UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 8

# Análisis, diseño e implementación de la capa lógica del negocio y acceso a datos del módulo **Estadística** perteneciente al Sistema de Investigación e Información Policial

---

**Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas**

**Autor:**
Gabriel Terry Morales

**Tutor:**
MSc. Yadiel Ramos Rodríguez

**Cotutora:**
Ing. Dayana Daniel Hernández

Ciudad de La Habana, 2010
"Año del 52 de la Revolución"

**Declaración de autoría**

Declaro que soy el único autor del trabajo “Análisis, diseño e implementación de la capa lógica del negocio y acceso a datos del módulo Estadística perteneciente al Sistema de Investigación e Información Policial” y autorizo a la Facultad 8 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los _____ días del mes de _____ del año 2010.

_____

**Autor**
Gabriel Terry Morales

_____

**Tutor**
Msc. Yadiel Ramos Rodríguez

**Agradecimientos**

*A todos los que de una forma u otra contribuyeron con este trabajo investigativo aportando su granito de arena, en especial a Yele, Dayan y Niño Prodigio "Yadiel" que me ayudaron incondicionalmente día y noche.*

*A mi suegra, por ayudarme incondicionalmente en cada momento.*

*Al piquete de Saldo, en especial Isabel dueña de la logística, importante.*

*A mis revisores técnicos, Aurelio, Rosa Elena, Sayli, Arlenys, mi vecina Carmen, en fin todos lo que velaron por la calidad del documento.*

**Dedicatoria**

*A la Revolución, a la Universidad de las Ciencias Informáticas y en especial a la Facultad 8 por hacer realidad nuestro sueño de formarnos como profesionales.*

*A mi familia y en especial a mi mamá Zoila, que gracias a su intransigencia no titubeó en forjarme en un ingeniero, a pesar de los problemas mentales que tuve en mi infancia.*

*A mi papá Jaime (El Gallo) que pesar de la distancia en los dos primeros años de mi carrera, supo guiarme y darme buenos consejos.*

*A mis hermanas Tezadis y Lizandra las cuales son la razón de mí existir.*

*A mi novia Aymee por apoyarme en los momentos más difícil de mi carrera y haberme colmado la vida de felicidad.*

*A mi madrina Raquel, que a pesar de nuestro encuentro tardío, me protege cada día con su aché, que dios la bendiga.*

**Resumen**

El Cuerpo de Investigaciones Científicas Penales y Criminalísticas de Venezuela (CICPC), constituye una institución única de su tipo dentro del territorio bolivariano. Su principal objetivo es detectar, procesar y esclarecer hechos delictivos cometidos a diario. Esta investigación centra su objetivo general en la automatización de los procesos de la Coordinación Nacional de Estadística del CICPC, pues actualmente no se recogen las estadísticas generadas por la institución y la información procesada no es fiable, ni certera. Esta idea se materializará con el diseño y desarrollo del Módulo Estadística, porción de software que permitirá una mejor gestión de la información, la reducción de los tiempos de respuesta de las peticiones y abarcará los procesos que se desarrollan diariamente en el coordinación.

Palabras clave: estadística, gestión policial, Sistema Estadísticos Policiales.

**Índice**

Introducción ..... 10
Capítulo 1. Fundamentación Teórica ..... 15
1.1 Estadística ..... 15
1.1.1Sistemas Estadísticos Policiales ..... 15
1.1.1.1 Sistema Integrado de Estadísticas Policiales ..... 15
1.1.1.2 Sistema de Denuncias Policiales ..... 16
1.2 Proceso de desarrollo de software ..... 16
1.2.1 Proceso Unificado de Desarrollo (RUP) ..... 18
1.2.2 Proceso Ágil Unificado (AUP) ..... 19
1.2.3Selección del proceso de desarrollo ..... 21
1.3 Lenguaje de Modelado ..... 22
1.4 Herramienta CASE ..... 23
1.4.1 Visual Paradigm ..... 23
1.5 Plataforma de Desarrollo ..... 24
1.6 Lenguajes de programación ..... 24
1.6.1 Java ..... 25
1.7 Frameworks utilizados en la solución ..... 26
1.7.1 Spring ..... 26
1.7.2 Hibernate ..... 27
1.7.3 JUnit ..... 28
1.8 Entorno de Desarrollo Integrado ..... 29
1.9 Sistema Gestor de Base de Datos ..... 30
1.10 Arquitectura Técnica ..... 30
1.11 Conclusiones ..... 32
Capítulo2. Análisis, Diseño e Implementación de la Propuesta de Solución ..... 33
2.1 Análisis del Módulo de Estadística ..... 33
2.1.1 Modelo de Casos de uso ..... 34
2.1.2 Casos de Uso significativos ..... 35
2.1.3 Requisitos no funcionales relevantes ..... 35
2.2 Diseño del Módulo Estadística. ..... 37

2.2.1 Tarjetas CRC. ..... 38
2.2.2 Diagrama de paquetes ..... 42
2.2.3 Patrones de diseño ..... 43
2.2.4 Realización de los Casos de Uso ..... 44
2.3 Modelo de Datos ..... 53
2.3.1 Diagrama de Clases Persistentes ..... 53
2.3.2 Diagrama de Entidad Relación ..... 54
2.4 Modelo de Implementación ..... 55
2.4.1Diagramas de subsistemas de implementación ..... 55
2.4.2Diagrama de componentes ..... 56
2.4Estándar de codificación ..... 59
2.5Conclusiones ..... 59
Capitulo 3. Validación de la solución propuesta ..... 60
3.1Introducción ..... 60
3.2Métodos de prueba. Pruebas Aplicadas al Sistema. ..... 60
3.2.1Pruebas Aplicadas al Sistema ..... 61
3.3Conclusiones ..... 71
Conclusiones ..... 72
Recomendaciones ..... 73
Referencias Bibliográficas ..... 74
Bibliografías ..... 75
Anexos ..... 78
Anexo 1: Tipos de Estadísticas ..... 78
Anexo 2: Tipos de Gráficos ..... 78
Anexo 3: Estilos para generar reportes. ..... 79
Anexo 4: Tarjetas CRC para la jerarquía del Módulo Estadística ..... 80
Anexo 5: Patrón Facade ..... 82
Glosario ..... 84

Índice de Figura.

Figura 1: Matriz de Boehm y Turner aplicada al proyecto CICPC ..... 18

Figura 2: Fases y flujos de trabajo definidos por AUP. ..... 21
Figura 3: Estructura de Spring ..... 27
Figura 4: Arquitectura de Hibernate ..... 28
Figura 5: Solución Propuesta. ..... 31
Figura 6: Modelo N-Capa. ..... 31
Figura 7: Modelo Vista Controlador. ..... 32
Figura 8: Modelo de Caso de Uso. Módulo Estadística. ..... 34
Figura 9: Tarjetas CRC <EstadísticaUtil>. ..... 39
Figura 10: Tarjetas CRC <ReporteEstadiscoDelitoService>. ..... 40
Figura 11: Tarjetas CRC <ReporteEstadiscoCTACService>. ..... 41
Figura 12: Diagrama de Paquetes. ..... 42
Figura 13: CU Generar Estadísticas de Delitos. ..... 45
Figura 14: CU Generar Estadísticas de Actas Procesales. ..... 45
Figura 15: CU Generar Estadísticas de Delitos<Estadística Predefinida>. ..... 46
Figura 16: CU Generar Estadísticas de Delitos<Estadística Personalizada>. ..... 47
Figura 17: CU Generar Estadísticas de Delitos< Estadística Comparativa>. ..... 48
Figura 18: CU Generar Estadísticas de Delito<Estadística Predefinida>. ..... 50
Figura 19: CU Generar Estadísticas de Delitos<Estadística Personalizada>. ..... 51
Figura 20: CU Generar Estadísticas de Delitos<Estadística Comparativas>. ..... 52
Figura 21: Diagrama de Clases Persistentes. ..... 53
Figura 22: Diagrama de Entidad Relación ..... 54
Figura 23: Diagrama de subsistemas de Implementación ..... 56
Figura 24: Diagrama de Componentes. Vista Acceso a Datos ..... 57
Figura 25: Diagrama de Componentes. Vista Lógica de Negocio ..... 58
Figura 26: Método Caja Blanca. ..... 61

Figura 27: Método de Caja Negra ..... 61
Figura 28: Pruebas Cruzadas ..... 63
Figura 29: Pruebas Cruzadas (Comparación con el resto de los módulos) ..... 63
Figura 30: Pruebas Cruzadas (no conformidades) ..... 64
Figura 31: Prueba de Calidad Interna ..... 65
Figura 32: Prueba de Calidad Interna (no conformidades) ..... 65
Figura 33: Prueba de Liberación ..... 66
Figura 34: Prueba de Liberación (Comparación con el resto de los módulos) ..... 67
Figura 35: Prueba de Liberación (no conformidades) ..... 67
Figura 36: Prueba de Aceptación (pedidos de cambios) ..... 68
Figura 37: Prueba de Aceptación (no conformidades) ..... 68
Figura 38: Prueba Piloto (pedido de cambios) ..... 69
Figura 39: Prueba Piloto (no conformidades) ..... 69
Figura 40: Resumen General de las Pruebas ..... 70

## Índice de Tablas

Tabla 1:Métodos de la clase EstadisticaUtil. ..... 39
Tabla 2: Métodos de la clase ReporteEstadiscoDelitoService. ..... 40
Tabla 3: Método de la clase ReporteEstadiscoCTACService ..... 41

**Introducción**

La República Bolivariana de Venezuela es un país situado al norte de América del Sur, constituido como un estado democrático y social, de derecho y de justicia, autónomo y soberano. Su revolución tiene características únicas, distintas a todos los procesos socialistas que se han conocido en el siglo XX. Es conocida como la *¨La revolución bonita¨,* la revolución que triunfó sin disparar un solo tiro, ha traído bienestar y un futuro que promete.

Sin embargo, se ha declarado en algunas ocasiones, como uno de los países más violentos del mundo. La inseguridad es reconocida como el principal problema del país y los venezolanos están sometidos a esta realidad. A pesar de este inconveniente, el problema de la seguridad en Venezuela es solucionable, sólo hace falta trabajo, compromiso y voluntad política.

Para garantizar la tranquilidad ciudadana existe el Cuerpo de Investigaciones Científicas, Penales y Criminalísticas de la República Bolivariana de Venezuela (CICPC), institución encargada de garantizar la eficiencia en la investigación del delito.

CICPC cuenta con una aplicación informática que se conoce como el Sistema Integrado de Información Policial (SIIPOL), que posee una tecnología obsoleta y no abarca todos los procesos que se realizan actualmente en esta institución, lo que impide a los funcionarios el correcto desarrollo de las investigaciones que conducen al esclarecimiento de los hechos delictivos. Dentro de la estructura organizativa de esta institución se encuentra la Coordinación Nacional de Estadísticas perteneciente al CICPC, que cuenta con un Sistema Estadístico conformado por seis módulos en forma de menú de elección.

1. MÓDULO DE ESTADÍSTICAS DIARIAS: Permite realizar consultas de relación diaria de montos denunciados y recuperados, casos de mayor importancia y detenciones y expedientes remitidos.
2. MÓDULO DE ESTADÍSTICAS SEMANALES: Contiene una serie de opciones para hacer las consultas semanales por tipo, días, hora o modus operandi de los delitos, casos conocidos por dependencia, sitio del suceso o zonas, total de delitos denunciados, zonas de mayor índice delictivo, total de casos conocidos, variación inter-semanal de delitos, expedientes remitidos a fiscalía por dependencia, expedientes remitidos a fiscalía respecto a casos conocidos y casos conocidos por dependencias.

3. MÓDULO DE ESTADÍSTICAS MENSUALES: Permite realizar estadísticas diarias y semanales sobre montos denunciados y recuperados, casos de mayor importancia, decomiso de estupefacientes, expedientes remitidos, niños y adolescentes extraviados, entre otros.
4. MÓDULO INCLUIR ESTADÍSTICA: Permite realizar inclusión de estadísticas diarias y semanales sobre montos denunciados y recuperados, casos de mayor importancia, decomiso de estupefacientes, expedientes remitidos, niños y adolescentes extraviados.
5. MÓDULO MODIFICAR ESTADÍSTICAS: Permite realizar modificaciones de las estadísticas que se encuentran en el sistema.
6. MÓDULO ELIMINAR ESTADÍSTICAS: Permite eliminar las estadísticas que se encuentran en el sistema. (1)

Este sistema posee interfaces poco amigables y su funcionamiento es engorroso (mediante comandos). Además, los informes son generados en documentos que no cuentan con el formato institucional, por lo cual su calidad es altamente limitada.

Adicionalmente, el sistema no abarca todas las estadísticas que se generan en el CICPC, puesto que todas están relacionadas con el Sistema de Investigación Penal, mientras que existen otras direcciones como la Coordinación Nacional de Criminalística y la Coordinación Nacional de Ciencias Forenses, por citar algunas, que generan sus estadísticas de forma manual o utilizan otras aplicaciones independientes a la institución. En estos casos no siempre se cuenta con los datos actualizados y la disponibilidad de la información es restringida, por lo que el proceso de generar cálculos estadísticos se dificulta y la información procesada no es altamente fiable, ni certera.

Con el propósito de desarrollar un nuevo sistema que permita mejorar la gestión de la información, disminuyendo los tiempos de respuesta de las peticiones, se firmó un contrato en el marco de las relaciones Cuba – Venezuela por la colaboración de los países del ALBA, con el fin de modernizar y automatizar el CICPC creando el Sistema de Investigación e Información Policial (SIIPOL).

A partir de la aplicación de la ingeniería de requerimientos basada en un análisis profundo del funcionamiento de la institución, así como sus procesos de negocio, se generó un conjunto de requisitos funcionales y no funcionales, que resumen las condiciones y

capacidades que debe cumplir el nuevo sistema y que dan al traste con el antiguo Sistema Integrado de Información Policial.

Para llevar a cabo la implementación de la solución informática se ha modelado un sistema dividido en subsistemas, siendo uno de ellos el Módulo de Estadística, que constituye el objeto de la presente investigación.

Planteada la situación problemática se puede enunciar el **problema a resolver** a partir de la siguiente interrogante: ¿Cómo garantizar el cumplimiento de los requisitos funcionales y no funcionales asociados al Módulo de Estadística del Sistema de Investigación e Información Policial (SIIPOL) desde el punto de vista de la lógica de negocio y acceso a datos?

El **objetivo general** de este trabajo es**:** Desarrollar las capas de lógica de negocio y acceso a datos del Módulo de Estadística perteneciente al Sistema de Investigación e Información Policial (SIIPOL).

**Objetivos Específicos:**

- Valorar los resultados obtenidos en la aplicación de la ingeniería de requerimientos.
- Analizar, definir, estructurar y detallar mecanismos de diseño que garanticen el cumplimiento de las necesidades del Módulo de Estadística.
- Analizar, diseñar e implementar las capas de lógica de negocio y acceso a datos del Módulo de Estadística perteneciente al SIIPOL respetando la arquitectura definida.
- Integrar el Módulo de Estadística con el resto de los módulos del SIIPOL.

El **objeto de estudio** de la presente investigación son los sistemas estadísticos asociados a la investigación policial, enmarcándose en la lógica de negocio y acceso a datos del Módulo de Estadística perteneciente al Sistema de Investigación e Información Policial (SIIPOL) como **campo de acción.**

La **idea a defender** es**:** Si se diseñan e implementan las capas de lógica de negocio y acceso a datos, asociadas al Módulo de Estadística, se garantizará el cumplimiento de los requisitos funcionales y no funcionales obtenidos en la Ingeniería de Requerimientos .

Para dar cumplimiento a los objetivos planteados se planificaron las siguientes **tareas investigativas**:

1. Investigar acerca de otras aplicaciones o soluciones similares.

2. Analizar los procesos, las herramientas y lenguaje a utilizar en el análisis, diseño e implementación del Módulo de Estadística perteneciente al SIIPOL.
3. Investigar la aplicación de Patrones de Diseño.
4. Estudiar y analizar la arquitectura definida para el SIIPOL.
5. Elaborar los artefactos UML necesarios.
6. Desarrollar los componentes que dan solución a los objetivos propuestos.
7. Probar los componentes desarrollados una vez que se integran al resto de los módulos del SIIPOL.

Con el desarrollo exitoso de las tareas investigativas, se espera lograr una porción de software funcional que satisfaga los estándares de calidad requeridos y que cumpla con la totalidad de requisitos funcionales y no funcionales arrojados a partir de la aplicación de la ingeniería de requerimientos a los procesos de la Coordinación Nacional de Estadística.

**Métodos Científicos**

En la investigación se utilizan los **métodos científicos,** que permiten valorar los sistemas estadísticos asociados a la investigación policial en su esencia, con el propósito de ver su funcionamiento y obtener las buenas prácticas que ayuden a desarrollarla.

Dentro de los métodos científicos utilizados se encuentra el **histórico-lógico**, pues se analizarán los sistemas estadísticos más reconocidos, desde el punto de vista de cómo ha evolucionado y su funcionamiento. Se utilizará el método **inductivo – deductivo,** ya que se identificarán las herramientas, metodologías y técnicas a utilizar para el desarrollo del trabajo, con el que se obtendrán los conocimientos generales y posteriormente será valorado cuál usar en particular.

Las **técnicas investigativas** desempeñarán un papel importante para posibilitar la obtención de un conjunto de datos elementales que enriquecerán de manera especial la investigación. Una de las técnicas utilizadas es la **entrevista**, pues mediante sesiones planificadas con especialistas en la materia, se recopiló información sobre las herramientas a utilizar para dar cumplimiento a los requerimientos del sistema que ayudan a tener una visión de cómo desarrollar el Módulo de Estadística del SIIPOL. Otra de las técnicas que se utiliza es la **observación**, ya que observando los resultados obtenidos en toda la investigación proporcionará la medida de la línea de trabajo a seguir, garantizando el cumplimiento de los objetivos propuestos.

El contenido a desarrollar en la investigación está estructurado en tres capítulos. A continuación se muestra un resumen de cada uno de ellos.

**Capítulo 1: Fundamentación teórica.** Incluye un estado del arte del tema tratado, con el objetivo de analizar los sistemas estadísticos asociados a la investigación policial, así como herramientas, metodología y lenguaje para darle soporte tecnológico a la solución informática.

**Capítulo 2: Análisis, Diseño e Implementación de la Propuesta de Solución.** Se abordarán los elementos relacionados con el Análisis y Diseño de la Solución Propuesta y se confeccionarán las descripciones de los casos de uso, en los cuales se encuentran plasmados los requisitos que deben ser cumplidos, tanto los funcionales como no funcionales. Luego se desarrollarán los modelos de datos y de implementación. Finalmente, se generan un gran número de diagramas durante el proceso de desarrollo, que ayudarán a la documentación necesaria para el proyecto.

**Capítulo 3: Validación de la Solución Propuesta.** En este capítulo se estudiarán los métodos de prueba: caja blanca y caja negra. Se seleccionará una estrategia de validación y verificación de errores, para posteriormente ser aplicada al módulo, con el objetivo de encontrar la mayor cantidad de fallos.

# Capítulo 1. Fundamentación Teórica

En este capítulo se estudian los Sistemas de Estadísticas encaminados a la investigación policial. Se identifican los principales problemas del Cuerpo de Investigaciones Científicas Penales y Criminalísticas (CICPC) valorando las posibles mejoras y nuevas funcionalidades para el desarrollo del Sistema de Investigación e Información Policial (SIIPOL), principalmente el Módulo de Estadística. Además de realizar un estudio de las metodologías, lenguaje y herramientas de desarrollo a utilizar.

## 1.1 Estadística

La estadística, en general, es la ciencia que trata sobre la recopilación, organización, presentación, análisis e interpretación de datos numéricos con el fin de realizar una toma de decisión más efectiva.

Conceptualmente expresado no existe definición exacta. La más admitida es la del estadístico Óscar Vázquez Mínguez, que define la Estadística como *"La ciencia que tiene por objeto aplicar las leyes de la cantidad a los hechos sociales para medir su intensidad, deducir las leyes que los rigen y hacer su predicción próxima*". (2)

### 1.1.1 Sistemas Estadísticos Policiales

Un Sistema Estadístico Policial puede definirse como un sistema integrado capaz de gestionar los procesos que involucran la información generada dentro y fuera de la organización, mejorando directamente la posición competitiva de la institución.

Las principales características que pueden lograrse al utilizar este tipo de sistema son: búsqueda fácil y dinámica, generación de cálculos estadísticos, seguridad y veracidad de la información, disponibilidad de la información de forma rápida, organización y control de las labores cotidianas. Mediante dichas capacidades es posible mejorar la gestión de la información, disminuir los tiempos de respuesta de las peticiones y cubrir la totalidad de las estadísticas generadas en la Institución. A continuación se caracterizan varios sistemas estadísticos existentes, de los cuales se obtienen las buenas prácticas.

#### 1.1.1.1 Sistema Integrado de Estadísticas Policiales

Entre los Sistemas Estadísticos Policiales más relevantes se encuentra el Sistema Integrado Estadístico Policial, el cual se especializa en llevar las estadísticas del monitoreo de la criminalidad; el desarrollo y fortalecimiento de la inteligencia policial.

Debido al elevado índice de criminalidad a nivel mundial los organismos policiales se ven obligados a optimizar las acciones de investigación criminal con el objetivo de lograr el esclarecimiento de los hechos delictivos. Es por eso que estos sistemas desempeñan un papel fundamental en las labores policiales y de investigación. Ellos llevan a la práctica el avance de la tecnología desarrollando un sistema de estadísticas policiales que implementa la vigilancia electrónica con circuito cerrado de televisión. Esto permitirá la mejor planificación de las acciones, el monitoreo de la criminalidad y la vigilancia de los espacios de alto índice de delincuencia. (3)

#### 1.1.1.2 Sistema de Denuncias Policiales

El Sistema de Denuncias Policiales es reconocido por el gran volumen de reportes estadísticos sobre delincuencia y el acceso a mapas del delito, mediante el Sistema Geográfico Policial; gracias a estas especialidades se incluye como un Sistema Estadístico Policial.

