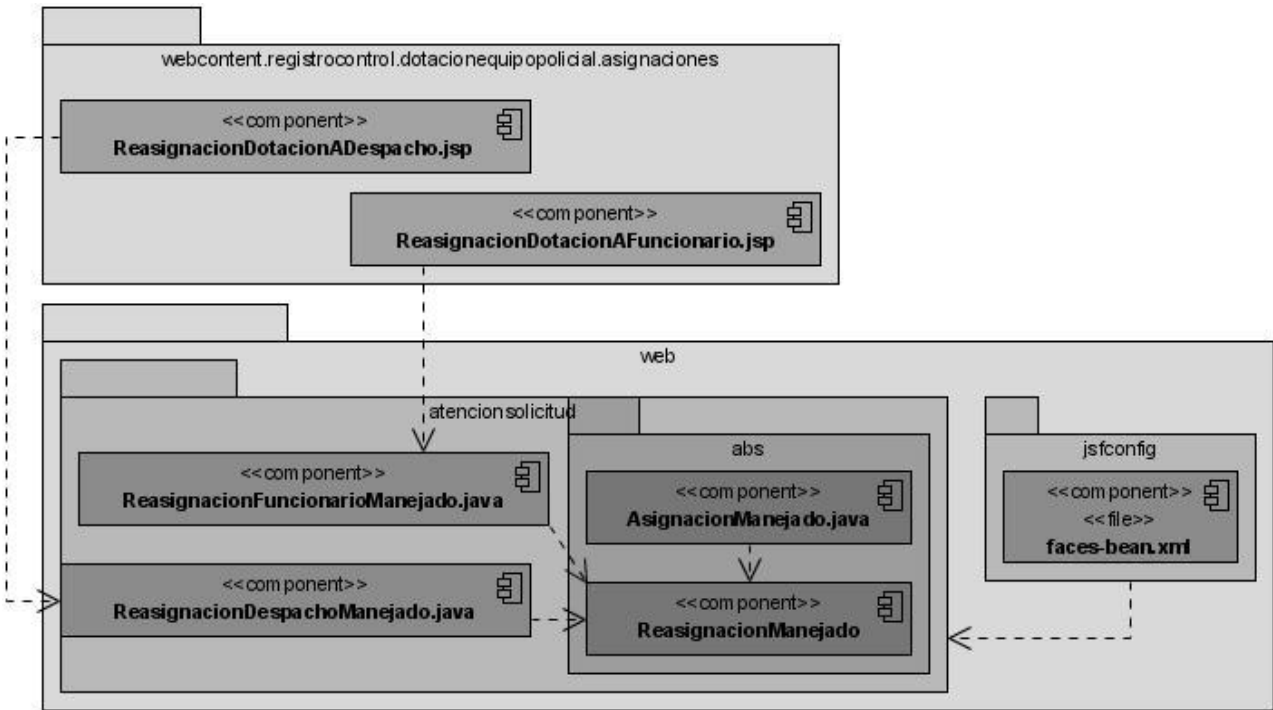


Fig. 27 Diagrama de Componentes - Solicitudes de Equipos Policiales

**Diagrama de Componentes Atención Solicitudes de Equipos Policiales.**



**Fig. 28 Diagrama de Componentes - Atención de Solicitud de Equipos Policiales**

### 2.5.3. ESTÁNDAR DE CODIFICACIÓN

#### **Convenciones de nombre**

Como principio, todos los nombres serán en español, exceptuando aquellos que correspondan al nombre de un patrón conocido, ejemplo: ServiceLocator, Facade, Builder, ect.

#### **Paquetes**

Los nombres de todos los paquetes comenzarán por CICPC y luego seguirán los nombres de acuerdo a la estructura de paquetes definida por la arquitectura, ejemplo:

```
<cicpc>.<nombre del subsistema>.<nombre del módulo>. <facade\web\config\dao\service\model>  
<cicpc>.<core><config\web\mensajería\seguridad\util>
```

#### **Clases**

Los nombres de las clases deben ser sustantivos. Cuando son compuestos tendrán la primera letra de cada palabra que lo forma en mayúscula. Intentar mantener los nombres de las clases simples y descriptivos. Usar palabras completas, evitar acrónimos y abreviaturas (a no ser que la abreviatura sea mucho más conocida que el nombre completo, como URL o HTML).

#### **Convención de nombre para los principales tipos de clases:**

1. **Bines manejados de JSF:** NombreManejado.  
Ejemplo: DatosDistribuidoraManejado.
2. **Clases del dominio:** Se les deja su nombre del dominio.  
Ejemplo: Expediente.
3. **Validadores de JSF:** NombreValidador.  
Ejemplo: NombreUsuarioValidador.



4. **Convertidores de JSF:** NombreConvertidor.  
Ejemplo: EstudianteConvertidor.
5. **Clases de Fachada:** NombreFacade.  
Ejemplo: ManejadorDistribucionesFacade.
6. **Clases que tengan rol de servicio:** NombreService.  
Ejemplo.: CasoService.
7. **Las clases que implementan interfaces:** NombreInterfaz Impl.  
Ejemplo: CasoServiceImpl.
8. **Servlets de java:** NombreServlet.  
Ejemplo: EscuchadorOcultoServlet.
9. **Filtros de java:** NombreFilter.  
Ejemplo: RegistradorFilter.

### ***Interfaces***

Los nombres de las interfaces siguen las mismas reglas que las clases.

### ***Métodos***

Los métodos deben ser verbos, cuando son compuestos tendrán la primera letra en minúscula y la primera letra de las siguientes palabras que lo forma en mayúscula.

Ejemplo: enviarSolicitud().

### ***Variables***

Excepto las constantes, todas las instancias y variables de clase o método empezarán con minúscula. Las palabras internas que lo forman (si son compuestas) empiezan con su primera letra en mayúscula. Los nombres de variables no deben empezar con los caracteres subrayado "\_" o signo del dólar "\$", aunque ambos están permitidos por el lenguaje.

Los nombres de las variables deben ser cortos pero con significado. Los de un solo caracter estarán incorrectos, excepto para variables e índices temporales, como son las de ciclos, para hacer intercambio de valores, ect. Nombres comunes para variables temporales son i, j, k, m, y n para enteros; c, d, y e para caracteres.



### Constantes

Los nombres de las variables declaradas como constantes deben ir totalmente en mayúsculas, separando las palabras con un subguión ("\_").