Este tipo de sistema contrarresta el alto índice de violencia a nivel mundial, particularmente en Perú. Interconecta a 136 comisarías que intercambian estadísticas sobre denuncias e investigaciones realizadas. También tiene una Comisaría Virtual, por la cual el ciudadano puede realizar su denuncia a través de Internet. Tiene como objetivo lograr una búsqueda fácil y dinámica así como disponibilidad de la información de forma rápida. (4)

Los sistemas anteriormente mencionados ayudarán en el desarrollo de la investigación y en la toma de decisiones, obteniendo de ellos las buenas prácticas que despuntan de los mismos, como son las estadísticas del monitoreo de la criminalidad, la vigilancia de altos índices de delincuencia y el intercambio de estadísticas entre las comisarías.

## 1.2 Proceso de desarrollo de software

Un proceso define quién está haciendo qué, cuándo, y cómo alcanzar un determinado objetivo. En la Ingeniería del Software el objetivo es construir un producto de software o mejorar uno existente. Un proceso efectivo proporciona normas para el desarrollo eficiente del software de calidad; en consecuencia, reduce el riesgo y hace el proyecto más predecible.

El proceso de desarrollo de software debería ser capaz de evolucionar durante muchos años. Durante esta evolución convendría limitar su alcance, en un momento del tiempo dado, a

las realidades que permitan las tecnologías, herramientas, personas y patrones de organización.

Existen cinco factores críticos involucrados en determinar la relativa idoneidad para un proyecto específico de una metodología independientemente de si ésta es ágil o formal, los cuales son: criticidad, personal, dinamismo, cultura y tamaño. Un proyecto que se ajuste a una metodología determinada para cuatro de estos factores pero no para el quinto, requiere una evaluación y probablemente mezclar en un proceso de desarrollo métodos ágiles y formales. (5)

A continuación se dará a conocer un breve resumen de los factores expuestos en la matriz.

Criticidad: El proyecto está encaminado a modelar los procesos de la Institución CICPC, sin embargo no influye en la toma de decisiones sobre las vidas o bienes de la sociedad, simplemente es una herramienta que agiliza el flujo de comunicación en la Institución y suministra los elementos ventajosos para el desarrollo de la investigación.

Personal: El proyecto está constituido principalmente por estudiantes, personas sin experiencia, las cuales nunca habían trabajado en un proyecto de esta envergadura, solo un pequeño grupo que lidera el proyecto tiene experiencia en un proceso de desarrollo de este tipo.

Dinamismo: Aunque los requisitos de la Institución no son altamente cambiantes, se ha evidenciado que existen diferencias en cuanto a la forma de trabajo en un despacho como consecuencia del cambio de jefes o se detectan nuevas necesidades una vez que el cliente se enfrenta realmente al software; por lo que existe en la institución una leve tendencia al cambio.

Cultura: El porciento de adaptación a entornos caóticos está dado por la necesidad de adaptabilidad del software a un entorno de requisitos cambiantes. Si bien está claro que en la primera fase del proyecto se realiza un levantamiento de requisitos y sobre esto se basa todo el desarrollo, el proyecto continuamente está aceptando pedidos de cambio por parte del cliente e insertándolo en distintas versiones del software que se generen.

Tamaño: El proyecto cuenta con un equipo de más de 80 personas.

La matriz propuesta por Boehm y Turner (5) para evaluar dichos factores aplicada al proyecto CICPC está representada en la siguiente figura.

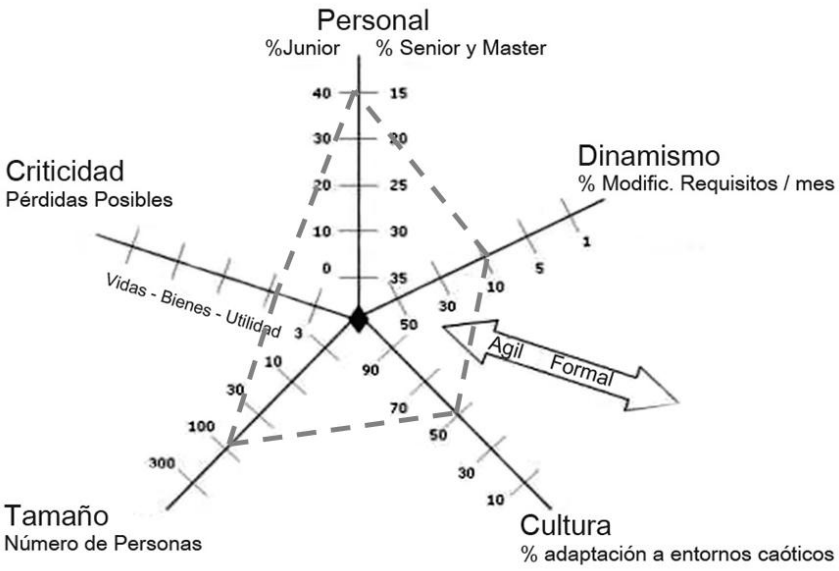



**Figura 1: Matriz de Boehm y Turner aplicada al proyecto CICPC**

Como se puede observar el factor criticidad tiende a los enfoques ágiles, mientras que el personal tiende a enfoques formales. Los factores tamaño, cultura y dinamismo se encuentran en el centro. Por esta razón se decidió ir ajustando las especificidades del proyecto entre las metodologías formales y ágiles para adecuar el proceso de desarrollo a seguir a las necesidades requeridas.

Esta selección está basada en la necesidad de un proceso bien definido teniendo en cuenta las grandes dimensiones del software a construir, pero donde lo más importante son las personas y su interacción sobre las herramientas y los procesos, el software que funcione por encima de la documentación excesiva y la respuesta al cambio por encima del seguimiento de un plan.

### 1.2.1 Proceso Unificado de Desarrollo (RUP)

RUP es un proceso de software genérico que puede ser utilizado para una gran cantidad de tipos de sistemas de software, para diferentes áreas de aplicación, organizaciones y niveles de competencia. Provee un enfoque disciplinado en la asignación de tareas y responsabilidades dentro de una organización de desarrollo. Su meta es asegurar la producción de software de muy alta calidad que satisfaga las necesidades de los usuarios finales, dentro de un calendario y presupuesto predecible. Es una metodología a tener en

cuenta debido a la experiencia desarrollada en la universidad porque en un primer momento fue la metodología adoptada por el proyecto.

### 1.2.2 Proceso Ágil Unificado (AUP)

El Proceso Ágil Unificado (*Agil Unified Process AUP*), es una versión simplificada de RUP que describe de forma simple y fácil de comprender el uso de técnicas ágiles para el desarrollo de aplicaciones que permanezcan dentro de los conceptos de RUP. AUP promueve principios ágiles que fueron utilizados desde las etapas iniciales del proyecto CICPC de los cuales se acentúan particularmente los siguientes:

1. **Integración continua:** En CICPC se optó por mantener una sola revisión para todo el equipo de programación y se recomendaron lapsos de integraciones de pocas horas a no más de un día. Durante todo el desarrollo del proyecto siempre se estuvo integrando tanto el código como los demás artefactos.
2. **Refactorización:** En el proyecto CICPC se mantuvo una depuración y simplificación constante del sistema. Una vez que se añadía alguna funcionalidad era necesario ser críticos para encontrar puntos de simplificación en el código. Durante todo el desarrollo del proyecto, se estuvo refactorizando constantemente para obtener un código lo más limpio posible.
3. **Propiedad colectiva del código:** El código en CICPC puede ser modificado por cualquier miembro del equipo de desarrollo y los programadores se rigen por estándares de codificación definidos, por tanto, luego de un tiempo razonable, cualquier programador tiene la habilidad de conocer y dominar todo el código.
4. **Bienestar del programador:** Se deben lograr condiciones de trabajo óptimas para programar “a máxima velocidad”. Cubrir una semana de 40 horas. Está demostrado que la productividad no se incrementa con horas extras, pues los programadores cansados son menos productivos y más propensos a errores.

Se hace necesario destacar un elemento esencial que distingue al Proceso Ágil Unificado del Proceso Unificado de Desarrollo. De acuerdo con las metodologías tradicionales, el cliente no está obligado a mantener una interacción constante con el equipo de desarrollo, puede seguir el avance del proyecto a través de reuniones planificadas y con un largo período de ocurrencia, siendo necesario por ello una mayor documentación para que exista un control de todo el proceso. AUP establece que debe determinarse una persona que se encargue de

velar por el cumplimiento de los requisitos y las prioridades del sistema, esta persona puede ser o bien el cliente en cuestión o un equipo de analistas. En el caso particular, descrito en el documento, este importante rol fue desempeñado por el equipo de analistas, quienes fueron los encargados de hacer cumplir, por parte del equipo de desarrollo, las exigencias del cliente de manera eficiente.

AUP mantiene las cuatro fases propuestas por RUP (inicio, elaboración, construcción y transición), no así para los flujos de trabajo donde el Modelo, primer flujo establecido por AUP, abarca las disciplinas Modelo de Negocio, Requerimientos y Análisis y Diseño. Seguidamente se detallarán las actividades fundamentales de cada fase:

1. **Inicio**: Se alcanza un acuerdo entre todos los interesados respecto a los objetivos del ciclo de vida para el proyecto, generando ámbitos de proyecto, casos de negocio, síntesis de la arquitectura posible y el alcance del proyecto.
2. **Elaboración**: Se establece la línea base de la arquitectura del sistema y proporciona una base estable para el diseño y el esfuerzo de implementación de la siguiente fase.
3. **Construcción**: Se completa el desarrollo del sistema basado en la línea de la arquitectura.
4. **Transición**: Se garantiza que el software esté listo para entregarlo a los usuarios.

Un flujo de trabajo muestra la secuencia de actividades, cómo se estructuran las tareas, cuál es su orden sucesivo, cómo se sincronizan, cómo fluye la información que soporta las tareas y cómo se le hace seguimiento al cumplimiento de las mismas. A continuación se muestran los flujos de trabajo de dicha metodología**:**

1. **Modelo**: Entender el negocio de la organización, el dominio del problema que el proyecto aborda e identificar una solución viable para abordarlo.
2. **Implementación**: Transformar el modelo en un código ejecutable y realizar una prueba de nivel básico en una unidad particular de prueba.
3. **Prueba**: Ejecutar una evaluación de los objetivos para asegurar la calidad. Esto incluye encontrar defectos, validar que el sistema funcione como fue diseñado y verificar que los requerimientos estén completos.
4. **Despliegue**: Planificar la entrega del sistema y ejecutar el plan para que el sistema esté disponible para los usuarios finales.

5. **Administración de Configuración**: Administrar el acceso a los entregables o productos del proyecto. Incluye no sólo el rastreo de versiones del producto en el tiempo, sino que también incluye controlar y administrar los cambios que ocurran.
6. **Administración de Proyecto**: Dirigir las actividades que se llevan a cabo en el proyecto. Incluye administración del riesgo, dirección de personas (asignar tareas, seguimiento de los procesos), y coordinar con los sistemas y personas fuera del alcance del proyecto para que termine en tiempo y dentro del presupuesto.
7. **Ambiente**: Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización. (6)

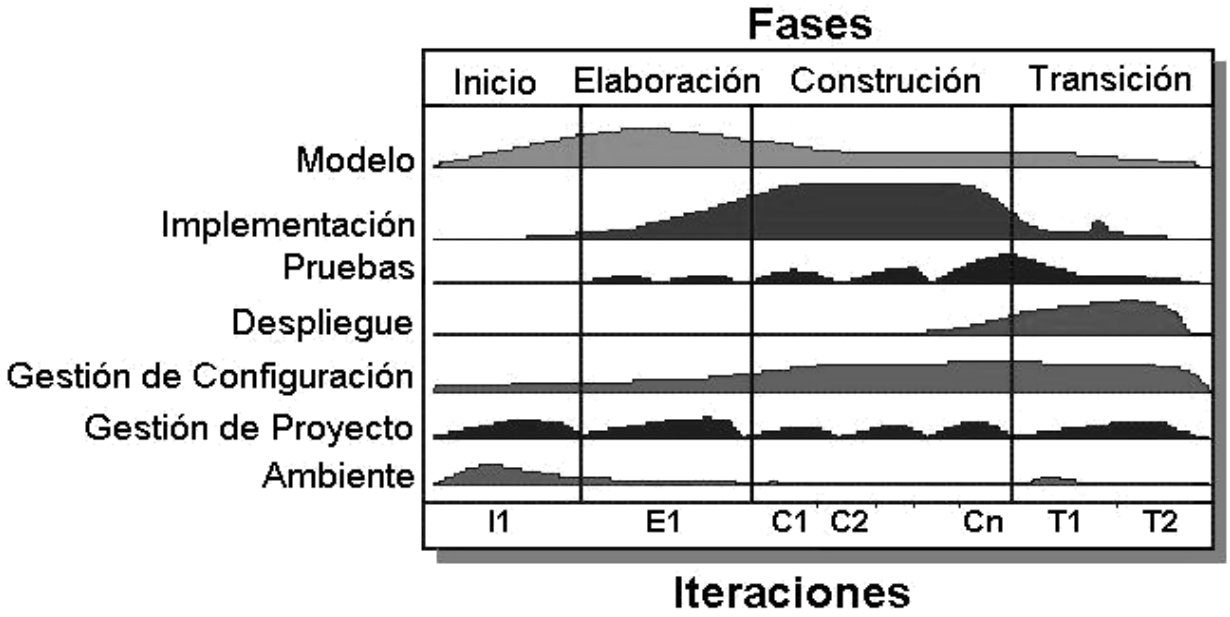



**Figura 2: Fases y flujos de trabajo definidos por AUP.**

### 1.2.3 Selección del proceso de desarrollo

De acuerdo con la realidad presentada durante los tres años de desarrollo que lleva el proyecto, se considera que la metodología AUP se ajusta más a su entorno, teniendo en cuenta

que incorpora los elementos ágiles necesarios a la metodología RUP para obtener un software en tiempo y con la calidad requerida.

### 1.3 Lenguaje de Modelado

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema y aspectos concretos como expresiones del lenguaje de programación, esquemas de bases de datos y componentes reutilizables. (7)

El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que es un lenguaje, cuenta con reglas para combinar tales elementos.

En UML 2.0 existen 13 tipos diferentes de diagramas. Para comprenderlos de manera concreta, es útil categorizarlos jerárquicamente. Dicha jerarquía, podría ilustrarse de la siguiente manera:

**Los diagramas de estructura estática**: Describen las propiedades estructurales del sistema.

• Diagrama de clases: Conjunto de clases, interfaces y colaboraciones; así como sus relaciones.

• Diagrama de objetos: Conjunto de objetos y sus relaciones.

• Diagrama de Casos de Uso: Conjunto de casos de uso y actores y sus relaciones.

**Los diagramas de comportamiento**: Acentúan en lo que debe suceder en el sistema modelado.

• Diagramas de interacción (secuencia y colaboración): Objetos y sus relaciones, incluyendo los mensajes que pueden ser enviados entre ellos.
• Diagrama de estados: Muestra una máquina de estado que consta de estados, transiciones, eventos y actividades.

• Diagrama de actividad: Es un tipo especial de diagrama de estados que muestra el flujo de actividades dentro de un sistema.

**Los diagramas de implementación**:

• Diagrama de componentes: Organización y las dependencias entre un conjunto de componentes.

• Diagrama de despliegue: Configuración de nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos.

UML, es considerado un potente lenguaje de modelado que garantiza la especificación de los procesos en el desarrollo del software así como una excelente vía de comunicación y documentación.

### 1.4 Herramienta CASE

Las **herramientas CASE** (Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software, reduciendo el costo de las mismas en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores, entre otras.

Las herramientas CASE se pueden clasificar teniendo en cuenta los siguientes parámetros:

1) Las plataformas que soportan.
2) Las fases del ciclo de vida del desarrollo de sistemas que cubren.
3) La arquitectura de las aplicaciones que producen.
4) Su funcionalidad.

#### 1.4.1 Visual Paradigm

Visual Paradigm es una herramienta CASE profesional que soporta todo el ciclo de vida del software: análisis y diseño orientado a objetos, construcción, prueba y despliegue. Dicha

herramienta ayuda a una rápida construcción de la aplicación con alta calidad y a un menor costo. Permite modelar todos los tipos de diagrama de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta también proporciona abundantes tutoriales, demostraciones y proyectos UML. (8)

Como característica atractiva para los desarrolladores, presenta una interfaz de uso intuitiva y con muchas facilidades a la hora de modelar los diagramas que soportan la Ingeniería de Requerimientos.

La herramienta está diseñada para una amplia gama de usuarios, incluidos los ingenieros de software, analistas de sistemas, analistas de negocio y del sistema, o para cualquiera que esté interesado en la construcción de forma fiable a gran escala de sistemas de software, utilizando un enfoque orientado a objetos. Por estas ventajas es seleccionado Visual Paradigm como herramienta CASE para el Módulo de Estadística.

### 1.5 Plataforma de Desarrollo

**Java Platform, Enterprise Edition o Java EE** es una plataforma de programación que reduce el coste y la complejidad del desarrollo multi-capa, puede desplegarse rápidamente y mejorarse fácilmente según responda la empresa a las presiones de la competencia. La plataforma contiene un conjunto de herramientas que permiten crear un escenario ideal para el desarrollo y despliegue de aplicaciones Web escalables. Dicha plataforma incluye varias especificaciones de API, y define cómo coordinarlos. (9)

Se decidió seleccionar la plataforma Java 2 Platform Enterprise Edition (J2EE) con el fin de garantizar una solución informática multiplataforma, de alto rendimiento, escalabilidad y seguridad.

Con esta elección se avala que el SIIPOL sea una solución informática potente, robusta y eficiente.

### 1.6 Lenguajes de programación

Los lenguajes de programación son un conjunto de programas que controlan el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Están formados por un conjunto de símbolos y reglas

sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

### 1.6.1 Java

**Java** es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. (10) Unas de las principales características de este lenguaje son:

- **Simple**: Elimina muchas de las características desventajosas de otros lenguajes como C++, para mantener reducidas las especificaciones del lenguaje y añadir características muy útiles como el reciclado de memoria dinámica. El mismo reduce en un 50% los errores más comunes de programación con lenguajes.
- **Robusto**: Maneja la memoria para eliminar las preocupaciones por parte del programador de la liberación o corrupción de memoria. Estas características reducen drásticamente el tiempo de desarrollo de aplicaciones en Java. El mismo realiza todo esto sin necesidad de que el programador se lo indique.
- **Seguro**: Las aplicaciones de Java resultan extremadamente seguras, ya que no acceden a zonas delicadas de memoria o de sistema, con lo cual evitan la interacción de ciertos virus, es decir, la seguridad se integra en el momento de compilación, con el nivel de detalle y de privilegio que sea necesario.
- **Portable**: Más allá de la portabilidad básica por ser de arquitectura independiente, Java implementa otros estándares de portabilidad para facilitar el desarrollo. Un programa compilado de Java puede ser utilizado por cualquier computadora que tenga implementado el intérprete de Java.
- **Multiprocesador**: El beneficio de ser multiprocesador consiste en un mejor rendimiento interactivo y mejor comportamiento en tiempo real. Aunque el comportamiento en tiempo real está limitado a las capacidades del sistema operativo subyacente (Unix, Windows, etc.), aun supera a los entornos de flujo único de programa, tanto en facilidad de desarrollo como en rendimiento.

- **Dinámico**: Java, para evitar que los módulos de clases haya que estar transfiriéndolos de la red cada vez que se necesiten, implementa las opciones de persistencia, para que no se eliminen cuando se borre la memoria caché de la máquina.

Por los beneficios planteados anteriormente y por resaltar Java como lenguaje de programación adecuado para aplicaciones de gran envergadura, fue seleccionado para el desarrollo de SIIPOL y en particular el Módulo de Estadística.

### 1.7 Frameworks utilizados en la solución

Un **framework** es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, en base a la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

#### 1.7.1 Spring

**Spring** constituye el framework utilizado en el desarrollo de la capa de lógica de negocio del Módulo de Estadística. El mismo es declarado código abierto para el desarrollo de aplicaciones web sobre la plataforma Java. No obliga a usar un modelo de programación en particular, por lo que se ha popularizado en la comunidad de programadores en Java. Por su diseño ofrece mucha libertad a los desarrolladores y soluciones muy bien documentadas y fáciles de usar para las prácticas comunes en la industria. (11)

Dentro de las ventajas que ofrece *Spring*, se encuentran:

1. Facilita la manipulación de los objetos.
2. Reduce la proliferación de *Singletons.*

3. Elimina la necesidad de usar distintos y variados tipos de ficheros de configuración.
4. Mejora la práctica de programación.

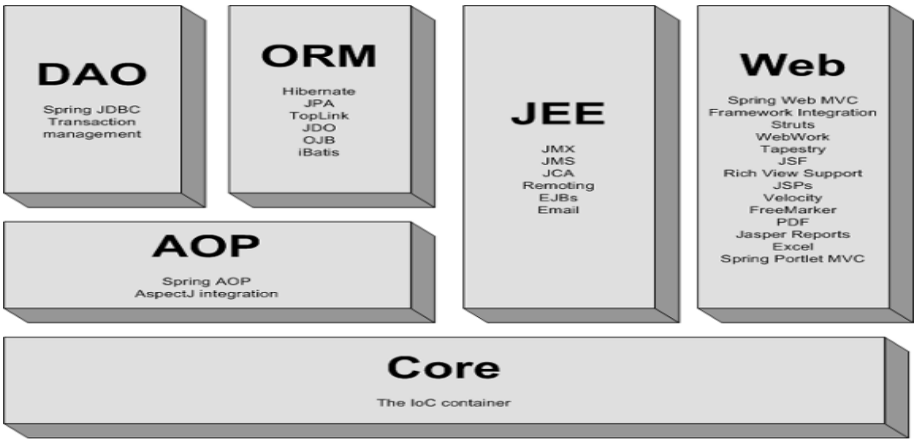



**Figura 3: Estructura de Spring**

### 1.7.2 Hibernate

**Hibernate:** solución ORM (Object-Relational Mapping) para Java, constituye el framework utilizado en el desarrollo de la capa de acceso a datos del Módulo de Estadística, se selecciona a partir de sus características y ventajas.