Ejemplo: ANCHURA\_MAXIMA = 999

### Organización de los ficheros fuentes. Reglas generales

Cada fichero fuente Java contiene una única clase o interfaz pública. Java permite que algunas clases o interfaces privadas que están asociadas a una clase pública, puedan ponerse en el mismo fichero que la clase pública, pero para CICPC esta práctica quedará prohibida.

Todo fichero fuente estará asociado solamente a una clase, o lo que es igual, todo fichero fuente solamente será portador de una única clase java.

<b>Partes de la declaración de una clase o interfaz</b>	
1- Comentario de documentación de la clase o interface.	Agregar comentario de documentación de clase o interface. Ver lineamientos para los comentarios.
2- Sentencia Import.	
3- Sentencia class o interface.	Primero las variables de clase public, después las protected, luego las de nivel de paquete (sin modificador de acceso) y posteriormente las private.
4- Variables de clase (static).	
5- Variables de instancia.	Primero las public, después las protected, luego las de nivel de paquete (sin modificador de acceso) y por últimas las private.
6- Constructores.	Agregar comentario de documentación.



7- Métodos.	<p>Se deben agrupar por funcionalidad más que por visión o accesibilidad. Por ejemplo, un método de clase privado puede estar entre dos métodos públicos de instancia.</p> <p>El objetivo es hacer el código más legible y comprensible.</p>
-------------	--

### ***Lineamientos para comentarios de documentación***

Los programas Java pueden tener dos tipos de comentarios: comentarios de implementación y comentarios de documentación. Los comentarios de implementación son aquellos que también se encuentran en C++, delimitados por `/*...*/`, y `//`. Los comentarios de documentación (conocidos como "doc comments") existen sólo en Java y se limitan por `/**...*/`.

Los comentarios de implementación son para comentar el código o para comentarios acerca de una implementación particular. Son aquellos clásicos y habituales comentarios que se insertan dentro del cuerpo de los métodos para describir qué se hace en determinadas líneas de código. Los comentarios de documentación son para describir la especificación del código, libre de una perspectiva de implementación y para ser leídos por desarrolladores que pueden no tener el código fuente a mano.

Cada comentario de documentación se encierra con los delimitadores de comentarios `/**...*/`, con un comentario por clase, interface o miembro (método o atributo). Este comentario debe aparecer justo antes de la declaración.

Comentarios de documentación para Clases e Interfaces:

Los comentarios de documentación para clases e interfaces deberán tener la siguiente información:

- Una breve descripción.
- La versión.
- Nombre del autor.



### ***Comentarios de documentación para métodos***

Los comentarios de documentación para métodos deberán tener la siguiente información:

- Una breve descripción del mismo.
- Los parámetros.
- El valor de retorno.
- Posible excepción.

### ***Inicialización***

Intentar inicializar las variables locales donde se declaran. La única razón para no inicializar una variable donde se declara es si el valor inicial depende de algunos cálculos que deben ocurrir.

### ***Colocación***

Poner las declaraciones sólo al principio de los bloques. No esperar al primer uso para declararlas; puede confundir a programadores no pre-avisados y limitar la portabilidad del código dentro de su ámbito de visibilidad.

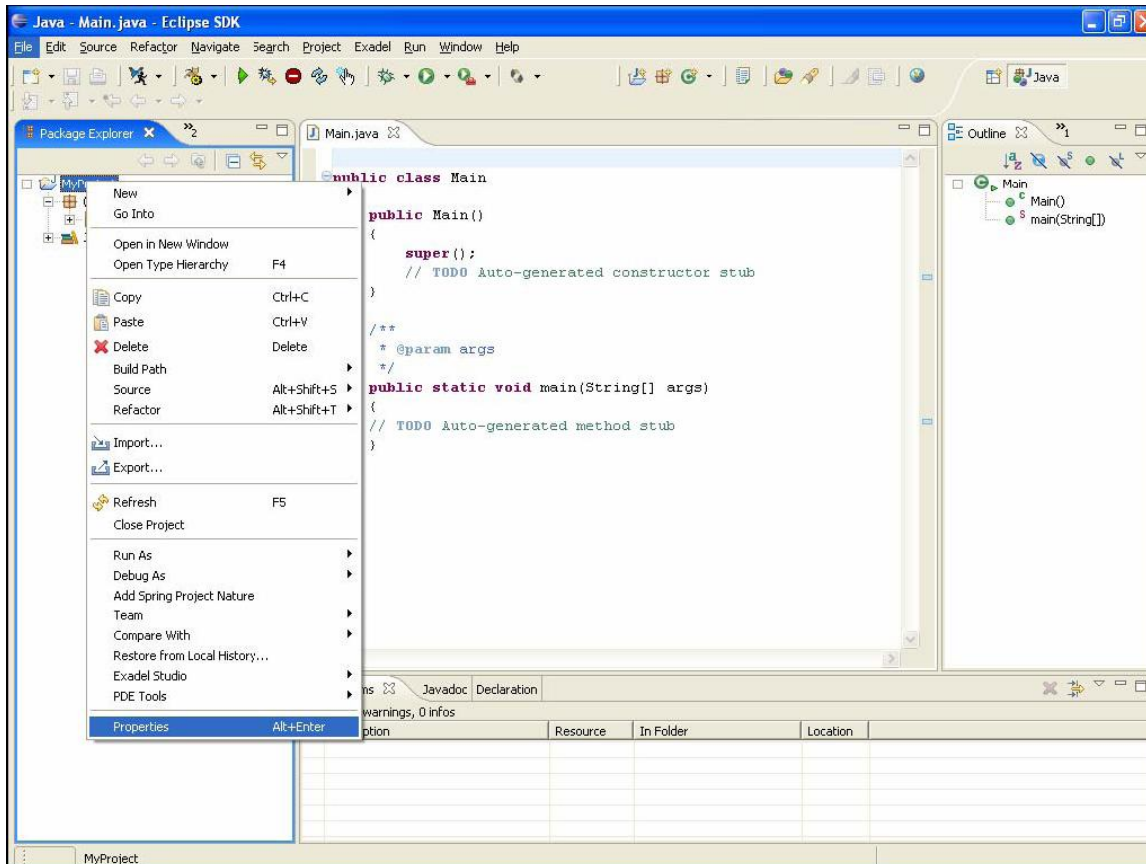
Nota: Evitar las declaraciones locales que ocultan declaraciones de niveles superiores. Por ejemplo, no declarar la misma variable en un bloque interno.

### ***Otras aclaraciones importantes***

Se usará idioma español para todos los nombres, excepto para aquellos elementos que representen algo bien conocido por su nombre en inglés o sus siglas.

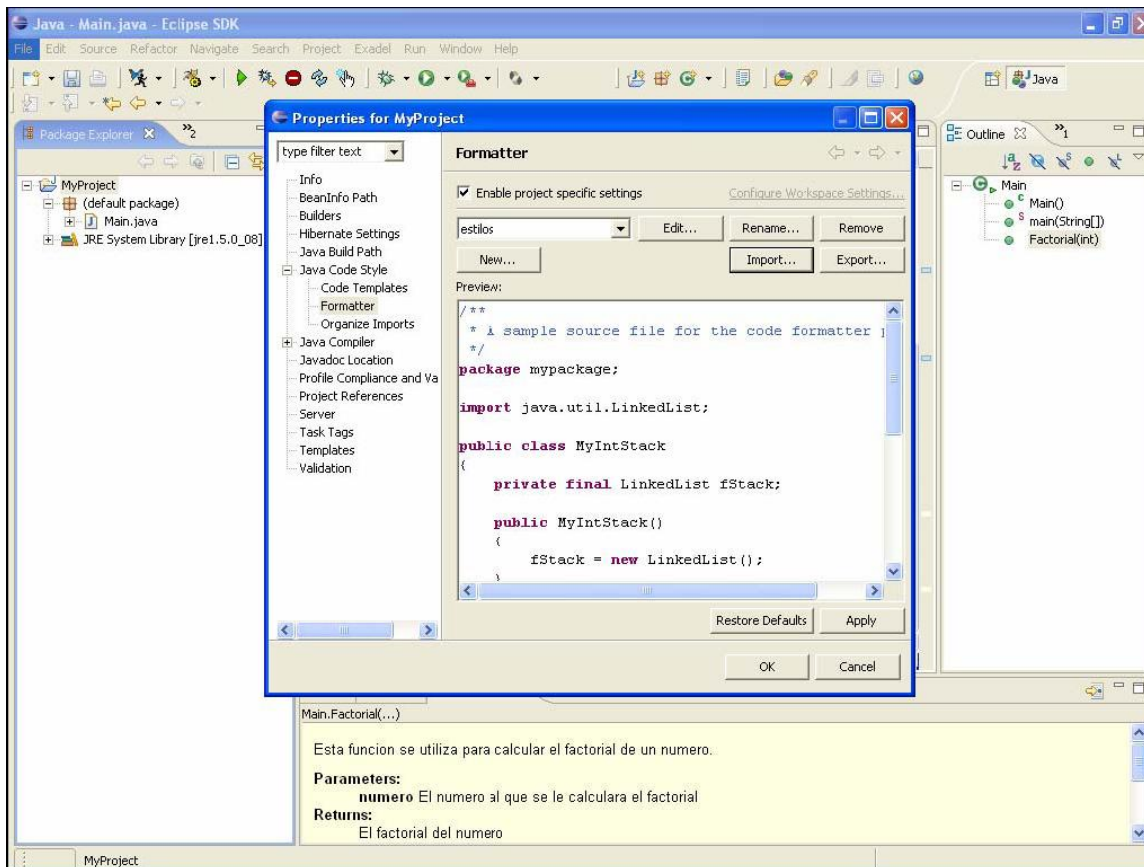
**Para aplicar un Estilo de Código al proyecto:**

1. Ir a las propiedades del proyecto.



2. Seleccionar la opción Java Code Style > Formatter.
3. Activar la opción Enable Project Specific Settings.





4. Importar el fichero adjunto (estilos.xml) con el botón Import.
5. Aplicar estilo (Apply).

Nota: Después de escribir código sólo será necesario presionar la combinación Ctrl+Shift+F para que el programa adopte el estilo.

## 2.6. CONCLUSIONES

En el desarrollo del capítulo se representó el modelo de análisis, el cual constituye una visión del sistema sobre las necesidades planteadas por el cliente. Además, se confeccionó la estructura de la aplicación mediante las clases del diseño con sus respectivas descripciones y los diagramas de paquetes con sus contratos. También, se mostraron los modelos lógico y físico de datos, las relaciones entre los componentes de implementación y por último, los estándares de codificación utilizados en el desarrollo del sistema.



## CAPÍTULO 3. VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

### 3.1. INTRODUCCIÓN

En este capítulo se presenta la validación de la propuesta de solución planteada para dar respuesta a los requisitos funcionales y no funcionales del sub-módulo: Dotación de Equipos Policiales perteneciente al módulo: Registro y Control del SIIPOL. Se explican los diferentes niveles y tipos de pruebas que se realizaron a la propuesta en función de validar dichos requisitos, así como la evaluación de los resultados obtenidos luego de aplicarle un conjunto de pruebas a la propuesta para de esta forma, fundamentar su correcta implementación y funcionamiento.

### 3.2. PRUEBAS DEL SOFTWARE

La prueba de software es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación.

Las pruebas se realizan a lo largo del desarrollo del sistema y no simplemente al final. Esto significa sacar a la luz problemas no conocidos y no demostrar la perfección de programas. Aunque la realización de las pruebas es un proceso tedioso constituye una serie de pasos esenciales que ayudan a asegurar la calidad del sistema. Las pruebas se realizan por subsistemas o módulos de programas según avanza el trabajo y se hacen en diferentes niveles y a diversos intervalos.

La realización de pruebas es una actividad en la cual un sistema o componente es ejecutado bajo condiciones o requisitos especificados, los resultados son observados y registrados, y se realiza una evaluación del sistema o componente. Su objetivo es descubrir los errores del software en la menor cantidad de tiempo posible, sin embargo no aseguran la ausencia de defectos.

Las siguientes definiciones son algunas de las recogidas en el diccionario de la IEEE en relación a las pruebas.

**Caso de prueba:** “un conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para un objetivo particular, ejemplo: ejercitar un camino concreto de un programa o



verificar el cumplimiento de un determinado requisito”. También se puede referir a la documentación en la que se describen las entradas, condiciones y salidas de un caso de prueba.

**Defecto:** “un defecto en el software como, por ejemplo, un proceso, una definición de datos o un paso de procesamiento incorrectos en un programa”.

**Fallo:** “La incapacidad de un sistema o de alguno de sus componentes para realizar las funciones requeridas dentro de los requisitos de rendimiento especificados”.