Este framework tiene como objetivo fundamental solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el orientado a objetos y el relacional. Para lograr esto le permite al desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Con esta información *Hibernate* hace posible a la aplicación manipular los datos de la base de datos operando sobre objetos, con todas las características de la programación orientada a objetos. *Hibernate* genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos. Está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. También tiene la funcionalidad de crear la base de datos a partir de la información disponible. Igualmente ofrece un lenguaje de consulta de datos llamado HQL (*Hibernate Query Language*), al mismo tiempo que una API para construir las consultas programáticamente conocida como "criteria". (12)

Hibernate además es Open Source y la licencia del producto está eximida de costo.

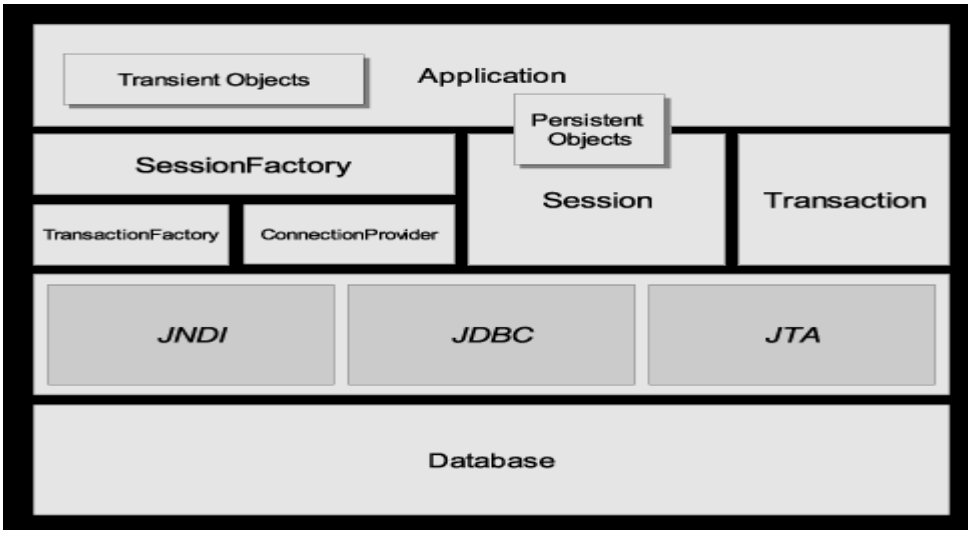



**Figura 4: Arquitectura de Hibernate**

### 1.7.3 JUnit

JUnit es un framework creado por Erich Gamma y Kent Beck que es utilizado por programadores para hacer pruebas unitarias de aplicaciones Java.

El framework permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera, es decir, en función de algún valor de entrada, se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente. (13)

Tiene como principal característica poder definir prueba de forma rápida y sencilla a través de la utilización de etiquetas. También es posible crear clases que sirvan para lanzar conjuntos de prueba y de esta manera se automatiza más el proceso. En ambiente de pruebas es una de las herramientas más usadas y conocidas. Como característica esencial para el desarrollo del módulo es que el IDE de Red Hat incluye al framework junit y al ser este código abierto, se podrá estudiar el código fuente por su cuenta. Por todas las ventajas planteadas es utilizado en las pruebas unitarias y de caja blanca, las cuales se abordarán en capítulos posteriores.

### 1.8 Entorno de Desarrollo Integrado

Obtenido J2EE como plataforma solo restaría determinar el Entorno de Desarrollo Integrado (IDE). El IDE es una aplicación que está compuesta por un conjunto de herramientas que son utilizadas por un programador. El mismo puede ser exclusivo para un lenguaje de programación o bien poder utilizarse para varios. Suele consistir de un editor de código, un compilador, un debugger [1] y un constructor de interfaz gráfica. Muchos IDE vienen en paquetes SDK, en particular se utiliza JDK.

#### 1.8.1 Red Hat Developer Studio

Red Hat ha incrementado su huella entre los desarrolladores con la versión beta a nivel mundial de Red Hat Developer Studio, el nuevo entorno de desarrollo integrado basado en Eclipse (IDE) que incluye JBoss Enterprise Middleware y Red Hat Enterprise Linux. Por primera vez, los desarrolladores disponen de herramientas de desarrollo basadas en Eclipse integradas y un entorno disponible completamente open source. La piedra angular del programa para desarrolladores de Red Hat Developer Studio, está diseñada para maximizar la productividad del desarrollador sobre las soluciones de Red Hat (14) Sus principales características son:

1. **Un modelo de programación unificado**: Developer Studio aumenta y proporciona nuevas herramientas alrededor de JBoss Seam para construir aplicaciones de manera sencilla y consistente. Hoy, el tipo de aplicación típicamente dicta el modelo de programación que el desarrollador utiliza, lo que significa que deben aprender a utilizar diferentes modelos. JBoss, que actualmente está siendo estandarizado como Web Beans en el Java Community Process; ofrece un modelo unificado y sencillo para desarrollar cualquier clase de aplicación, eliminando la necesidad de múltiples modelos de programación.
2. **Potentes capacidades Ajax**: Proporciona un entorno de desarrollo Ajax integrado y potente con JBoss Seam y JBoss Ajax4jsf, componentes Web JBoss RichFaces, y herramientas What You See Is What You Get (WYSIWYG) para crear interfaces y páginas web que soportan Ajax.
3. **Utilidades Java Platform 2 Enterprise Edition (J2EE)**: Developer Studio hace más sencilla la construcción de aplicaciones Java 2 EE, con capacidades como WYSIWYG y edición de Java Server Faces JSF y páginas Facelets, asistencia

[1] depurador

de código dinámico y una paleta de componentes. Además, al incluir e integrar JBoss Application Server, Developer Studio simplifica el despliegue, la ejecución y la depuración de las aplicaciones Java 2 EE.

4. **Un tiempo de ejecución integrado con las herramientas de desarrollo**: Red Hat Developer Studio es el primer entorno de desarrollo open source basado en Eclipse que une el runtime con las herramientas. Esto elimina la necesidad de que el desarrollador improvise estructuras y componentes open source antes de que empiece a desarrollar. (15)

### 1.9 Sistema Gestor de Base de Datos

Se utilizará como sistema gestor de base datos Oracle, para el desarrollo del proyecto SIIPOL, en particular del Módulo de Estadística, por el gran prestigio a nivel mundial. Oracle es considerado como uno de los sistemas de bases de datos más completos, destacando: soporte de transacciones, estabilidad, escalabilidad y soporte multiplataforma.

**Oracle** fue desarrollado por Oracle Corporation la cual es una de las mayores compañías de software del mundo. Sus productos van desde bases de datos hasta sistemas de gestión. Cuenta además, con herramientas propias de desarrollo para realizar potentes aplicaciones, como Oracle Designer.

Es una herramienta cliente/servidor para la gestión de bases de datos. Es un producto vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio hacen que sólo se vea en empresas muy grandes y multinacionales. Posee un lenguaje de diseño de bases de datos muy completo (PL/SQL) que permite implementar diseños "activos", con triggers [2]y procedimientos almacenados, con una integridad referencial declarativa muy fuerte. Permite el uso de particiones para la mejora de la eficiencia, de recopilación e incluso ciertas versiones admiten la administración de bases de datos distribuidas. El software del servidor puede ejecutarse en multitud de sistemas operativos.

### 1.10 Arquitectura Técnica

Como propuesta de solución se contará con el servidor web Apache Tomcat que en conjunto con la base datos Oracle, compartirán recursos para gestionar múltiples necesidades empresariales.

[2] disparador

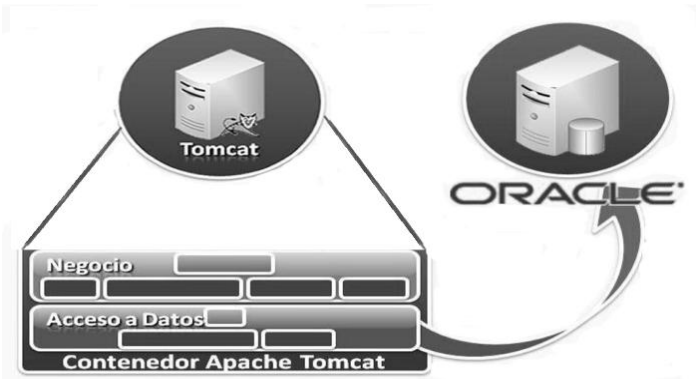



**Figura 5: Solución Propuesta.**

Se escogió el estilo arquitectónico en capas por sus resaltantes características en la solución, el cual proporciona ventajas para el desarrollo de la aplicación como son:

1. Facilita la migración. El acoplamiento con el entorno está localizado en las capas inferiores. Estas son las únicas a reimplementar en caso de transporte a un entorno diferente.

2. Cada nivel implementa interfaces claras y lógicas, lo que facilita la sustitución de una implementación por otra.

3. Permite trabajar en varios niveles de abstracción. Para implementar los niveles superiores no se necesita conocer el entorno subyacente, solo las interfaces que proporcionan los niveles inferiores.

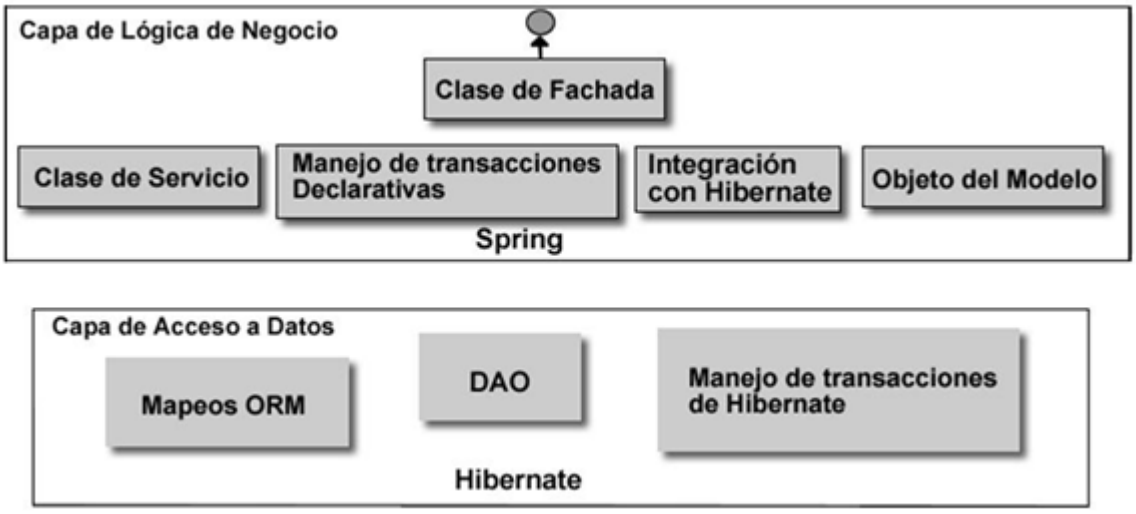



**Figura 6: Modelo N-Capa.**

Para el desarrollo de la solución se escogió el patrón de arquitectura Modelo Vista Controlador (MVC) el cual separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página.

1. **Modelo:** Representación específica del dominio de la información sobre la cual funciona la aplicación.
2. **Vista:** Presenta el modelo en un formato adecuado para interactuar, usualmente un elemento de interfaz de usuario.
3. **Controlador:** Responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

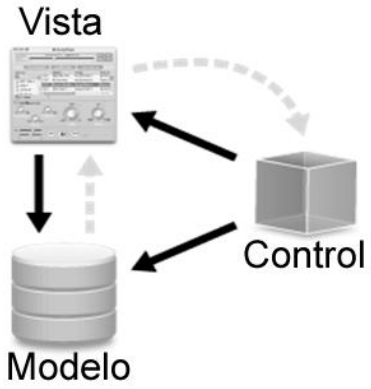



**Figura 7: Modelo Vista Controlador.**

## 1.11 Conclusiones

La utilización de Java como lenguaje de desarrollo, J2EE como plataforma y Visual Paradigm como herramienta CASE, permitirá el desarrollo claro y fluido de un sistema construido sobre bases sólidas y un entorno de desarrollo bien definido.

La utilización de la metodología seleccionada AUP, facilitará el seguimiento y monitoreo de un proceso de desarrollo que agilice el trabajo de los implicados en el mismo, de ahí la obtención de un producto con la calidad requerida.

Con la culminación de este capítulo se dio a conocer un conjunto de elementos que marcan el punto de partida del cual se iniciará la construcción de un software que cumpla con los objetivos propuestos.

.

# Capítulo2. Análisis, Diseño e Implementación de la Propuesta de Solución

En este capítulo se desarrollará la solución del Módulo de Estadística, a través de la realización de los artefactos UML necesarios para su compresión y definidos por cada flujo de trabajo de la metodología seleccionada.

Primeramente se realiza un análisis del Módulo de Estadística a partir de los resultados obtenidos por la ingeniería de requerimientos realizada por el equipo de analistas. Se mostrará el modelado e implementación del sistema a través del seguimiento de varios Casos de Uso considerados significativos y la generación de artefactos como modelo de datos, modelo de implementación, entre otros.

## 2.1 Análisis del Módulo de Estadística

El módulo de estadística brinda la posibilidad de crear reportes estadísticos en formatos preestablecidos. También se tendrá el control de las estadísticas relacionadas con delitos, actas procesales, notificaciones telefónicas, entre otras, brindando la posibilidad de generar diferentes tipos de estadísticas (**Anexo 1**), como:

- **Predefinidas**: Estadísticas claves que se realizan en la Institución.
- **Personalizadas:** Posibilidad de elaborar tipos de estadísticas según la Entidad.
- **Comparativas:** Comparaciones de períodos de tiempo.

Para adquirir mejor calidad en los reportes generados, el módulo ofrece la funcionalidad de personalizar el formato del reporte a generar tanto en PDF como Excel, así como seleccionar los tipos de cálculos y de gráficos (**Anexo 2**), que se corresponden con la estadística seleccionada. De esta manera ayudará en la toma de decisiones de la Institución así como en la obtención de forma automática de datos de un elevado interés en la investigación policial.

### 2.1.1 Modelo de Casos de Uso

El Módulo de Estadística cuenta con once casos de uso, se seleccionaron en su totalidad para la investigación por su complejidad y valor para el resto del sistema. A continuación se muestra el Modelo de Casos de Uso:

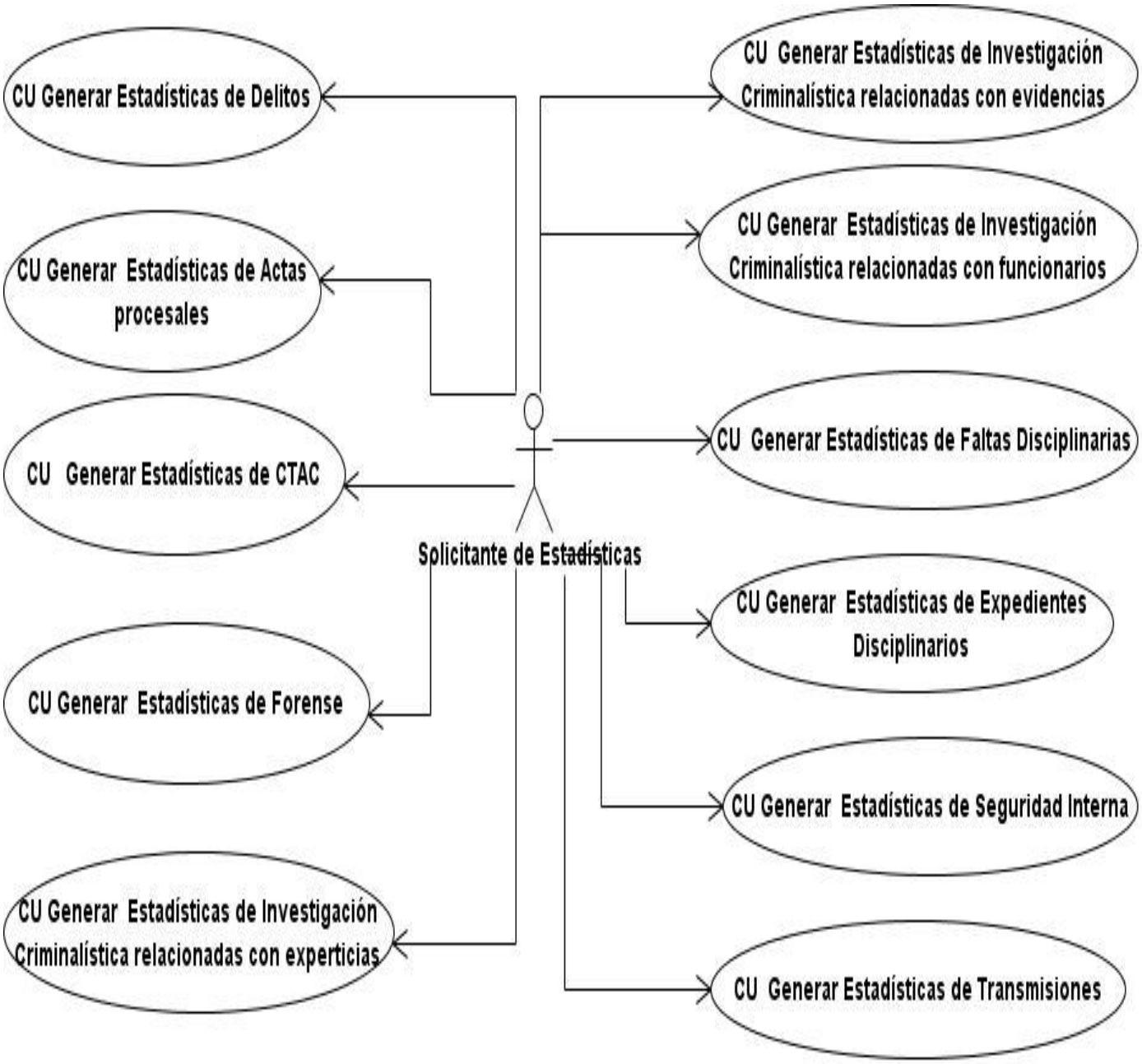



**Figura 8: Modelo de Casos de Uso. Módulo Estadística.**

### 2.1.2 Casos de Uso significativos

Se ha seleccionado el Caso de Uso (Generar Estadísticas de Delito) pues contiene la lógica necesaria para modelar e implementar el resto de los Casos de Uso pertenecientes al módulo Estadística.

| | |
|---|---|
| **Nombre del CU** | Generar Estadísticas de Delitos |
| **Actor** | Solicitante de Estadísticas |
| **Descripción** | El caso de uso inicia cuando el actor accede a la opción de generar estadísticas de Delitos. El sistema le muestra la interfaz con los tipos de estadísticas que se pueden generar según sus privilegios relacionados con los Delitos. El actor selecciona el tipo de estadística a generar y una fecha. El actor selecciona, además, en cuál de los formatos predeterminados desea visualizar el reporte y si desea generar gráficos selecciona el tipo de gráfico y la información que desea que contenga el mismo. El sistema genera los datos estadísticos solicitados terminando así el Caso de Uso. |
| **Referencia** | R.F: Generar Reporte de estadísticas de Delitos<br>R.F: Mostrar Reporte de Estadísticas de Delitos<br>R.F: Imprimir o Exportar a PDF.<br>R.F: Exportar a Excel.<br>R.F: Generar gráficos según datos referentes a Delitos.<br>R.F: Mostrar gráfico según datos referentes a Delitos. |

### 2.1.3 Requisitos no funcionales relevantes

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. (16)

Debido al estudio que se realizó de los Casos de Uso que componen el Módulo de Estadística, surgen un conjunto de requisitos no funcionales, de gran vitalidad para el desarrollo del sistema y en particular del módulo. A continuación se darán a conocer algunos de estos:

**RNF 1 Interfaz de Usuario.**

**Reporte:**

- Tipografía: Arial
- Tablas Cabezales:
  - Texto Cabezales: Arial 9 pts (+3) versión bold.
  - Color: (Depende del estilo). Blanco o Negro (para versiones de lujo con fondo amarillo y las versiones económicas).**(Anexo 3)**
  - Versión: bold.
- Contenido:
  - Texto Contenido: Arial 9 pts (+1), versión normal, interlineado.
  - Color: (Depende del Estilo). Negro. **(Anexo 3)**
  - Versión: normal, bold o cursiva para el caso de porcientos o totales al final del deporte.
- Imágenes:
  - Groso de Línea: 1 pto.
  - Color Línea: Negro 50%.
- Pie de Página:
  - Tamaño de letra: Arial 9 ptos.
  - Tamaño de resalte: Arial 9 ptos versión bold, color negro.
  - Texto de datos específico:

**RNF 2 Rendimiento.** (17)

- El sistema procesará una transacción en un tiempo no mayor a 1 segundo, a partir de que la petición se recibe en el servidor, es decir, esta cifra no incluye los retardos por concepto de tráfico de red.

- El sistema minimizará el volumen de datos en las peticiones y además optimizará el uso de recursos críticos como la memoria. Para ello se potenciará como regla guardar en la memoria caché datos y recursos de alta demanda.

- El sistema identificará por cada caso de uso aquellas operaciones que impliquen un elevado nivel de procesamiento en la base de datos y usará procedimientos almacenados para manejarlas.

**RNF 3 Fiabilidad.** (17)

- El sistema estará disponible durante 24 horas, los 7 días de la semana, los 365 días del año.