**Error:** tiene varias acepciones:

- La diferencia entre un valor calculado, observado o medido y el valor verdadero, especificado o teóricamente correcto. Por ejemplo, una diferencia de dos centímetros entre el valor calculado y el real.
- Un defecto. Por ejemplo, una instrucción incorrecta en un programa.
- Un resultado incorrecto. Por ejemplo, un programa ofrece como resultado de la raíz cuadrada de 36 el valor 7 en vez de 6.
- Una acción humana que conduce a un resultado incorrecto (una metedura de pata). Por ejemplo, que el operador o el programador pulse una tecla equivocada.

### 3.3. ESTRATEGIAS DE PRUEBAS

**Pruebas de Caja Blanca:** Las pruebas de caja blanca son una estrategia que se encarga de la estructura del código. Las pruebas escritas basadas en esta estrategia tienen cobertura para el código escrito, branches, caminos, sentencias y lógica interna del código.

Con el propósito de implementar métodos basados en esta estrategia el probador debe encargarse del código y por lo tanto debe tener conocimiento de la lógica interna de este.

**Métodos de prueba de Caja Blanca:**

- Prueba del camino básico.
- Prueba del flujo de datos.



- Prueba de condición.
- Prueba de bucles.

**Pruebas de Caja Negra:** Las pruebas de caja negra son una estrategia de prueba, tal y como su nombre lo indica “Caja Negra” no necesita el conocimiento del diseño interno ni de la lógica del código. El diseñador de prueba selecciona entradas válidas e inválidas y determina la salida correcta. Los tipos de pruebas basados en esta estrategia están totalmente enfocados en probar los requerimientos y las funcionalidades del software.

A fin de implementar un método basado en una estrategia de caja negra el probador debe ser minucioso con las especificaciones de los requerimientos del sistema y como usuario, debe saber cómo el sistema debe responder a una acción en particular.

#### **Métodos de Prueba de Caja Negra:**

- Particiones equivalentes.
- Métodos de prueba basados en grafos.
- Análisis de valores límites.
- Prueba de la tabla ortogonal.
- Adivinando el error.

### **3.4. NIVELES DE PRUEBAS**

La prueba es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo.

Los niveles de pruebas son los siguientes:

**Nivel de Unidad:** Pretende probar cada función en un archivo de programa simple (una clase en terminología de objetos). Las librerías de pruebas de unidad formalizan este trabajo al proporcionar clases para pruebas. La prueba de unidad ayuda a que el módulo se haga independiente.



**Nivel de Integración:** Algunas veces llamadas “integración y testeo”, son las pruebas de software en la cual módulos individuales son combinados y probados como un grupo. Son las pruebas posteriores a las pruebas unitarias y preceden el testeo de sistema.

**Nivel de Sistema:** Son las pruebas dirigidas a un software completo (ya integrado) para evaluar su obediencia a los requisitos especificados, las pruebas del sistema están dentro del alcance de las pruebas de caja negra y como tal no requieren del conocimiento del diseño interno ni de la lógica del código.

**Nivel de Aceptación:** El objetivo de las pruebas de aceptación es validar que un sistema cumple con el funcionamiento esperado y permitir al usuario de dicho sistema que determine su aceptación desde el punto de vista de su funcionalidad y rendimiento. Las pruebas de aceptación son definidas por el usuario del sistema y preparadas por el equipo de desarrollo, aunque la ejecución y aprobación final corresponden al usuario.

Para garantizar el cumplimiento de los requisitos funcionales del sub-módulo desarrollado se le realizaron varias iteraciones de pruebas de calidad, para las cuales fueron especificados una serie de casos de pruebas; estas pruebas, llevadas a cabo por un grupo de probadores arrojaron una serie de errores.

Iteración de Prueba	Total Conformidades SIIPOL	No Conformidades de Registro y Control	No Conformidades del sub-módulo DEP	% DEP/total
Internas 2da Etapa 1ra Iteración	667	55	41	74
Pruebas Calidad Interna 2da Entrega	618	106	9	8
Calidad UCI	3	1	-	-



SIIPOL 1ra Etapa 4ta Iteración				
Pruebas Cruzadas al SIIPOL V2.0	357	23	12	52

Los casos de prueba utilizados se encuentran registrados en la documentación general del proyecto, en caso de surgir alguna duda, puede solicitar consultar dicha información.



Contando directamente con la presencia e interacción del cliente con el sistema, se le realizaron al sub-módulo Dotación de Equipos Policiales 4 iteraciones de pruebas de aceptación y una piloto al software SIIPOL. Las especificaciones de las pruebas se muestran a continuación.

Iteración de Prueba	Total Conformidades SIIPOL	No Conformidades de Registro y Control	No Conformidades del sub-módulo DEP	Pedidos de Cambio para el sub-módulo DEP
Aceptación 1.5	42	12	1	8
Aceptación 2.0	23	7	0	6
Aceptación 2.01	15	0	0	11
Aceptación 2.02	25	0	-	-
Pruebas Piloto	78	10	4	4

Apreciando lo mostrado en la tabla anterior se puede concluir, que el nivel de aceptación del sistema por parte del cliente fue satisfactoriamente alto, debido a que no se detectaron problemas significativos en la aplicación; comprobando así, que se cumplieron cabalmente con los requisitos establecidos en la etapa de inicio.





### **3.5. CONCLUSIONES**

En este capítulo se mencionaron y detallaron los tipos de pruebas que pueden aplicarse a un sistema informático. Se expusieron además, los diferentes niveles de prueba que existen, así como las pruebas aplicadas al sub-módulo: Dotación de Equipos Policiales y por último, se reflejaron los resultados obtenidos de las pruebas realizadas.



## CONCLUSIONES

Después de realizar la investigación se puede concluir que las tareas de investigación y el objetivo general, así como los específicos del trabajo de diploma se cumplieron íntegramente, obteniéndose los siguientes resultados:

- Se analizaron los programas de gestión policial: STEGPOL, SIGEP y SIIPOL.
- Se realizó un estudio de la metodología, lenguajes, herramientas y tecnologías de desarrollo que se utilizaron en la confección de la solución.
- Se describieron y detallaron los patrones de diseño que se utilizaron en la elaboración de la aplicación.
- Se construyó una aplicación capaz de garantizar rapidez y efectividad en los procesos de gestión de equipos policiales.



## RECOMENDACIONES

El análisis, diseño e implementación del sub-módulo Dotación de Equipos Policiales perteneciente al módulo de Registro y Control de SIIPOL se efectuó satisfactoriamente, sin embargo, con el propósito de alcanzar mejores resultados se recomienda:

- Refactorizar el sub-módulo: Dotación de Equipos Policiales.
- Implementar los nuevos Casos de Uso resultantes de las Peticiones de Cambio de las pruebas.
- Poner este trabajo a disposición de la comunidad universitaria como referencia para proyectos similares, teniendo en cuenta las reglas de confidencialidad establecidas.
- Estudiar la posibilidad de realizar un sistema similar para Organismos de Cuba como el MININT y las FAR.
- Capacitar Personal para futuras versiones y refactorizaciones.



## BIBLIOGRAFÍA

1. **Sun.** java.sun.com. [En línea] [Citado el: 10 de octubre de 2008.] <http://java.sun.com/javaee/javaserverfaces/overview.html>.
2. Manual Java. *Manual Java*. [En línea] [Citado el: 25 de agosto de 2008.] <http://www.manual-java.com/manualjava/caracteristicas-java.html>.
3. **Eclipse.** Eclipse.org. *Eclipse.org*. [En línea] [Citado el: 25 de agosto de 2008.] <http://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.html>.
4. **Gosling, James, et al.** *The Java™ Language Specification Third Edition*. s.l.: Addison Wesley Professional, 2005.
5. **Flores., Maykell Frómata.** *Guía de estilo de código para el proyecto CICPC*. Ciudad de La Habana, Cuba : s.n., 9 de abril de 2007.
6. **Rivero Guevara, Humberto.** *Análisis, diseño e implementación del módulo Aprehensión del SIIPOL*. Ciudad de La Habana : s.n.
7. **Pereira Ojeda, Maikel y Gago Martinez, Yudanis.** *Análisis, Diseño e Implementación del módulo Experticias Criminalísticas del Sistema de Investigación e Información Policial*. Ciudad de La Habana : s.n.
8. **Universidad Distrital "Francisco Jose de Caldas".** [udistrital.edu.co](http://www.udistrital.edu.co). *udistrital.edu.co*. [En línea] <http://www.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML - Pruebas de software/node55.html>.
9. **Buzzle.** Buzzle.com. *Buzzle.com*. [En línea] [Citado el: 23 de febrero de 2009.] <http://www.buzzle.com/editorials/4-10-2005-68350.asp>.
10. —. Buzzle.com. *Buzzle.com*. [En línea] [Citado el: 23 de febrero de 2009.] <http://www.buzzle.com/editorials/4-10-2005-68349.asp>.



11. **Mendoza Sanchez, María A.** informatizate.net. *informatizate.net*. [Online] [Cited: diciembre 8, 2008.] [http://www.informatizate.net/articulos/metodologias\\_de\\_desarrollo\\_de\\_software\\_07062004.html](http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html).
12. **S. Pressman, Roger.** *Ingeniería del Software. Un Enfoque Practico*. s.l. : Felix Varela.
13. **Visual Paradigm.** visual-paradigm.com. *visual-paradigm.com*. [En línea] <http://www.visual-paradigm.com>.
14. **Sun.** java.sun.com. *java.sun.com*. [Online] [Cited: enero 25, 2009.] <http://java.sun.com/blueprints/patterns/MVC-detailed.html>.
15. **Kruchten, Philippe.** *Rational Unified Process, The: An Introduction, Third Edition*. s.l. : Addison Wesley, 2003.
16. **Gamma, Erich, y otros.** *Design Patterns: Elements of Reuseable Object-Oriented Software*. s.l. : Addison-Wesley, 1995.
17. **Toll Palma, Yuniet del Carmen y Mendoza Torres, Ylennis.** *Propuesta de manual de procedimiento de Pruebas de Sistema*. Ciudad de La Habana : s.n.
18. *IEEE*. 1990.



## GLOSARIO DE TÉRMINOS

### A

Aplicaciones Web: Es un sistema informático que los usuarios utilizan accediendo a un servidor Web a través de Internet.

Adaptador enchufable: Es una clase que tiene ya definidas internamente las posibles adaptaciones de su interfaz.

### B

Base de datos: Un conjunto de información almacenada en memoria auxiliar que permite acceso directo y un conjunto de programas que manipulan esos datos.

### C

Código abierto: Es el término con el que se conoce al software distribuido y desarrollado libremente.

### D

Diagramas: Facilitan el entendimiento de grandes cantidades de datos y la relación entre diferentes partes de los datos.

### F

FrameWork: Es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado.

### H

HTTPS: Es un protocolo de red basado en el protocolo HTTP, destinado a la transferencia segura de datos de hipertexto, es decir, es la versión segura de HTTP.

### I



Interfaz: Es la forma en que los usuarios pueden comunicarse con una computadora.

IDE: Integrated Development Environment. Un entorno de desarrollo integrado es un programa compuesto por un conjunto de herramientas para un programador.

## **J**

Java: Es un lenguaje de programación de alto nivel orientado a objetos.

## **L**

Logs: Es un registro oficial de eventos durante un período de tiempo en particular.

## **M**

Mac OS: Macintosh Operating System (Sistema Operativo de Macintosh).

## **P**

Plataforma: Se refiere al sistema operativo o a sistemas complejos que a su vez sirven para crear programas, como las plataformas de desarrollo.

Principio de Hollywood: Toma su nombre del cliché dictado a los actores amateurs en Hollywood: “No nos llames nosotros te llamaremos”, describe la forma en que un objeto resuelve sus dependencias con otros objetos.

## **S**

Software: Se refiere al equipamiento lógico o soporte lógico de un computador digital.

Sistema Operativo: Es un conjunto de programas de computadora destinados a permitir una administración eficaz de sus recursos.

## **T**

Tecnologías: Conjunto de teorías y de técnicas que permiten el aprovechamiento práctico del conocimiento científico.

**W**

W3C: Es un consorcio internacional, donde el personal a tiempo completo y el público en general trabajan conjuntamente para desarrollar estándares Web.