- El sistema hará un uso eficiente de la capacidad de almacenamiento de los servidores.

    - Una partición para los ficheros del Oracle (10 GB en el servidor).

    - Una partición para los Archive_log y Redolog (500 GB en el EVA).

**RNF 4 Seguridad.** (17)

- El sistema manejará la seguridad de acceso y administración de usuarios: otorgamiento de privilegios y roles, asignación de perfiles.

- El sistema solicitará los permisos de acceso después de determinado tiempo de inactividad por parte del usuario.

- El sistema manejará mecanismos de encriptación de determinados datos que sean transmitidos por la red y que así lo requieran.

## 2.2 Diseño del Módulo Estadística.

Es necesario resaltar que el Modelo, primer flujo de AUP, no se evade del análisis y diseño, ya que es esencial para el desarrollo del sistema. No obstante AUP para el análisis

plantea el uso de las tarjetas CRC (*Class, Responsibilities, Collaborators*), esta técnica es empleada para modelar el sistema a través de análisis guiados por la responsabilidad. Las clases se examinan, se filtran y se refinan en base a sus responsabilidades con respecto al sistema.

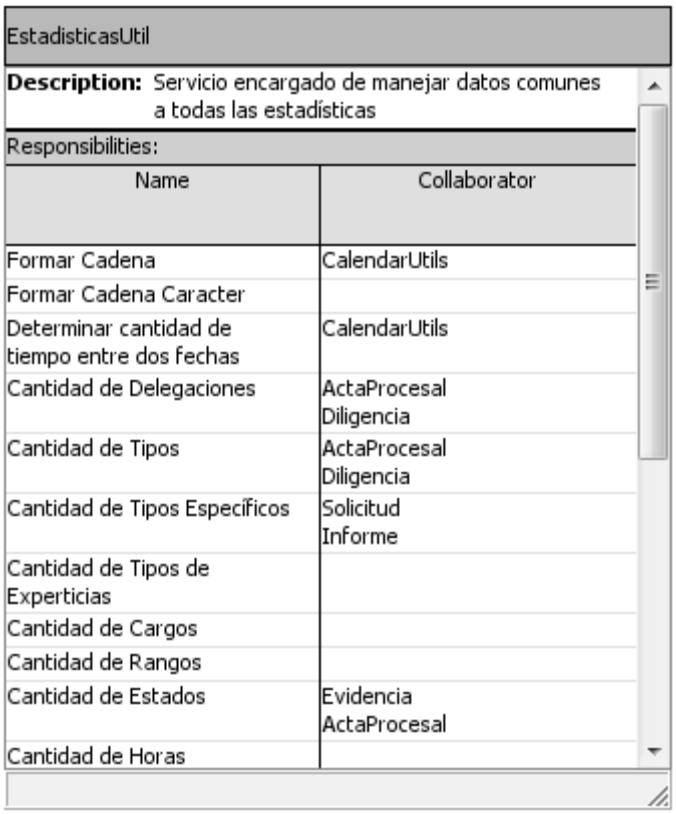
Nótese que las CRC sí son importantes a este nivel de detalles, ya que al no realizarse diagramas de secuencia debido a la complejidad, lo grande y la cantidad de lógica que englobarían, se opta por realizar CRC donde se muestra de forma más concisa y representativa por cada método de una clase. (18)

La utilización de la técnica CRC contribuyó en gran medida a la comunicación entre los miembros del equipo de desarrollo, modelar el sistema de forma tal que los desarrolladores hablen el mismo idioma con respecto al estilo de trabajo y prioridad de funcionalidades a tener en cuenta para la implementación, son cuestiones que sin lugar a dudas influyeron de manera determinante en la fluidez durante el proceso.

### 2.2.1 Tarjetas CRC.

Las tarjetas CRC representan las Clases de servicios con sus respectivas Responsabilidades y Colaboraciones, aquí solo se habla de las de Servicios teniendo en cuenta que realmente toda la lógica de negocio se debe encontrar en los Servicios. (18)

En el **Anexo 4** se muestra la jerarquía de clases, con sus responsabilidades y colaboración entre ellas. A continuación se observan las tarjetas más utilizadas y una descripción breve de algunas de las responsabilidades.



EstadisticasUtil

**Description:** Servicio encargado de manejar datos comunes a todas las estadísticas

Responsibilities:

| Name | Collaborator |
|---|---|
| Formar Cadena | CalendarUtils |
| Formar Cadena Caracter | |
| Determinar cantidad de tiempo entre dos fechas | CalendarUtils |
| Cantidad de Delegaciones | ActaProcesal<br>Diligencia |
| Cantidad de Tipos | ActaProcesal<br>Diligencia |
| Cantidad de Tipos Específicos | Solicitud<br>Informe |
| Cantidad de Tipos de Experticias | |
| Cantidad de Cargos | |
| Cantidad de Rangos | |
| Cantidad de Estados | Evidencia<br>ActaProcesal |
| Cantidad de Horas | |

**Figura 9: Tarjetas CRC <EstadísticaUtil>.**

**Tabla 1:Métodos de la clase EstadísticaUtil.**

| Responsabilidades | Descripción |
|---|---|
| Formar Cadena | Funcionalidad encargada de formar cadenas para realizar consultas a la base de datos. |
| Determinar cantidad de tiempo entre dos fechas | Funcionalidad encargada de obtener la diferencia de tiempo entre 2 fechas. |
| Cantidades | Funcionalidades encargadas de determinar el número de elementos a mostrar en gráfico de un reporte para calcular el área que debe ocupar dicho gráfico. |

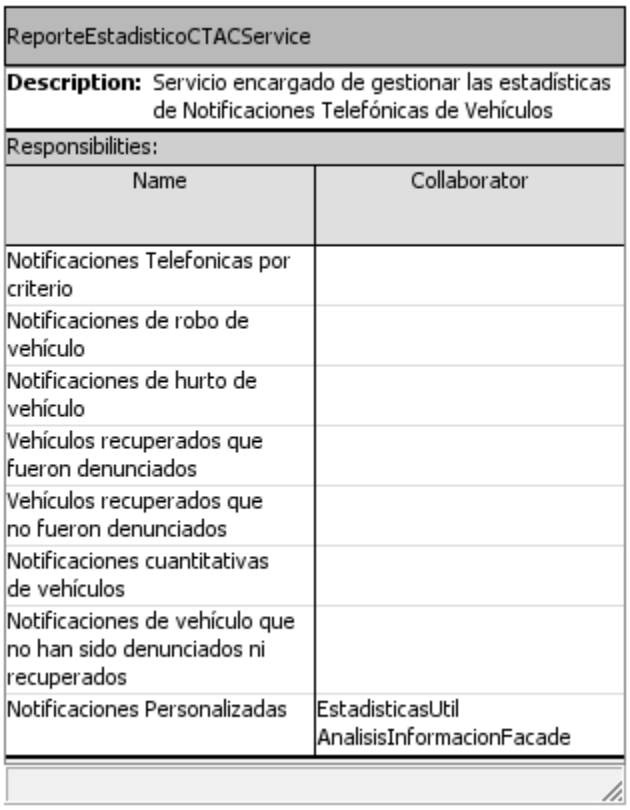
| ReporteEstadisticoDelitoService | |
|---|---|
| **Description:** Servicio encargado de gestionar las estadísticas de Delitos | |
| Responsibilities: | |
| Name | Collaborator |
| Principales delitos | Persona |
| Delitos totales | Persona |
| Delitos personalizados | Persona<br>EstadisticasUtil |
| Días de mayor índice delictivo | |
| Zonas de mayor índice delictivo | |
| Horas de mayor índice delictivo | |
| Delitos más frecuentes | |
| Homicidios | Persona |
| Robos de autos | Persona |
| Zonas de robo o hurto de vehiculos | |
| Zonas de homicidios | |
| Promedio de delitos diarios | |

**Figura 10: Tarjetas CRC <ReporteEstadisticoDelitoService>.**

**Tabla 2: Métodos de la clase ReporteEstadisticoDelitoService.**

| Responsabilidades | **Descripción** |
|---|---|
| Principales delitos | Funcionalidad encargada de generar por día, semana o mes, un reporte con la cantidad de los principales delitos ocurridos en dicho período en un rango de tiempo seleccionado por el usuario.<br><br>Los principales delitos están definidos por los clientes y entre ellos se encuentran robo, hurto, homicidio, entre otros. |
| Delitos totales | Funcionalidad encargada de generar por día, semana o mes, un reporte con la cantidad de delitos ocurridos en dicho período en un rango |

| | |
|---|---|
| | de tiempo seleccionado por el usuario. |
| Días de mayor índice delictivo | Funcionalidad encargada de generar los X días de mayor índice delictivo en un rango de tiempo seleccionado por el usuario, con el índice correspondiente a cada día.<br><br>La X es un número no mayor de 3 cifras seleccionado por el usuario que define la cantidad de días que se deben mostrar en el reporte. |



| ReporteEstadisticoCTACService | |
|---|---|
| **Description:** Servicio encargado de gestionar las estadísticas de Notificaciones Telefónicas de Vehículos | |
| Responsibilities: | |
| Name | Collaborator |
| Notificaciones Telefonicas por criterio | |
| Notificaciones de robo de vehículo | |
| Notificaciones de hurto de vehículo | |
| Vehículos recuperados que fueron denunciados | |
| Vehículos recuperados que no fueron denunciados | |
| Notificaciones cuantitativas de vehículos | |
| Notificaciones de vehículo que no han sido denunciados ni recuperados | |
| Notificaciones Personalizadas | EstadisticasUtil<br>AnalisisInformacionFacade |

**Figura 11: Tarjetas CRC <ReporteEstadisticoCTACService>.**

**Tabla 3: Método de la clase ReporteEstadisticoCTACService**

| Responsabilidades | Descripción |
|---|---|
| Notificaciones de robo de vehículo | Funcionalidad encargada de generar por día, semana o mes, un reporte con la cantidad de |

| | |
|---|---|
| | notificaciones telefónicas realizadas por robo de vehículo en dicho período en un rango de tiempo seleccionado por el usuario. |
| Notificaciones de hurto de vehículo | Funcionalidad encargada de generar por día, semana o mes, un reporte con la cantidad de notificaciones telefónicas realizadas por hurto de vehículo en dicho período en un rango de tiempo seleccionado por el usuario. |

### 2.2.2 Diagrama de paquetes

El Módulo de Estadística se encuentra estrechamente relacionado con varios componentes del sistema SIIPOL, de esta forma mejora la gestión de los servicios, ayudando al funcionamiento del mismo. Dentro de los subsistemas de los que depende el módulo se encuentran: Análisis de Información, Registro y Control, Administración, Investigación Penal, Investigación Interna, Investigación Criminalística e Investigación Forense.

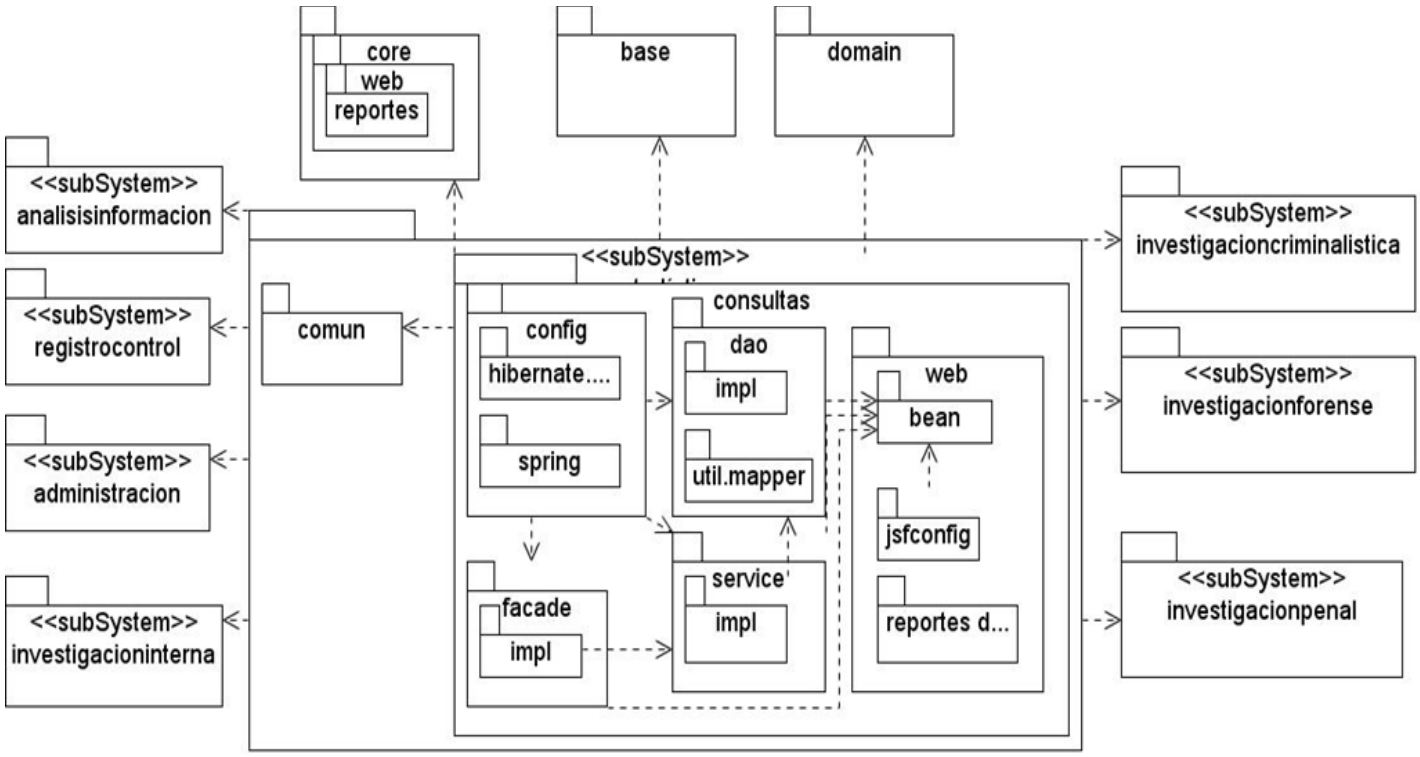



**Figura 12: Diagrama de Paquetes.**

### 2.2.3 Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. (19)

Con el plan de lograr la sencillez y eficacia en el sistema, luego de abordar las ventajas que proporcionan los mismos, se seleccionan varios patrones como son:

*Patrones de Diseño en la Capa de Negocio.*

- **Facade:** Es un patrón de diseño que sirve para proveer de una interfaz unificada sencilla que haga de intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas. Este implementa métodos convenientes para tareas comunes y a su vez hace que el código sea legible disminuyendo la dependencia del código externo en los trabajos internos de una biblioteca o submódulo. En el proyecto se evidencia el uso de este patrón, pues para cada submódulo se crea una clase fachada, cuya funcionalidad es ser el punto de acceso a los servicios del mismo.
  **(Anexo 5)**
- **Service Locator:** El patrón de localización de servicio es un patrón de diseño utilizado en el desarrollo de software para encapsular los procesos implicados en la obtención de un servicio con una capa de abstracción. El mismo es transparente para los desarrolladores del módulo, ya que utilizando el framework Spring, se hace uso del patrón, pues internamente reduce la complejidad de realizar el proceso de búsqueda y creación, los cuales consumen muchos recursos.
- **Interface:** Es un patrón fundamental de tipo estructural. Mantiene una clase (la interfaz) que usa datos y servicios provistos por otras clases independientes. Esta clase provee a sus clases herederas acceso uniforme a métodos y atributos

específicos, sin que deban saber a cuál clase específica pertenecen. En el módulo está presente el uso del patrón, pues para cada nivel (DAO, SERVICE, FACADE) se implementa una interfaz donde se define un contrato con las clases que implementen dichas interfaces. **(Anexo 6)**

*Patrones de Diseño en la Capa de Acceso a Datos.*

- **Data Access Object (DAO, Objeto de Acceso a Datos):** Es un componente de software que suministra una interfaz común entre la aplicación y dispositivos de almacenamiento de datos, tales como una base de datos o un archivo. El DAO es utilizado para crear la capa de persistencia que separa la lógica del negocio y a la misma vez interactúa con la base de datos. Este patrón facilita el trabajo de emigrar de un motor base de datos a otros.

### 2.2.4 Realización de los Casos de Uso

Los diagramas de interacción se emplean para representar gráficamente cómo los objetos interactúan a través de mensajes para realizar las tareas, dando lugar a los diagramas de colaboración que simbolizan las interacciones entre las instancias del modelo (objetos). Estos artefactos contribuyen a la documentación del proyecto, en particular al Módulo de Estadística, que es vital para el entendimiento de los procesos por parte del cliente.

#### 2.2.4.1 Diagrama de Clase del Análisis

A continuación se muestra una representación del diagrama de clases del análisis correspondiente a los Casos de Uso (Generar Estadísticas de Delitos y Generar Estadísticas de Actas Procesales).

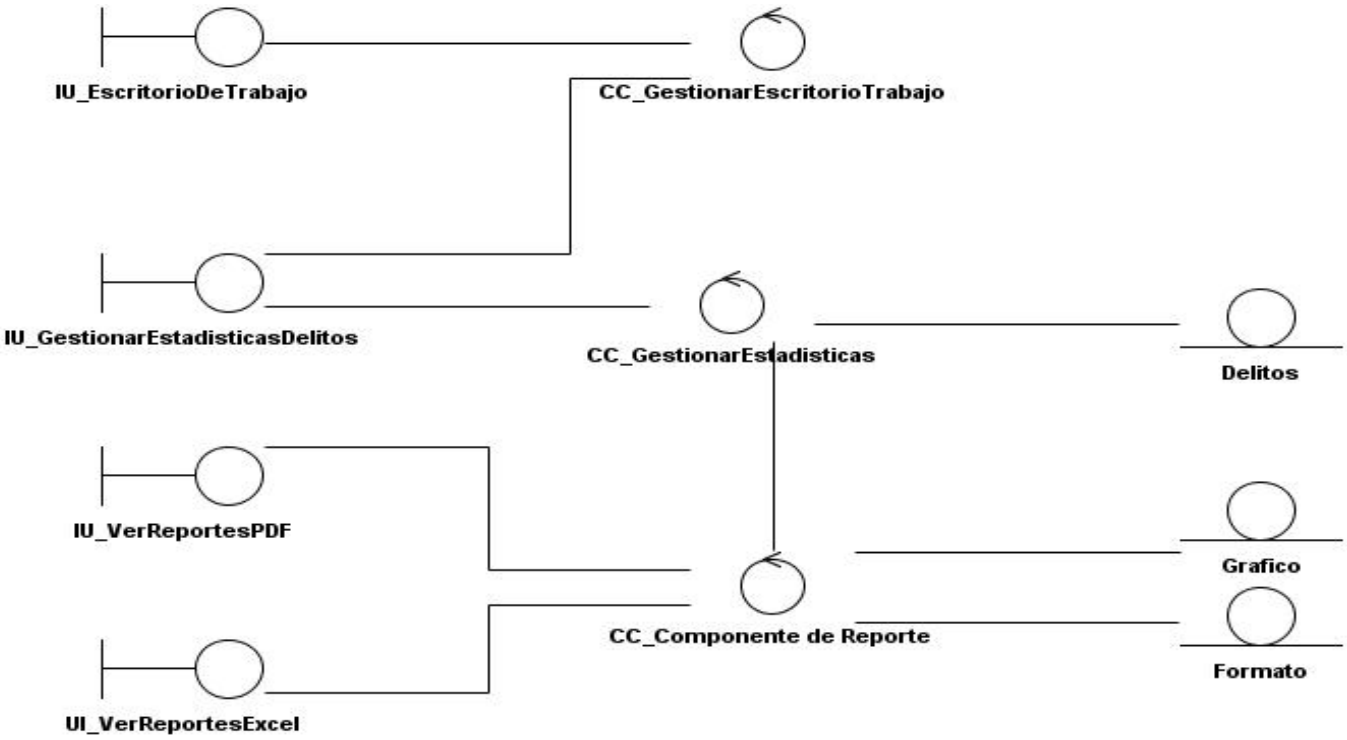



**Figura 13: CU Generar Estadísticas de Delitos.**

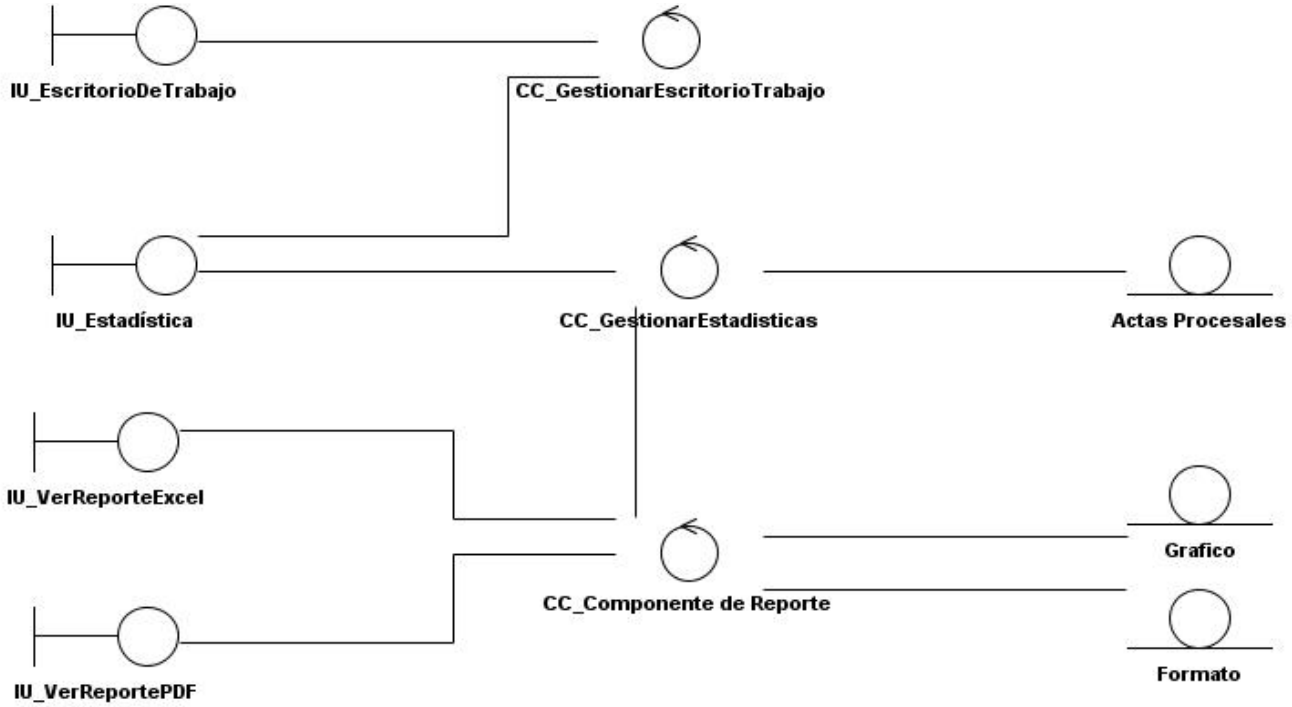



**Figura 14: CU Generar Estadísticas de Actas Procesales.**

### 2.2.4.2 Diagrama de Colaboración

El diagrama de colaboración del análisis se le realizará solamente a los Casos de Uso arquitectónicamente significativos, ya que en estos se deben recoger todos los comportamientos asociados a los demás Casos de Uso, por comportarse la mayoría de la misma forma. (18)

Posteriormente se muestra una representación correspondiente al caso de uso Generar Estadísticas de Delitos incluyendo los flujos más importantes.