# ANEXOS

## Anexo 1 [Interfaz] Solicitud Equipos Policiales a Funcionario

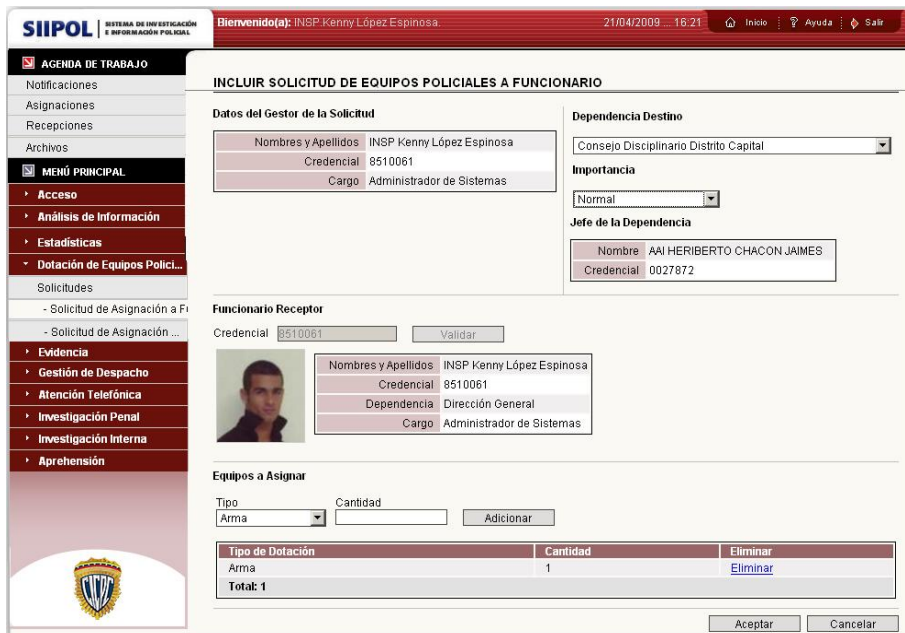


The screenshot shows the SIIPOL web application interface. The top navigation bar includes the SIIPOL logo, the user name 'Bienvenido(a): INSP Kenny López Espinosa', the date '21/04/2009 ... 16:21', and links for 'Inicio', 'Ayuda', and 'Salir'. The left sidebar contains a menu with 'AGENDA DE TRABAJO' (Notificaciones, Asignaciones, Recepciones, Archivos) and 'MENÚ PRINCIPAL' (Acceso, Análisis de Información, Estadísticas, Dotación de Equipos Polici..., Solicitudes, Evidencia). The main content area is titled 'INCLUIR SOLICITUD DE EQUIPOS POLICIALES A FUNCIONARIO' and contains the following fields:

- Datos del Gestor de la Solicitud:** Nombres y Apellidos: INSP Kenny López Espinosa; Credencial: 8510061; Cargo: Administrador de Sistemas.
- Dependencia Destino:** A dropdown menu.
- Importancia:** A dropdown menu.
- Funcionario Receptor:** Credencial: [input field]; Validar button.

Buttons for 'Aceptar' and 'Cancelar' are located at the bottom right of the form.

## Anexo 2 [Interfaz] Solicitud Equipos Policiales a Funcionario

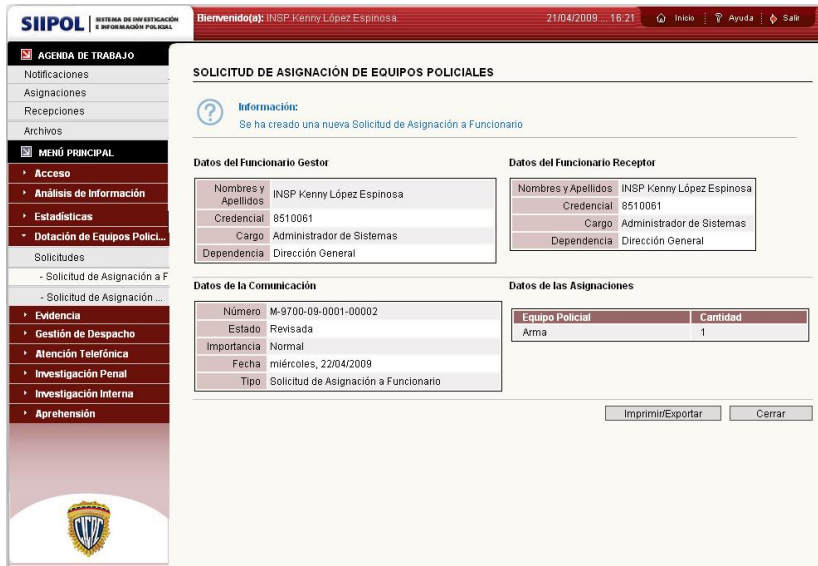


This screenshot shows the same SIIPOL interface as Anexo 1, but with additional data entered. The 'Dependencia Destino' is set to 'Consejo Disciplinario Distrito Capital' and 'Importancia' is 'Normal'. The 'Jefe de la Dependencia' field is populated with 'Nombre: AAI HERIBERTO CHACON JAIMES' and 'Credencial: 0027872'. The 'Funcionario Receptor' section now includes a photo and full details for 'INSP Kenny López Espinosa' (Credencial: 8510061, Dependencia: Dirección General, Cargo: Administrador de Sistemas). The 'Equipos a Asignar' section shows a dropdown for 'Arma' and a 'Cantidad' of 1. Below this is a table:

Tipo de Dotación	Cantidad	Eliminar
Arma	1	<a href="#">Eliminar</a>
<b>Total:</b>	<b>1</b>	

'Aceptar' and 'Cancelar' buttons are at the bottom right.

**Anexo 3 [Interfaz] Ver Solicitud Asignación a Funcionario**



**BIENVENIDO(A):** INSP Kenny López Espinosa      21/04/2009 ... 16:21      Inicio    Ayuda    Salir

**AGENDA DE TRABAJO**

- Notificaciones
- Asignaciones
- Recepciones
- Archivos

**MENÚ PRINCIPAL**

- Acceso
- Análisis de Información
- Estadísticas
- Dotación de Equipos Polici...
- Solicitudes
  - Solicitud de Asignación a F...
  - Solicitud de Asignación ...
- Evidencia
- Gestión de Despacho
- Atención Telefónica
- Investigación Penal
- Investigación Interna
- Aprehensión

**SOLICITUD DE ASIGNACIÓN DE EQUIPOS POLICIALES**

**Información:**  
Se ha creado una nueva Solicitud de Asignación a Funcionario

**Datos del Funcionario Gestor**

Nombres y Apellidos	INSP Kenny López Espinosa
Credencial	8510061
Cargo	Administrador de Sistemas
Dependencia	Dirección General

**Datos del Funcionario Receptor**

Nombres y Apellidos	INSP Kenny López Espinosa
Credencial	8510061
Cargo	Administrador de Sistemas
Dependencia	Dirección General

**Datos de la Comunicación**

Número	M-9700-09-0001-00002
Estado	Revisada
Importancia	Normal
Fecha	miércoles, 22/04/2009
Tipo	Solicitud de Asignación a Funcionario