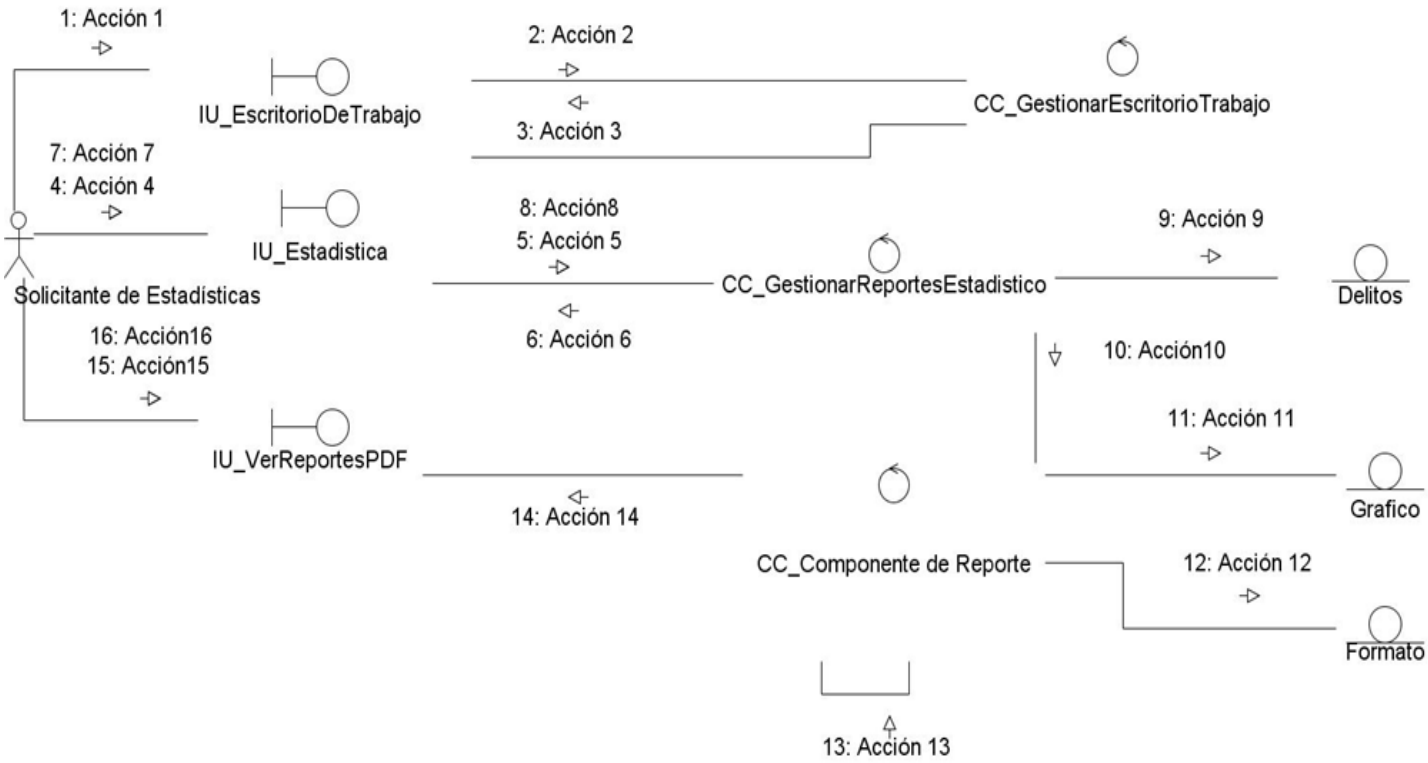



**Figura 15: CU Generar Estadísticas de Delitos<Estadística Predefinida>.**

Leyenda

- Acción 1-Generar Estadísticas de Delitos
- Acción 2-Mostrar(IUGeneralEstadísticaDelitos)
- Acción 3-Mostrar(IUGenerarEstadísticasDelitos_Predefinidas)
- Acción 4-Seleccionar tipo de Estadística
- Acción 5-seleccionarEstadística(tipoEstadística)
- Acción 6-Mostrar(datosComplementarios, cálculos y gráficos)

- Acción 7-Generar Estadística Predefinida en PDF(rango de fechas, tipo de Estadística, datos adicionales)
- Acción 8-generarEstadísticaPredefinida(fechainicio, fechafin, criterio)
- Acción 9-obtenerListadoDelitos(fechainicio, fechafin, criterio)
- Acción 10-GenerarPDF(listaDelitos, gráfico, formato)
- Acción 11-ConformarGráfico(gráfico)
- Acción 12-ConformaFormato(formato)
- Acción 13-GenerarReporte()
- Acción 14-MostarPDF(pdf)
- Acción 15-GuardarPDF
- Mensaje 16-CerrarPDF

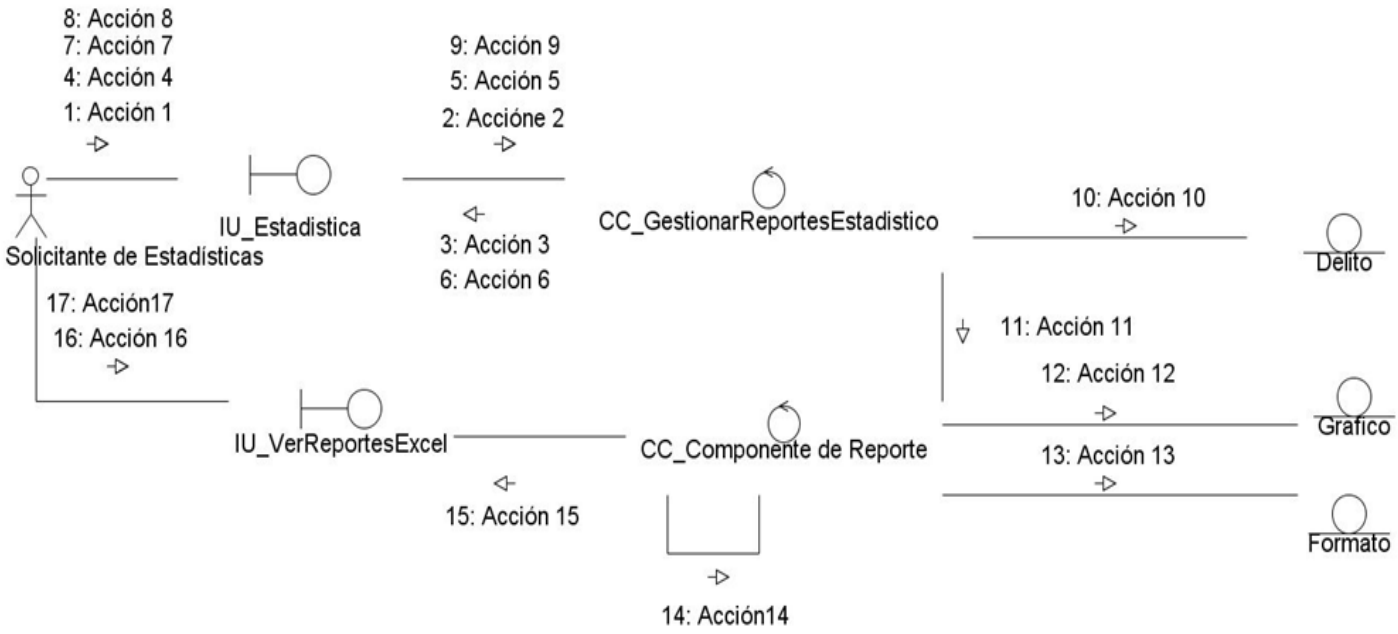



**Figura 16: CU Generar Estadísticas de Delitos<Estadística Personalizada>.**

Leyenda

- Acción 1-Seleccionar estadísticas personalizadas
- Acción 2-EstadísticaPersonalizada()
- Acción 3-Mostrar(variables a consultar, valor de la variable, estados, municipios, parroquias)
- Acción 4-Seleccionar variable estadística

- Acción 5-listarValoresVariables(variable)
- Acción 6-Mostrar(listado de valores)
- Acción 7-Formular consulta(variable, valores)
- Acción 8-Generar Estadística Personalizada en Excel(rango de fechas, consultas, datos adicionales)
- Acción 9-generarEstadísticaPersonalizada(fechainicio, fechafin, consultas, datos adicionales)
- Acción 10-obtenerActasProcesales(fechainicio, fechafin, consulta)
- Acción 11-GenerarExcel(listaDelitos, gráfico, formato)
- Acción 12-ConformarGráfico(gráfico)
- Acción 13-ConformaFormato(formato)
- Acción 14-GenerarReportes()
- Acción 15-MostarExcel(xls)
- Acción 16-GuardarExcel
- Acción 17-CerrarExcel

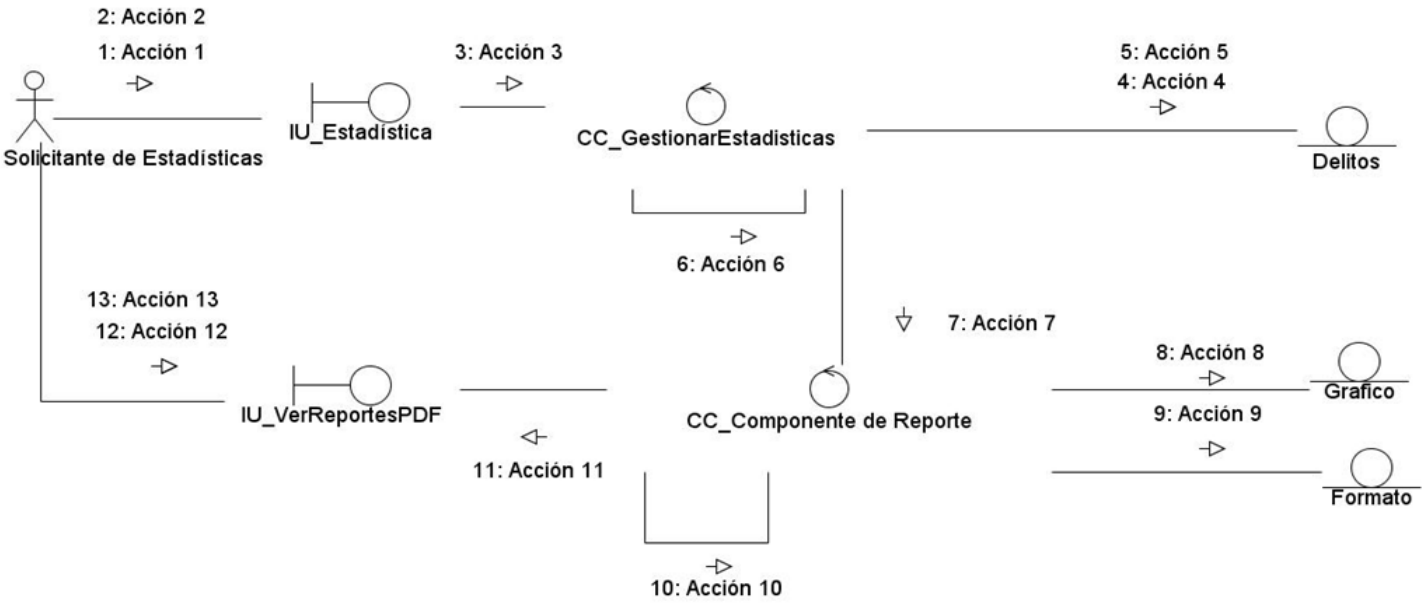



**Figura 17: CU Generar Estadísticas de Delitos< Estadística Comparativa>.**

Leyenda:

- Acción 1-Seleccionar estadísticas comparativas

- Acción 2-Generar Estadística Comparativa en PDF(rango de fechas 1, rango de fechas 2, datos adicionales)
- Acción 3-EstadísticaComparativa(rango de fechas 1, rango de fechas 2, datos adicionales)
- Acción 4-listado1:=obtenerListadoDelitos(fechainicio, fechafin, criterio)
- Acción 5-listado2:= obtenerListadoDelitos(fechainicio2, fechafin2, criterio)
- Acción 6-datos:=DameCantidadesTiempo(listado1, listado2)
- Acción 7-GenerarPDF(datos, gráfico, formato)
- Acción 8-ConformaGráfico(gráfico)
- Acción 9-ConformaFormato(formato)
- Acción 10-GenerarReportes()
- Acción 11-MostarPDF(pdf)
- Acción 12-GuardarPDF
- Acción 13-CerrarPDF

#### 2.2.4.3 Diagrama de contrato entre paquetes

Los diagramas de contrato entre paquetes constituyen un artefacto generado, a partir de necesidades propias del proyecto, como documentación al cliente que facilita un mayor entendimiento de los procesos una vez se haga la transferencia tecnológica. Los diagramas de contrato entre paquetes representan la interacción entre subsistemas, constituyen diagramas de secuencia que ilustran las interacciones entre paquetes, abstrayéndose de los detalles internos de cada capa (20).

Los contratos entre paquetes, se pueden realizar tanto por Casos de Uso, de necesitarse, o a nivel de submódulo si se da el caso de que no existe ninguna diferencia observable entre ellos.

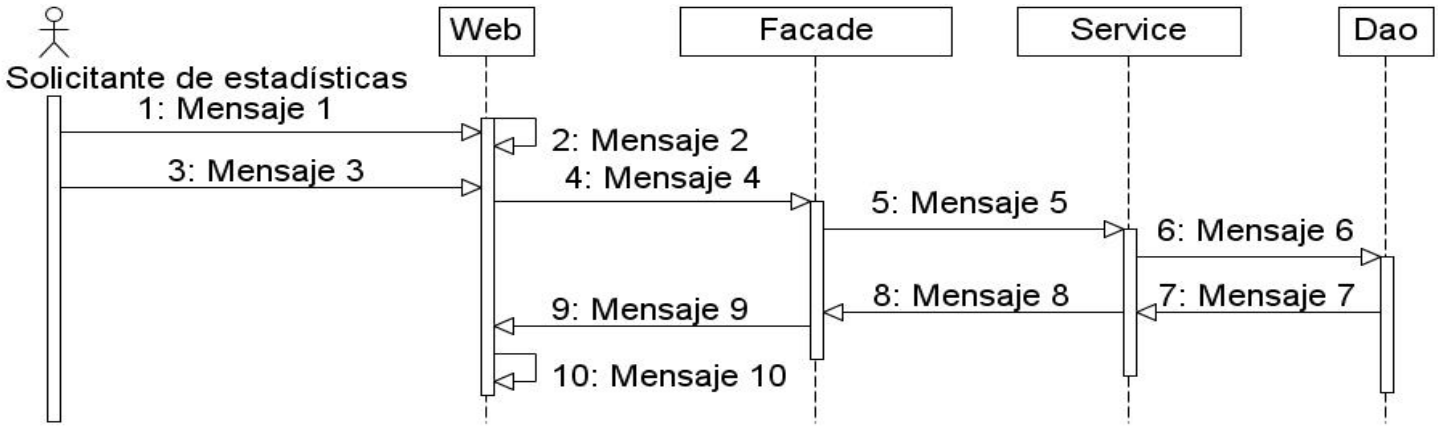



**Figura 18: CU Generar Estadísticas de Delito<Estadística Predefinida>.**

Leyenda

- Mensaje1:Seleciona Estadística Predefinidas
- Mensaje 2: Mostrar(datosComplementarios, cálculos y gráficos)
- Mensaje 3: Generar Estadística Predefinida en PDF
- Mensaje 4: generarEstadísticaPredefinida(fechainicio, fechafin, criterio)
- Mensaje 5: obtenerListadoDelitos(fechainicio, fechafin, criterio)
- Mensaje 6: obtenerListadoDelitos(fechainicio, fechafin, criterio)
- Mensaje 7: return listadoDelitos
- Mensaje 8: return listadoDelitos
- Mensaje 9: return listadoDelitos
- Mensaje 10: GenerarPDF(listaDelitos, gráfico, formato)

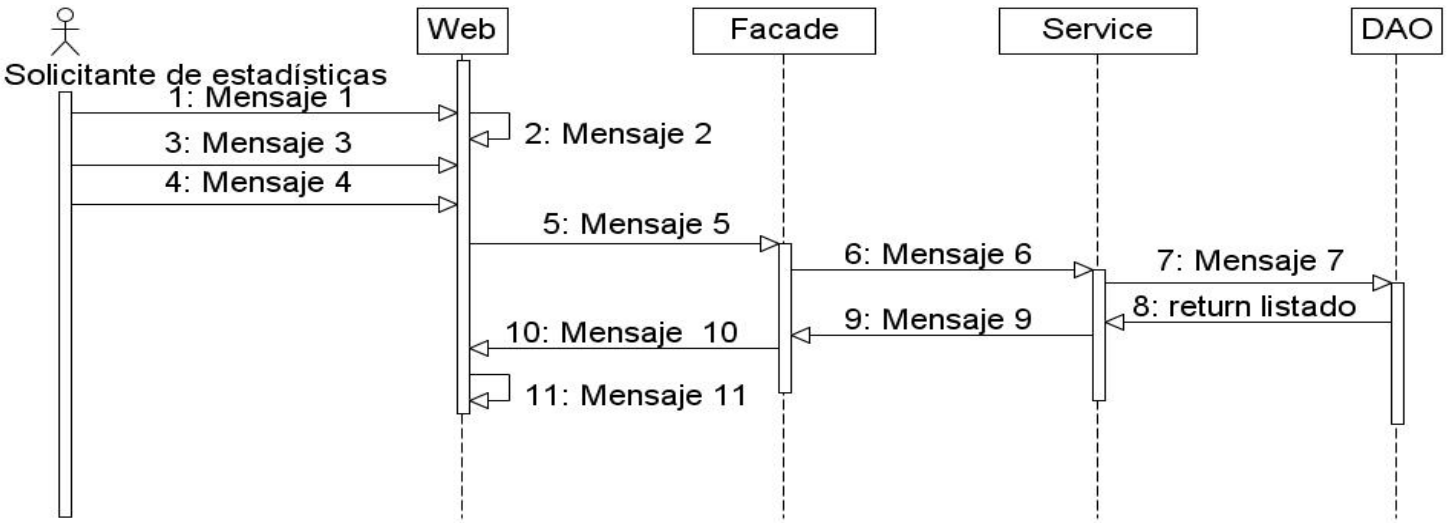



**Figura 19: CU Generar Estadísticas de Delitos<Estadística Personalizada>.**

Leyenda

- Mensaje 1-Selecciona Estadística Personalizada
- Mensaje 2-Mostrar(variables a consultar, valor de la variable, estados)
- Mensaje 3-Formular consulta(variable, valores)
- Mensaje 4-Generar Estadística Personalizada en Excel
- Mensaje5-generarEstadísticaPersonalizada(fechainicio,fechafin,consultas,datos adicionales)
- Mensaje 6-obtenerListadoActasProcesales(fechainicio, fechafin, consulta)
- Mensaje 7-obtenerListadoActasProcesales(fechainicio, fechafin, consulta)
- Mensaje 8-return listado
- Mensaje 9-return listado
- Mensaje 10-return listado
- Mensaje 11-GenerarExcel(listado, gráfico, formato)

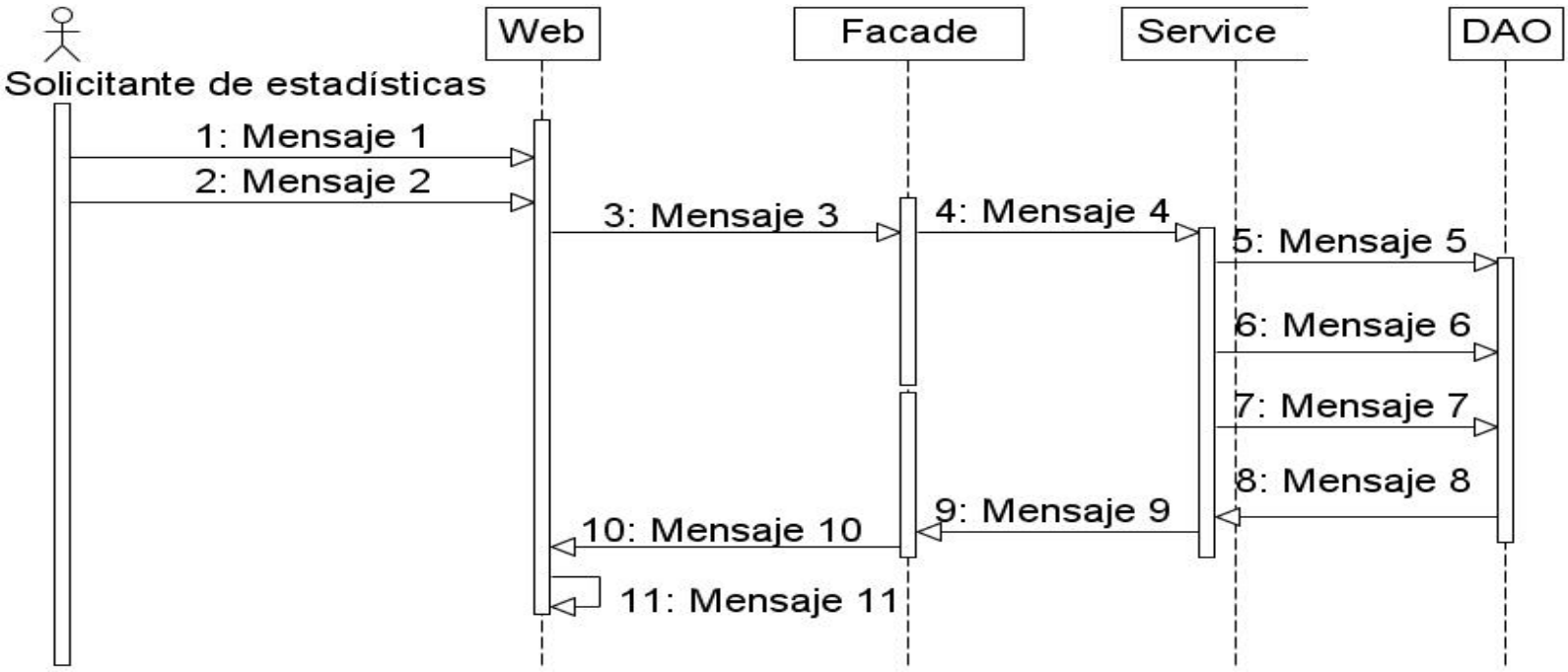



**Figura 20: CU Generar Estadísticas de Delitos<Estadística Comparativas>.**

Leyenda

- Mensaje 1-Selecciona Estadística Comparativa
- Mensaje 2- Generar Estadística Comparativa en PDF(rango de fechas 1, rango de fechas 2, datos adicionales )
- Mensaje 3-EstadísticaComparativa(rango de fechas 1, rango de fechas 2, datos adicionales)
- Mensaje 4-EstadísticaComparativa(rango de fechas 1, rango de fechas 2, datos adicionales)
- Mensaje 5-listadoDelitos1:=obtenerListadoDelitos(fechainicio, fechafin, criterio)
- Mensaje 6-listadoDelitos2:=obtenerListadoDelitos(fechainicio2, fechafin2, criterio)
- Mensaje 7-datos:=DameCantidadesTiempo(listadoDelitos1, listadoDelitos2)
- Mensaje 8-return datos
- Mensaje 9-return datos
- Mensaje 10-return datos
- Mensaje 11-GenerarPDF(datos, gráfico, formato)

## 2.3 Modelo de Datos

El modelo de datos describe la representación lógica y física de la información persistente manejada por el sistema. Estudia los datos independientemente del procesamiento que los transforma. Se compone de tres piezas de información interrelacionadas: el objeto de datos, los atributos que describen el objeto de datos y la relación que conecta objetos de datos entre sí.

### 2.3.1 Diagrama de Clases Persistentes

Un diagrama de clases persistente incluirá solamente las clases del dominio del subsistema y su interacción directa con las demás clases del dominio de otros subsistemas. (18)

Una de las características fundamentales del Módulo de Estadística es que no posee clases persistentes, al contrario utiliza clases de dominio de otros módulos. En el siguiente diagrama de clases persistentes se establecen relaciones de herencia y asociaciones significativas para la generación de los distintos informes estadísticos.

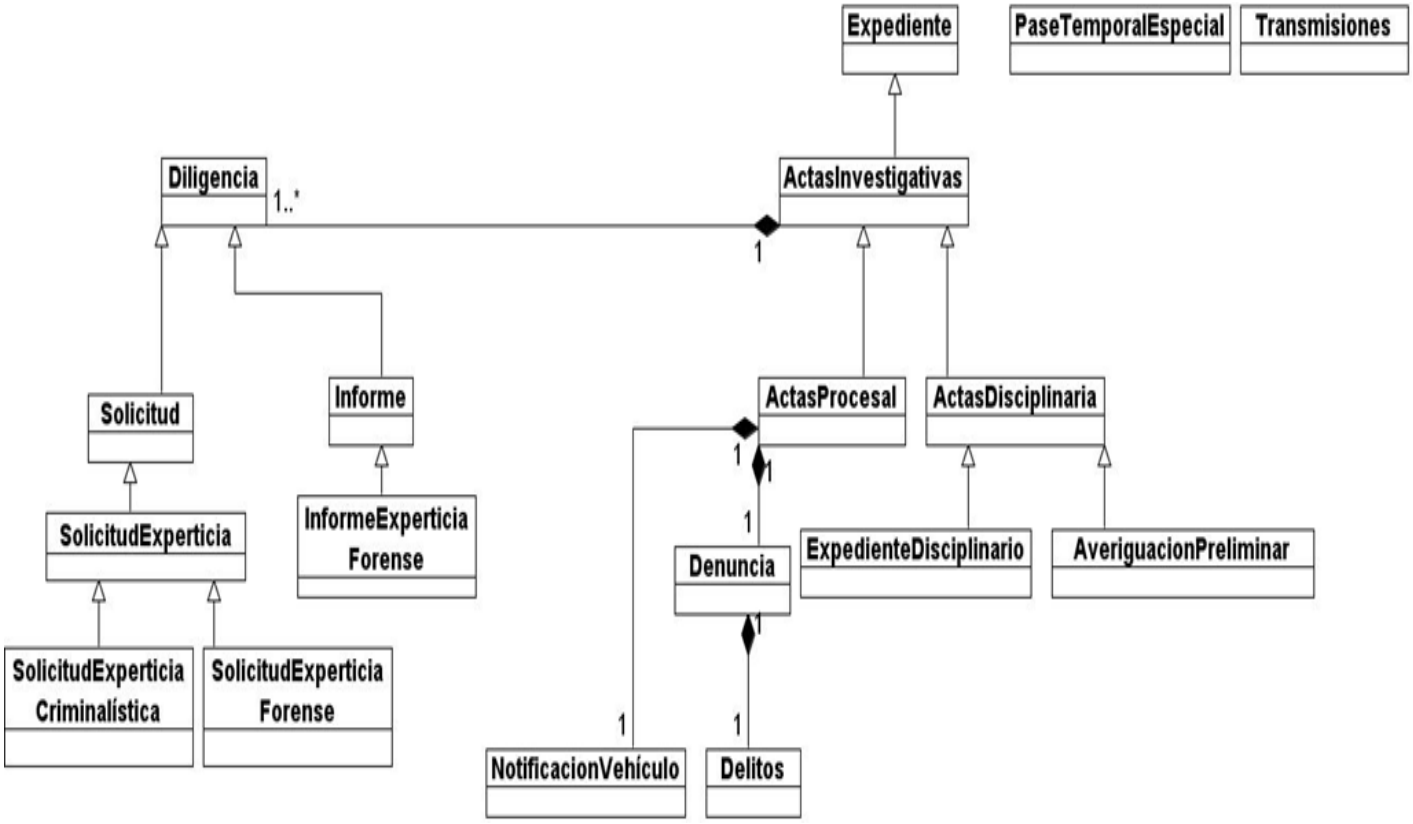



**Figura 21: Diagrama de Clases Persistentes.**

### 2.3.2 Diagrama de Entidad Relación

La lógica principal del módulo está basada en los procedimientos almacenados, ya que se utilizan varios elementos del modelo de datos para dar respuesta a una misma funcionalidad. A continuación se muestra la estructura física de las clases de dominio utilizada por el módulo.

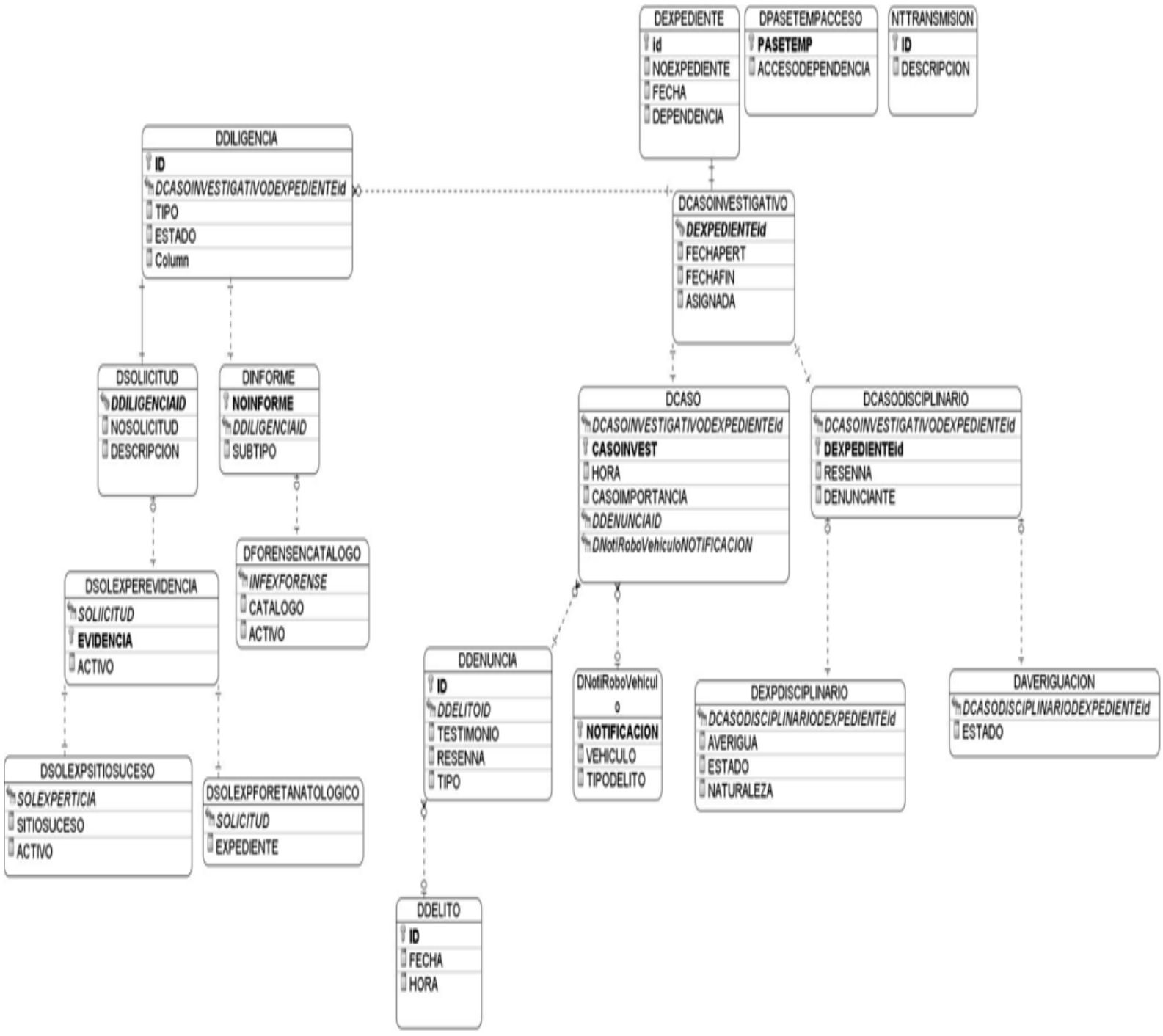



**Figura 22: Diagrama de Entidad Relación**

## 2.4 Modelo de Implementación

La etapa de implementación surge con el resultado del diseño e implementado el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares. El objetivo principal de esta etapa es desarrollar la arquitectura y el sistema como un todo, de la forma más específica. Teniendo en cuenta los propósitos de la implementación:

- Definir la organización del código.
- Planificar las integraciones del sistema necesario en cada iteración.
- Implementar las clases y subsistemas encontrados durante el diseño.

El modelo de Implementación es la unión de varios artefactos que se generan de gran valor, pues denota la implementación actual del sistema en términos de componentes y subsistemas de implementación, es natural mantener el mismo a lo largo de todo el ciclo de vida del software.

### 2.4.1 Diagramas de subsistemas de implementación

Los subsistemas de implementación están muy relacionados con los subsistemas de diseño en el modelo de diseño. De hecho, los subsistemas de implementación deberían seguir la traza uno a uno de sus subsistemas de diseño correspondientes. Proporcionan una forma de organizar los artefactos del modelo de implementación en trozos más manejables. Un subsistema puede estar formado por componentes, interfaces y otros subsistemas. Además, un subsistema puede implementar las interfaces que representan la funcionalidad que exportan en forma de operaciones.

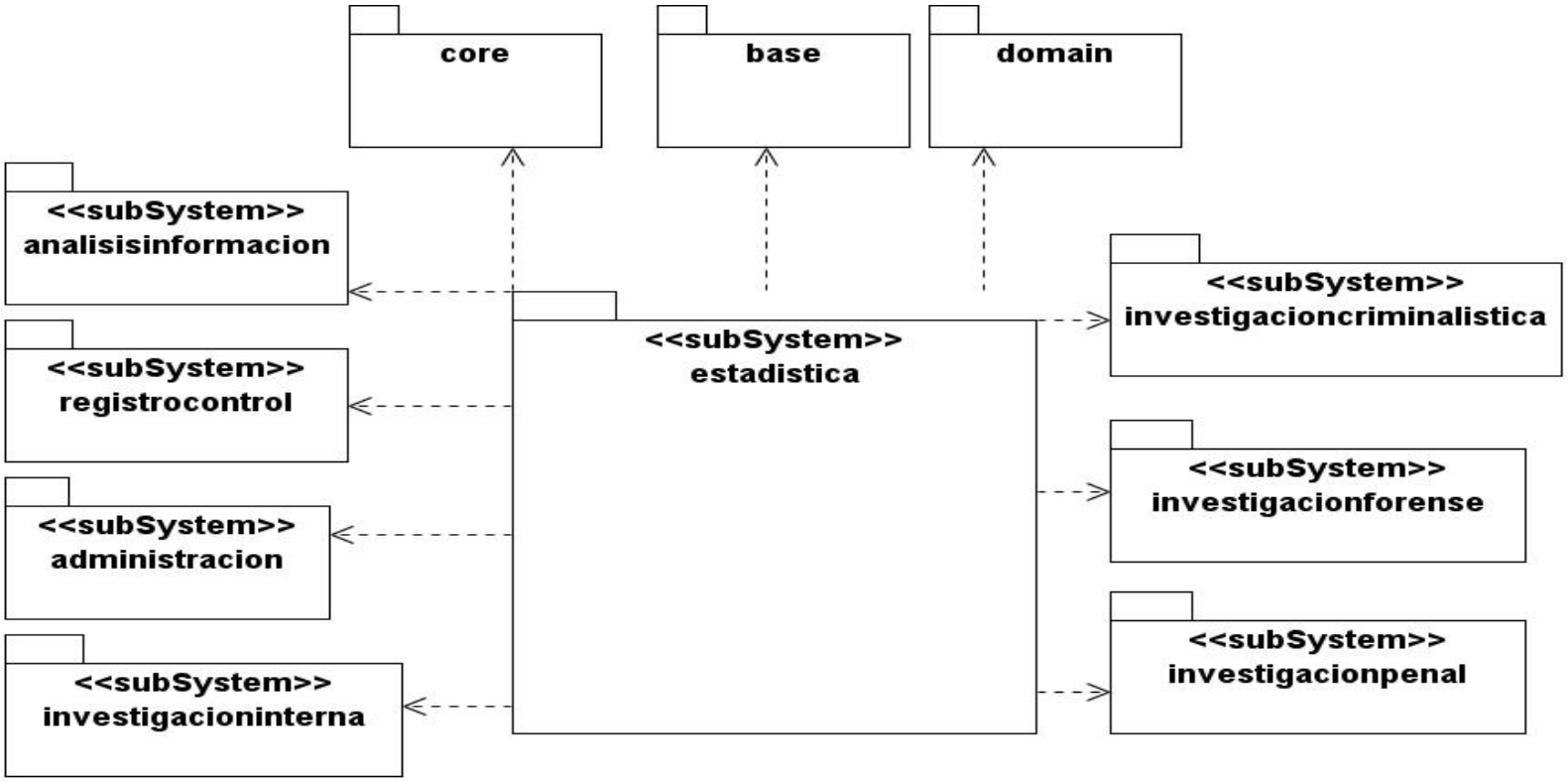



**Figura 23: Diagrama de subsistemas de Implementación**

### 2.4.2 Diagrama de componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Característica de estos diagramas con respecto a otros tipos de diagramas, es sin duda su contenido. Normalmente, contienen componentes, interfaces y relaciones entre ellos. Los componentes pertenecen a un mundo físico, es decir, representan a un bloque de construcción al modelar aspectos físicos de un sistema. Cada uno debe tener un nombre que lo distinga de los demás.