**Datos de las Asignaciones**

Equipo Policial	Cantidad
Arma	1

Imprimir/Exportar    Cerrar

**Anexo 4 [Interfaz] Asignar Dotacion Funcionario**



**BIENVENIDO(A):** INSP Kenny López Espinosa      21/04/2009 ... 17:26      Inicio    Ayuda    Salir

**AGENDA DE TRABAJO**

- Notificaciones
- Asignaciones
- Recepciones
- Aprobaciones
- Revisiones
- Remisiones
- Borradores
- Archivos

**MENÚ PRINCIPAL**

- Acceso
- Dotación de Equipos Polici...
- Evidencia
- Auditoría
- Administración

**ASIGNAR DOTACIÓN A FUNCIONARIO**

**Datos de la Bóveda**

Responsable	INSP Kenny López Espinosa
Cargo	Administrador de Sistemas
Dependencia	Dirección General
Fecha	martes, 21/04/2009 17:37

**Datos de la solicitud seleccionada**

No.	M-9700-09-0001-00003
Dependencia Emisora	Dirección General
Funcionario Emisor	CG Administrator Administrator
Fecha de Emisión	martes, 21/04/2009 17:21

**Funcionario Receptor**

Nombres y Apellidos	INSP Kenny López Espinosa
Cargo	Administrador de Sistemas
Dependencia	Dirección General

**Autenticación del Receptor**

Usuario: kenny  
 Contraseña:

**Equipos seleccionados**

Tipo	Serial	Marca	Modelo	Eliminar
Arma	C-12345	BENELLI	NOVA	<a href="#">Eliminar</a>
<b>Total:</b>	<b>1</b>			

Incluir    Cancelar



## Anexo 5 [Interfaz] Ver Asignación de Dotación a Funcionario

**SIIPOL** SISTEMA DE INVESTIGACIÓN E INFORMACIÓN POLICIAL Bienvenido(a): INSP Kenny López Espinosa 21/04/2009 ... 17:26 Inicio Ayuda Salir

**AGENDA DE TRABAJO**

- Notificaciones
- Asignaciones
- Recepciones
- Aprobaciones
- Revisiones
- Remisiones
- Borradores
- Archivos

**MENÚ PRINCIPAL**

- Acceso
- Dotación de Equipos Polici...
- Evidencia
- Auditoría
- Administración

**ASIGNAR DOTACIÓN A FUNCIONARIO - VER ATENCIÓN DE SOLICITUD**

**Información:**  
Se ha incluido una nueva Asignación de Equipo Policial a Funcionario

**Datos generales**

Responsable de la Bóveda	INSP Kenny López Espinosa
Cargo	Administrador de Sistemas
Dependencia	Dirección General
Fecha	21/04/2009 05:37 PM

**Datos de la Solicitud**

No. de Solicitud	M-9700-09-0001-00003
Dependencia emisora	Dirección General
Funcionario emisor	INSP Kenny López Espinosa
Fecha de emisión	22/04/2009

**Datos del receptor de la dotación**

Nombres y Apellidos	INSP Kenny López Espinosa
Cargo	Administrador de Sistemas
Dependencia	Dirección General

**Equipos asignados**

Equipo	Serial	Marca	Modelo
Arma	C-12345	BENELLI	NOVA

Imprimir/Exportar Cerrar

## Anexo 6 [Interfaz] Consultar Equipo Policial

**SIIPOL** SISTEMA DE INVESTIGACIÓN E INFORMACIÓN POLICIAL Bienvenido(a): INSP Kenny López Espinosa 21/04/2009 ... 17:57 Inicio Ayuda Salir

**AGENDA DE TRABAJO**

- Notificaciones
- Asignaciones
- Recepciones
- Aprobaciones
- Revisiones
- Remisiones
- Borradores
- Archivos

**MENÚ PRINCIPAL**

- Acceso
- Gestión de Despacho
- Atención Telefónica
- Investigación Penal
- Investigación Interna
- Aprehensión
- Archivos Históricos
- Investigación Criminalística