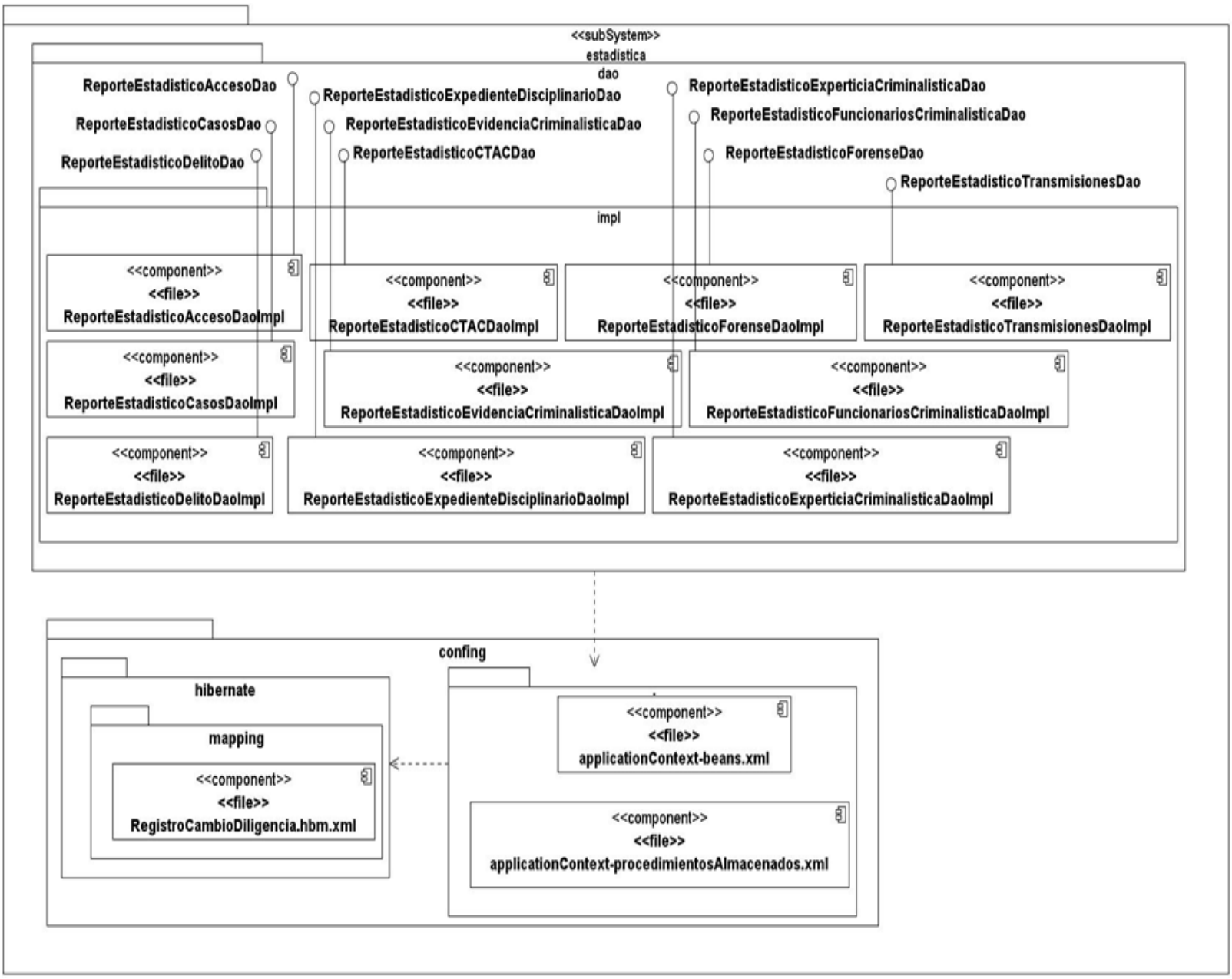



**Figura 24: Diagrama de Componentes. Vista Acceso a Datos**

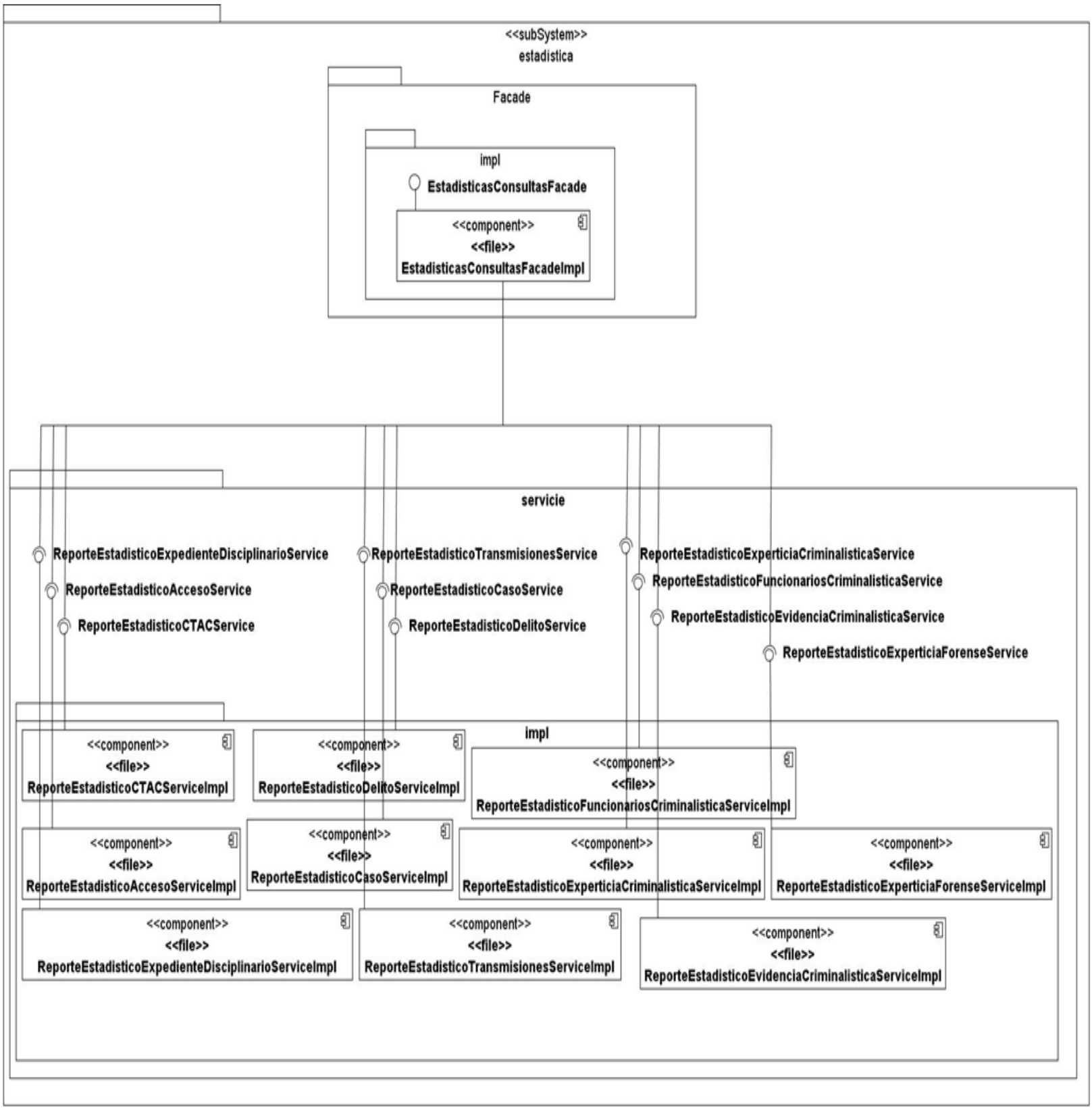



**Figura 25: Diagrama de Componentes. Vista Lógica de Negocio**

### 2.4 Estándar de codificación

La estandarización del código es una actividad de carácter vital para el desarrollo de la solución propuesta, pues es una decisión centralizada del equipo de Arquitectura donde se plasma el estilo de código a utilizar, el mismo es dependiente del lenguaje en que se desee desarrollar la aplicación. Para tener constancia de las pautas del equipo de Arquitectura, se confeccionó un documento nombrado Guía de Estilo de Código para el proyecto CICPC, el cual refleja las características primordiales del estilo de código vigente en el proyecto, siguiendo las Convenciones de Código Java.

## 2.5 Conclusiones

La utilización de los patrones de diseño como el FACADE y DAO, ayudó al desarrollo de la solución propuesta, formalizó un vocabulario común entre el equipo de desarrollo y proporcionó elementos reusables en el diseño del sistema de software.

A través de la realización del modelo de diseño, utilizando las tarjetas CRC, se logró agilizar el proceso de desarrollo, generando solo la documentación necesaria que sirviera de guía para la construcción del software.

Se desarrollaron varios artefactos como: modelo de diseño, modelo de datos y modelo de implementación, quedando la aplicación lista para entrar al flujo de pruebas.

Con el desarrollo de la propuesta de solución se garantizó la automatización de los reportes generados por la Institución.

# Capítulo 3. Validación de la solución propuesta

## 3.1 Introducción

Este capítulo resumirá el conjunto de pruebas realizadas al Módulo Estadística para comprobar la satisfacción de los requisitos funcionales y no funcionales asociados al mismo. Se abordarán los tipos de pruebas realizadas, resultados obtenidos en las mismas y la evaluación de esos resultados.

## 3.2 Métodos de prueba. Pruebas Aplicadas al Sistema.

El Módulo Estadística fue sometido a diversas pruebas en cada una de las iteraciones por las que pasó. Las que permitieron verificar y revelar la calidad del producto y son utilizadas para identificar posibles fallos de implementación, calidad y usabilidad. Básicamente consiste en probar la aplicación construida, para detectar un gran número de errores.

Los métodos de prueba definen qué estrategia seguir en cuanto a la verificación y validación del sistema, ya que están diseñados con el propósito de descubrir fallos y no para demostrar que el software funciona, siendo más razonable diseñar pruebas en aquellas partes donde la probabilidad de fallo es mayor. Existen dos tipos de pruebas que se detallarán a continuación:

Prueba de Caja Blanca: se denomina **caja blanca** a un tipo de pruebas de software que se realizan sobre las funciones internas de un módulo, las que son ejecutadas por los miembros del equipo. Entre las técnicas usadas se encuentran; la cobertura de caminos, pruebas sobre las expresiones lógico-aritméticas y comprobación de bucles. Estas pruebas son de gran valor para la validación de software, ya que los propios programadores se dan cuenta de los errores cometidos y a su vez corrigen los futuros riesgos del proyecto. En el módulo se realizaron pruebas utilizando el framework JUnit, analizado en capítulos anteriores. Dicho framework asegura que el módulo funcione correctamente por separado, entrando al nivel de detalle del código y depurando los métodos utilizados en cada clase creada.

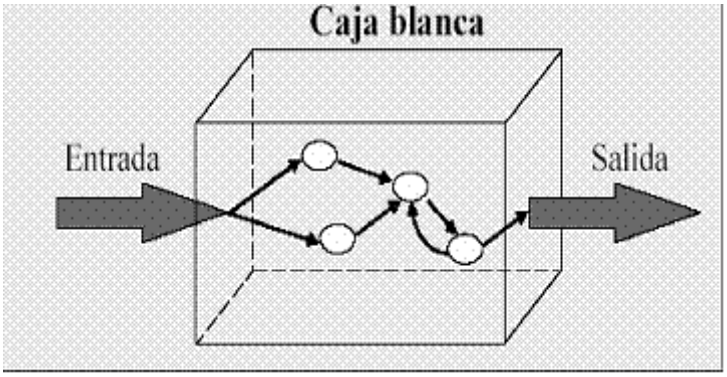



**Figura 26: Método Caja Blanca.**

Prueba de Caja Negra: se denomina **caja negra** a aquel elemento que es estudiado desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno. En otras palabras, de una caja negra interesa su forma de interactuar con el medio que le rodea, entendiendo **qué es lo que hace**, pero sin dar importancia a **cómo lo hace**. Por tanto, de una caja negra deben estar muy bien definidas sus entradas y salidas, es decir, su interfaz; en cambio, no se precisa definir ni conocer los detalles internos de su funcionamiento.

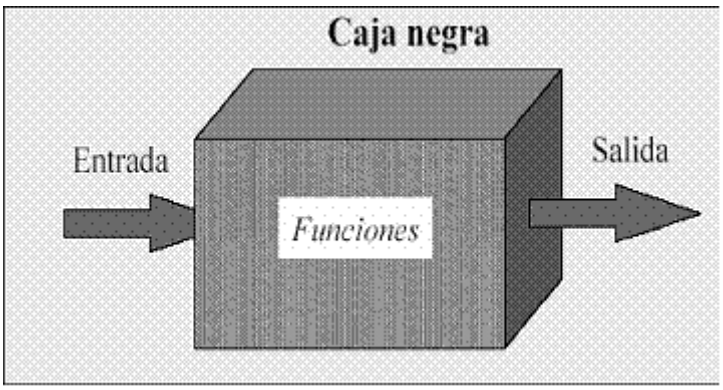



**Figura 27: Método de Caja Negra**

### 3.2.1 Pruebas Aplicadas al Sistema

Para encontrar la mayor cantidad de errores y hacer una buena validación de software, las pruebas se dividieron de la siguiente forma: Pruebas Internas, Pruebas Cruzadas, Pruebas de Liberación, Pruebas de Aceptación y Pruebas Piloto. Para cada una de ellas se generaron varios gráficos, donde se tuvo en cuenta la prioridad de las no conformidades y pedidos de cambios. Dichas pruebas pertenecen al método de caja negra, las cuales están regidas por los

casos de pruebas, que fueron generados a partir de la especificación de casos de uso. Los casos de prueba representan los requisitos que se esperan que cumpla la aplicación, por lo que al escribirlos también se documenta el sistema. Es fácil comprobar si el código desarrollado es correcto, simplemente hay que pesar los casos de prueba. De esta forma, se reducen los tiempos de depuración de errores.

Seguidamente se muestra una explicación breve de las pruebas utilizadas en la estrategia.

**Pruebas Cruzadas:** Pruebas realizadas al sistema por los equipos de desarrollo del proyecto. Se realizaron con el fin de encontrar la mayor cantidad de errores posibles en término de validaciones, pautas definidas por la arquitectura de información, formato de los campos, entre otras. Esta prueba tiene un beneficio extra, ya que se encuentran errores altamente complejos antes que se conviertan en un riesgo.

**Pruebas Internas**: Pruebas realizadas al sistema por el equipo de calidad interna del proyecto, se realizaron con el fin de entregar un producto lo más libre de errores posibles. Se centraron en el cumplimiento de las funcionalidades descritas en el listado de requisitos y de Casos de Uso.

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

**Pruebas de Aceptación**: Pruebas realizadas por los clientes para comprobar que el sistema cumple con sus expectativas desde el punto de vista funcional y no funcional.

**Pruebas Piloto**: Pruebas realizadas por los clientes para comprobar el rendimiento de la aplicación y su respuesta en un entorno real de trabajo.

A continuación se muestran de forma detallada los errores encontrados durante las pruebas realizadas en la tercera etapa del software.

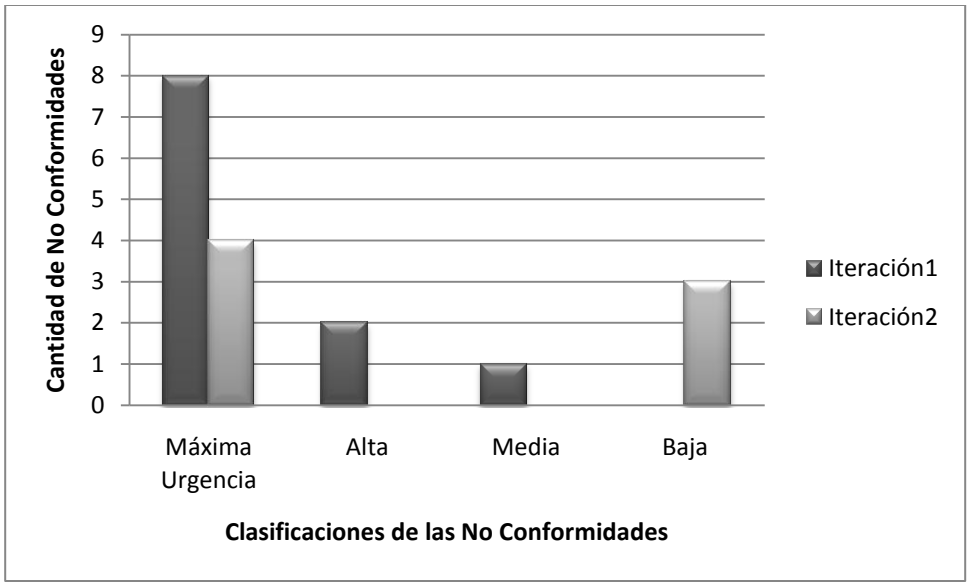



**Figura 28: Pruebas Cruzadas**

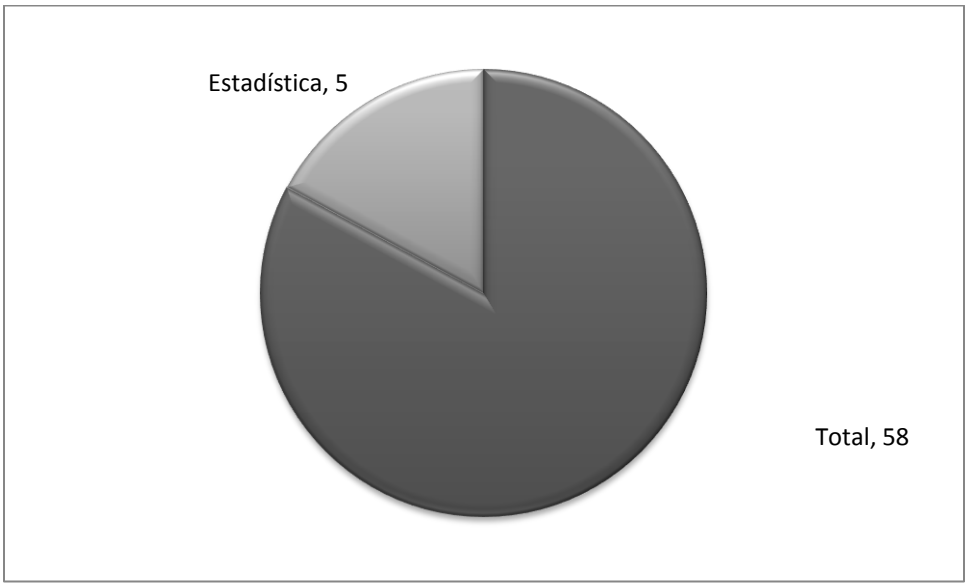



**Figura 29: Pruebas Cruzadas (Comparación con el resto de los módulos)**

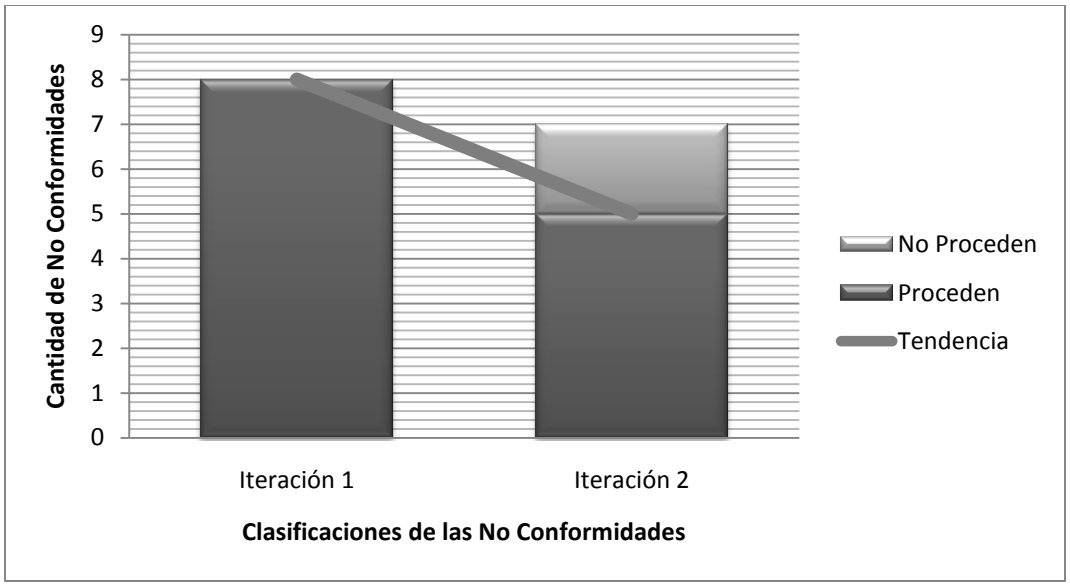



**Figura 30: Pruebas Cruzadas (no conformidades)**

En las gráficas se puede apreciar un descenso en cuanto a la cantidad de no conformidades detectadas durante la primera y la segunda iteración, tendiendo el módulo a mejorar en toda su totalidad. Se puede concluir que los errores detectados fueron subsanados en su mayor totalidad.

Las iteraciones están concebidas en la herramienta TRAC con los siguientes nombres respectivamente: Pruebas Cruzadas 1ra Iteración Liberación 3ra Etapa y Pruebas Cruzadas 2da Iteración Liberación 3ra Etapa.

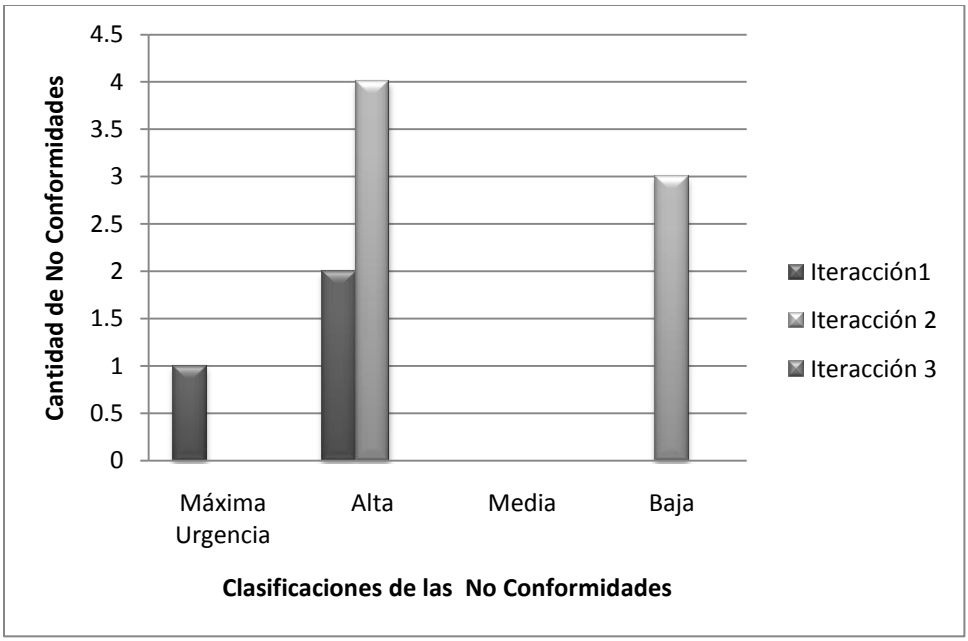



**Figura 31: Prueba de Calidad Interna**

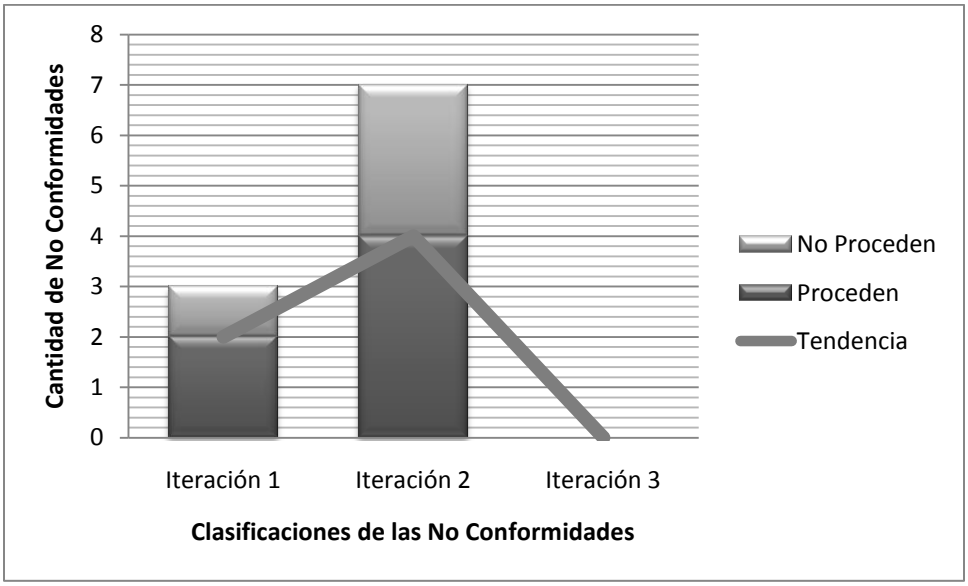



**Figura 32: Prueba de Calidad Interna (no conformidades)**

Las pruebas de Calidad Interna tienen un beneficio extra, ya que a medida que se itera, se revisa si fueron corregidos los fallos de la etapa anterior y surgen nuevos errores, perfeccionando la aplicación a la máxima expresión. Como se observa, hubo un ascenso en la cantidad de no conformidades detectadas durante la segunda iteración de pruebas con respecto a la primera. Este fenómeno ocurrió debido a que los probadores eran estudiantes poco experimentados, tanto en el proceso de prueba como en el funcionamiento del sistema y

no fueron capaces de detectar todos los defectos existentes en el sistema en una primera iteración. Para la segunda, ya habían adquirido un cúmulo de experiencias que les facilitaron encontrar nuevos defectos, que como se puede observar, fueron completamente subsanados para la tercera.