**CONSULTAR EQUIPOS POLICIALES**

Armas  Chalecos  Esposas  Otros Equipos

**Criterios de Búsqueda**

Serial Primario  Tipo  Serial Secundario  Tipo  Serial Terciario  Tipo

Tipo de Arma  Marca  Modelo

Estado

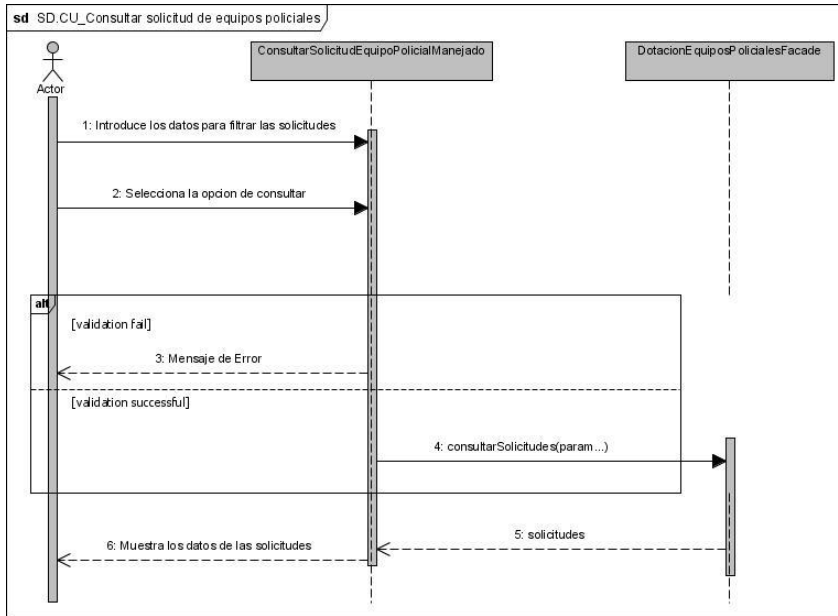
Consultar Nueva Búsqueda Cerrar

**Listado de Resultados**

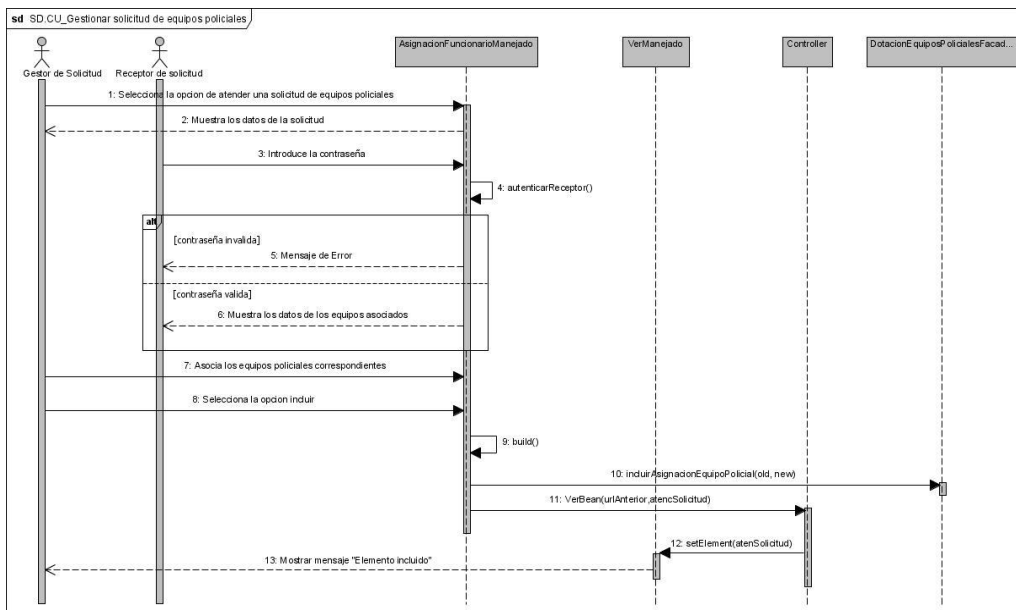
Tipos y Seriales	Tipo	Marca	Modelo	Estado
C-12345	Ametralladora	BENELLI	NOVA	Asignado
C-999	Ametralladora	BENELLI	NOVA	Nuevo
C-717 C-111	Ametralladora	BENELLI	NOVA	Nuevo

Total: 3 Imprimir/Exportar

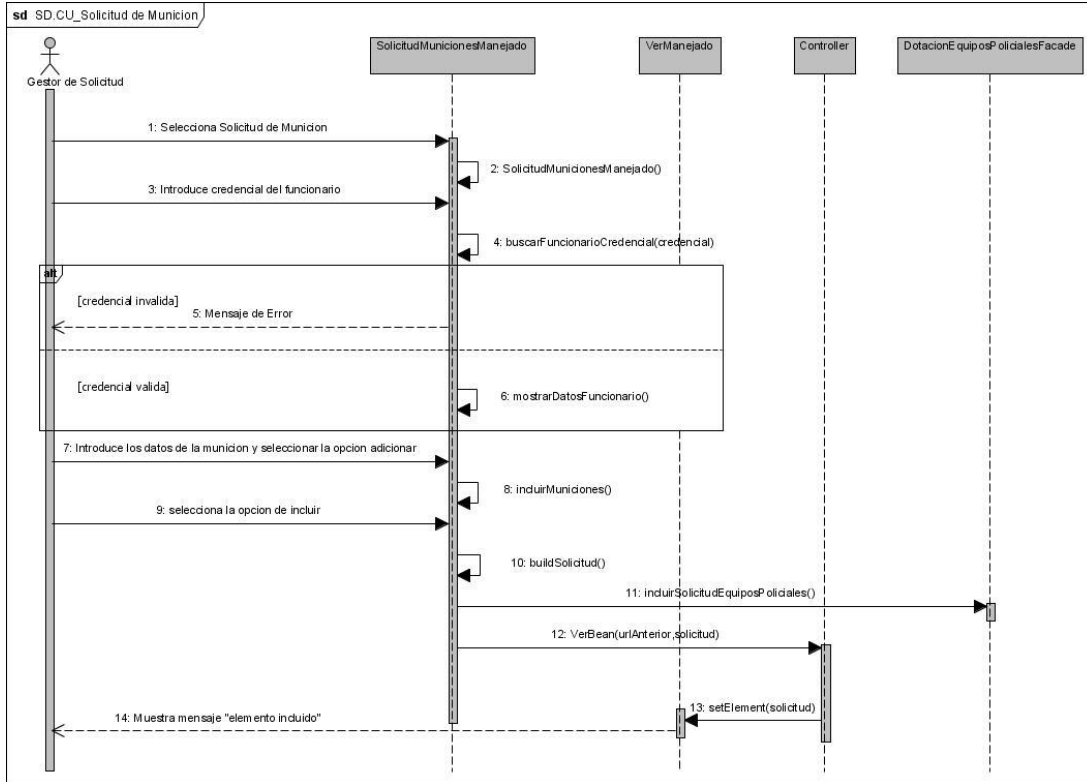
**Anexo 7 [Diagrama de Secuencia] Consultar Solicitud de Equipos Policiales**



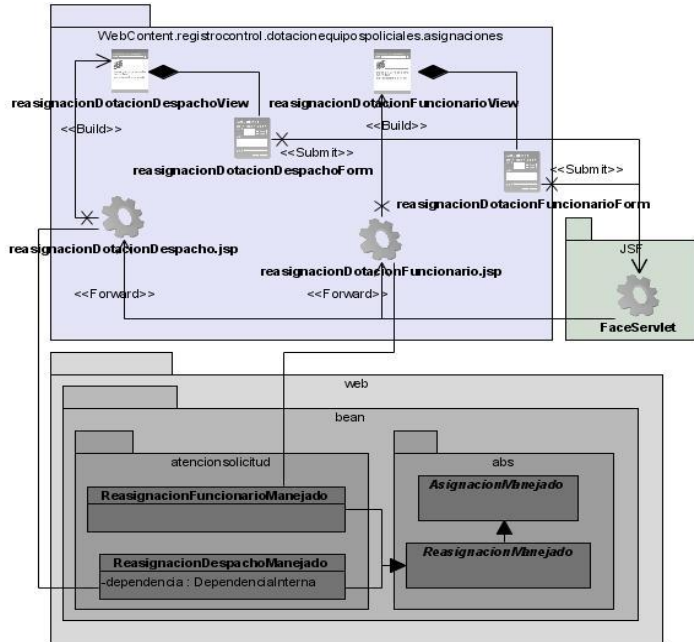
**Anexo 8 [Diagrama de Secuencia] Gestionar Solicitud de Equipos Policiales**



Anexo 9 [diagrama de Secuencia] Solicitud de Munición



**Anexo 10 [Diagrama de Clases] Reasignar Equipos Policiales**



**Anexo 11 [Diagrama de Clases] Registrar Envío DARFA**