Las iteraciones están creadas en la herramienta TRAC con los siguientes nombres respectivamente: Pruebas de Calidad Interna 3ra Etapa 1ra Iteración, Pruebas de Calidad Interna 3ra Etapa 2da Iteración y Prueba Final de Liberación Calidad Interna del SIIPOL v3.0

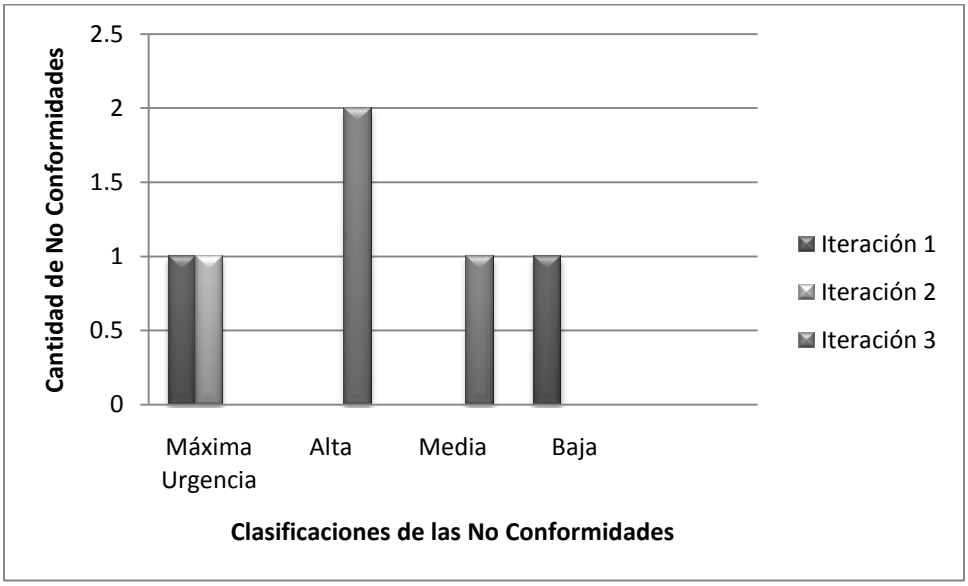



**Figura 33: Prueba de Liberación**

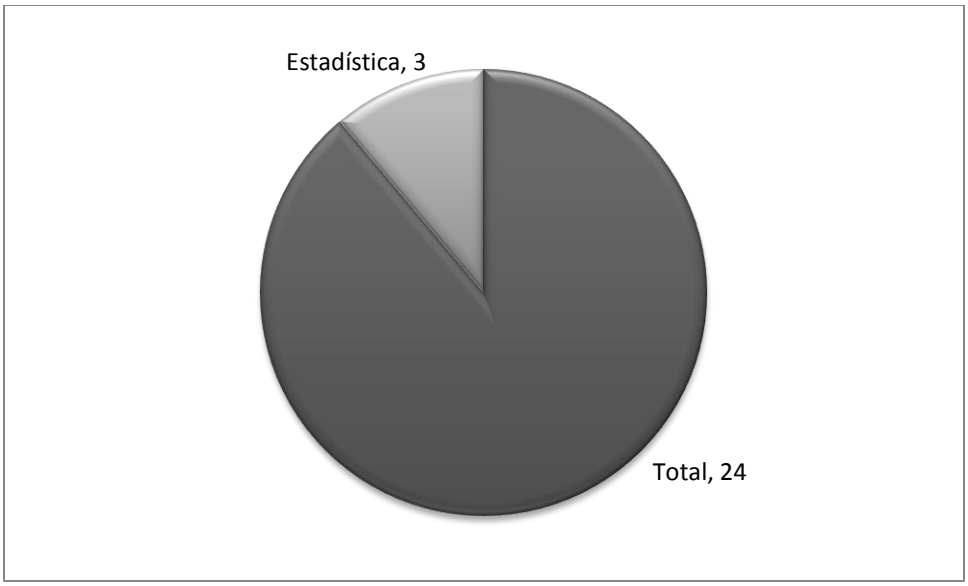



**Figura 34: Prueba de Liberación (Comparación con el resto de los módulos)**

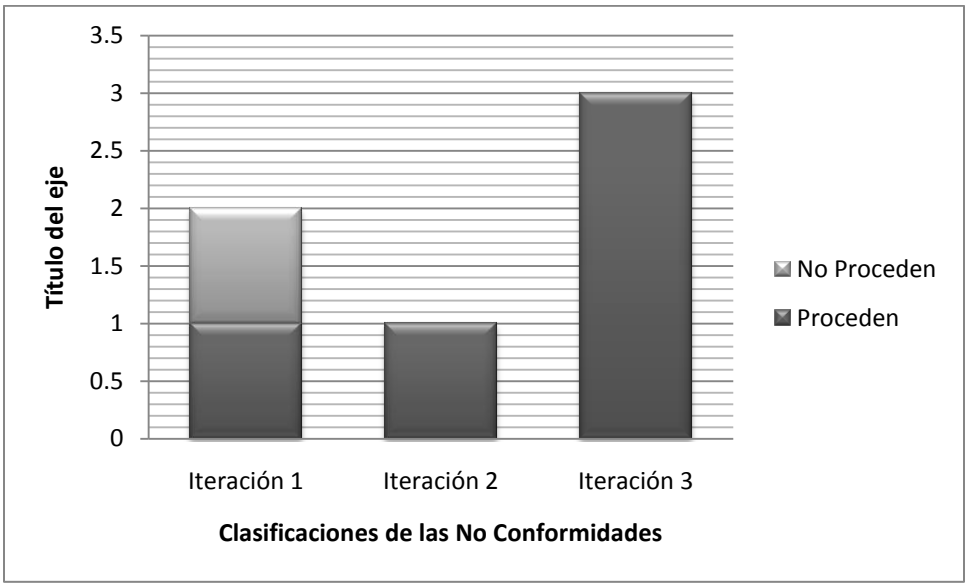



**Figura 35: Prueba de Liberación (no conformidades)**

Durante las pruebas de liberación se puede observar que la cantidad de no conformidades disminuyó considerablemente, detectándose un número ínfimo de errores. Esto demuestra la importancia de las pruebas cruzadas y las de calidad interna realizadas anteriormente, las cuales permitieron entregar a Calisoft un software lo más libre de errores posibles.

Las iteraciones están establecidas en la herramienta TRAC con los siguientes nombres respectivamente: Pruebas de Liberación 3ra Etapa 1ra Iteración, Pruebas de Liberación 3ra Etapa 2da Iteración y Pruebas de Liberación 3ra Etapa 3ra Iteración.

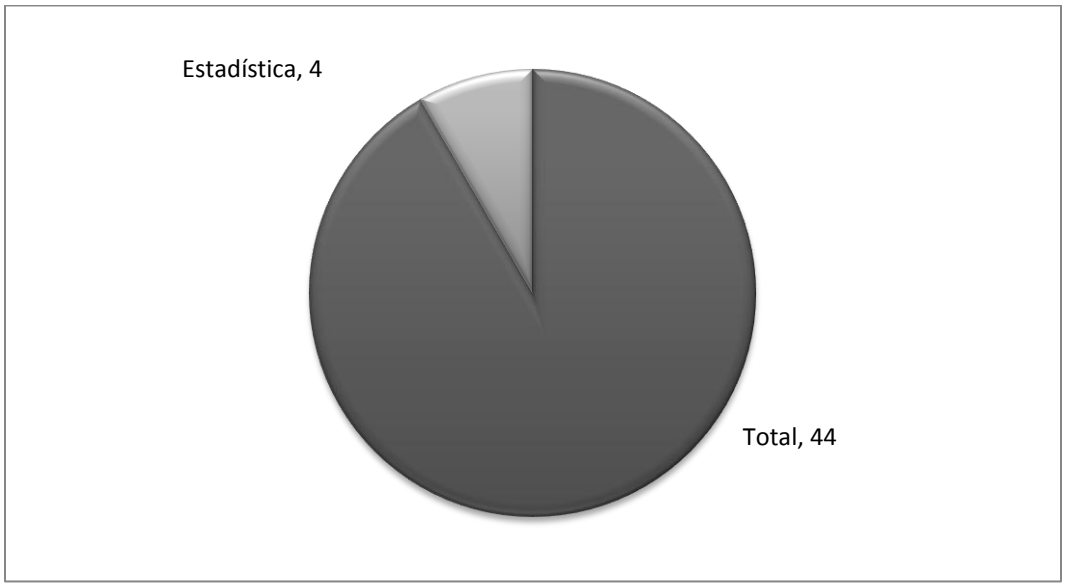



**Figura 36: Prueba de Aceptación (pedidos de cambios)**

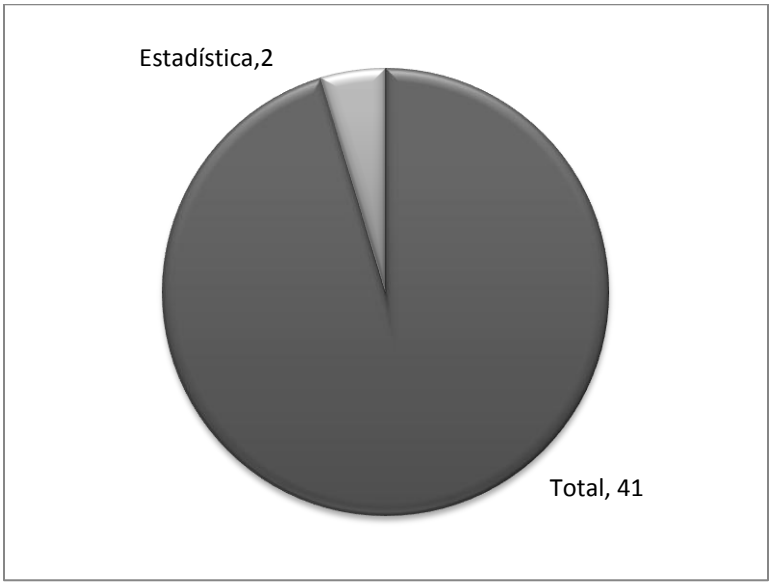



**Figura 37: Prueba de Aceptación (no conformidades)**

Con las pruebas de aceptación se logró una elevada satisfacción del cliente, pues los pedidos de cambio y las no conformidades fueron mínimas comparadas con el resto de los módulos. No se encontraron errores latentes o escondidos que pudieran ir degradando el

funcionamiento del sistema. Estas pruebas son muy importantes, ya que definen el paso hacia nuevas fases del proyecto, como el despliegue y mantenimiento.

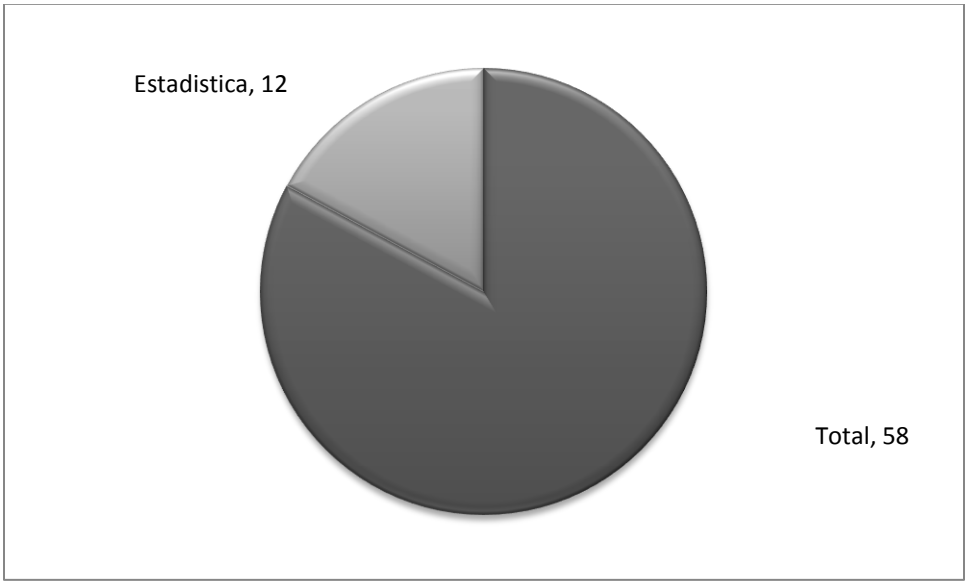



**Figura 38: Prueba Piloto (pedido de Cambios)**

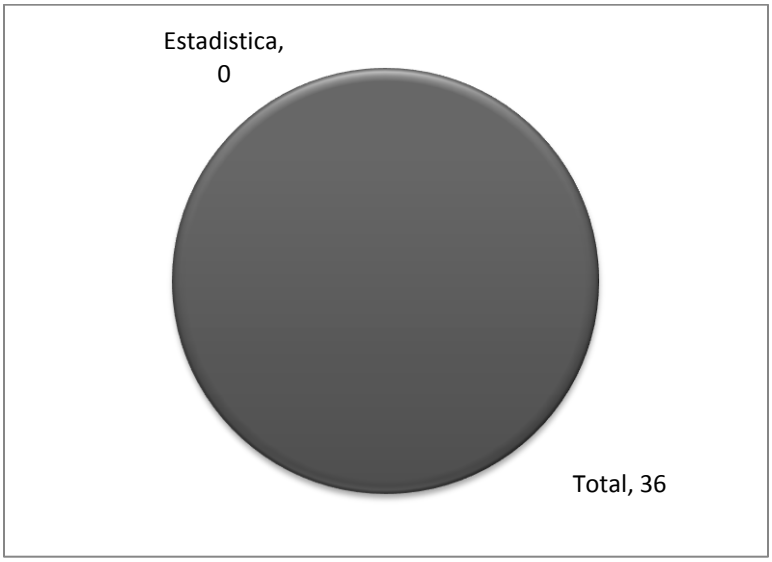



**Figura 39: Prueba Piloto (no conformidades)**

Una vez realizadas las pruebas piloto del sistema, se encontraron más pedidos de cambio con respecto a las pruebas de aceptación, esto ocurrió debido a que los usuarios se enfrentaron por primera vez directamente al sistema, comprobando que sustentaba de forma automática su trabajo diario. Su puesta en práctica en un entorno real demostró que eran necesarios un conjunto de cambios que aumentarán la usabilidad del sistema y su total

correspondencia con el trabajo diario de la institución, fundamentalmente desde el punto de vista de estadística.

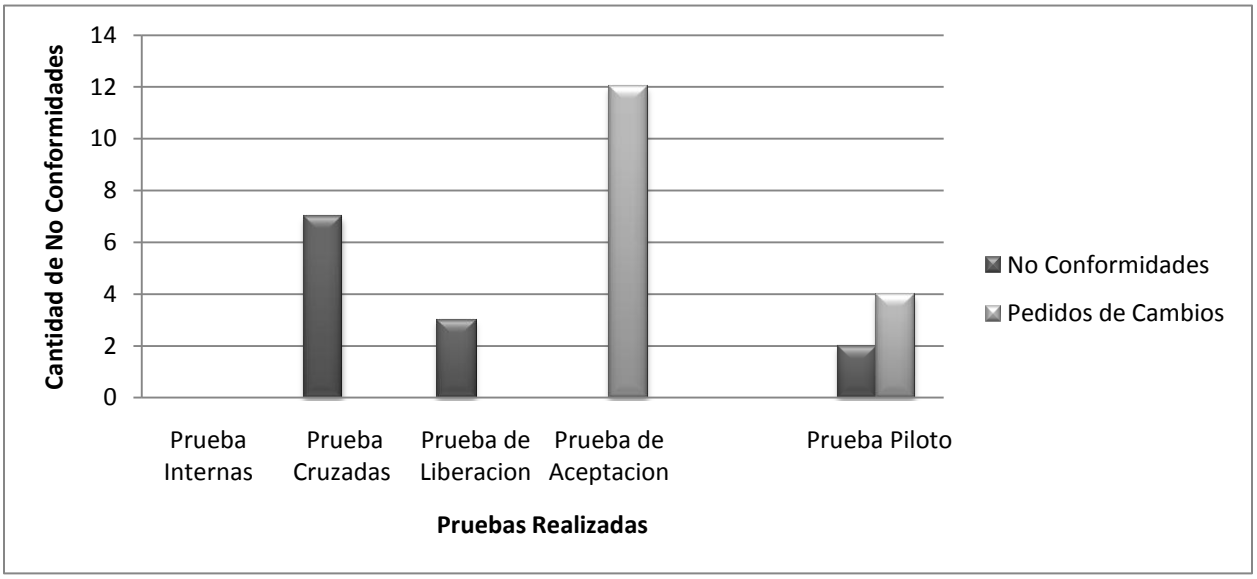



**Figura 40: Resumen General de las Pruebas**

Para el gráfico resumen de las pruebas se tuvo en cuenta la última iteración de cada prueba realizada. Como muestra la figura, durante las tres primeras pruebas los pedidos de cambio fueron nulos, ya que las mismas no fueron realizadas por expertos en el funcionamiento de la institución y fueron guiadas por los Casos de Prueba, que a su vez, sufrieron transformaciones posteriormente una vez detectados los pedidos de cambio. Sin embargo, en las dos últimas pruebas es todo lo contrario, ya que con la estrategia de pruebas seleccionada llegó el módulo a los usuarios finales prácticamente libre de errores, pero siendo estos los expertos de la institución detectaron varios pedidos de cambios que aumentarán la usabilidad del sistema y su total correspondencia con el trabajo diario de la institución.

### 3.3 Conclusiones

Una correcta estrategia de validación y verificación ayuda a mejorar y asegurar la calidad de su desarrollo y permite reducir los costos de corrección de errores. Las ventajas son claras: un mayor control sobre el proceso, una identificación temprana de errores, problemas y una reducción drástica de los costos para subsanar dichos errores, por eso la atención esmerada que se le presta a esta actividad.

Las pruebas realizadas dieron la medida de cómo el módulo iba mejorando en toda su totalidad, donde los pedidos de cambio y no conformidades fueron pequeños comparados con el resto de los módulos.

**Conclusiones**

El estudio del arte marcó un momento importante en la investigación, ya que se fijaron las herramientas a utilizar de las cuales se enunciaron sus principales características, ventajas y desventajas.

AUP facilitó la creación de los artefactos necesarios y un producto con una elevada calidad.

Las herramientas y definiciones arquitectónicas puestas en práctica permitieron el desarrollo claro y fluido de un sistema construido sobre bases sólidas y un entorno bien definido.

Se presentó una serie de conceptos importantes para la comprensión de la solución informática que da origen a esta investigación; datos referentes a nivel mundial como fuente de estudio y consulta por parte del autor.

El uso de las tarjetas CRC permitió realizar un diseño adecuado para el Módulo Estadística y a su vez garantizó un elevado porcentaje de los procesos a automatizar.

Se elaboraron el modelo de diseño, modelo de datos y el modelo de implementación quedando una aplicación lista para el flujo de prueba.

La validación del software fue de gran importancia, pues se comprobó el adecuado funcionamiento de las principales funcionalidades del módulo, logrando una elevada satisfacción con el usuario final.

**Recomendaciones**

Al concluir las pruebas se impartieron cursos a los funcionarios de distintos estados de Venezuela, donde se recomendó incluir funcionalidades en el módulo entre las que se encuentran:

- Dar la posibilidad al consultor de imprimir las estadísticas predefinidas de forma general o por estado.
- Adicionar estadísticas para cuantificar los informes realizados en los distintos despachos de la dirección de investigación Criminalística.
- Incorporar al sistema estadístico las funcionalidades de un sistema desarrollado para el Ministerio del Interior y Justicia con el fin de combinar todas las funcionalidades referentes a las estadísticas de la institución en un solo sistema.

Además:

- Incorporar las estadísticas asociadas a los nuevos módulos a incluir en el SIIPOL.
- Promover la utilización de metodologías ágiles en el desarrollo de futuros proyectos a partir de las experiencias positivas obtenidas.
- Realizar un proceso de refactorización no solo al Módulo Estadística, sino al sistema de forma general, con el fin de limar cualquier aspereza detectada tras la validación del software.

.

## Referencias Bibliográficas

1. **Rodríguez, Yadiel Ramos.** *Sistema de Apoyo a la Investigación Criminalística para el Cuerpo de Investigación Científica Penales y Criminalística de Venezuela.* Habana : s.n., 2007.

2. **Muñoz, David Ruiz.** *Manual de Estadística.*

3. **reserved, Laura Chinchilla – Seguridad 2010. All rights.** Laura firme y honesta. *Sistema integrado de estadísticas policiales / Vigilancia electrónica.* [Online] 2010. http://www.lauracr.com/seguridad/2010/sistema-integrado-de-Estadísticas-policiales-vigilancia-electronica/.

4. Agenda Magna. *Sistema de denuncias policiales.* [Online] diciembre 7, 2009. http://agendamagna.wordpress.com/2009/12/07/sistema-de-denuncias-policiales/.

5. **Boehm, Barry and Turner, Richard.** *Balancing Agility and Discipline: a guide for the perplexed.* Addison Wesley : s.n., 2005. 0321186125..

6. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Pearson Education.S.A, 2000. 84-7829-036-2.

7. **Copyright © 1997-2010 Object Management Group, Inc. All Rights Reserved.** UNIFIED MODELING LANGUAGE. *UML® Resource Page.* [Online] 2007-2010. http://www.uml.org/.

8. *Visual Paradigm for UML.* [Online] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

9. **Oracle Corporation.** Java EE at a Glance. [Online] 2010. http://java.sun.com/javaee/index.jsp.

10. **Sitio Oficial JAVA. [En línea] 2009.** Sitio Oficial JAVA. [Online] 2009. http://java.com/es.

11. **www.springsource.org .** Spring Source Community. [Online] 2010. [Cited: enero 13, 2010.] http://www.springsource.org/.

12. HIBERNATE. *Relational Persistence for Java & .NET.* [Online] http://www.hibernate.org/.

13. JUnit.org Recursos para la Prueba de Desarrollo Impulsado. [Online] 2009. http://www.junit.org/.

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

15. *La flecha,tu diario de ciencia y tecnologia.* [Online] http://www.laflecha.net/canales/softlibre/noticias/red-hat-presenta-developer-studio-basado-en-eclipse.

16. **Presman, Roger S.** *Ingeniaría de Software. Un enfoque práctico.*

17. **Tamayo, Lic. Deymis.** *REQUERIMIENTOS SUPLEMENTARIOS.* Habana : s.n., 2007.

18. **Rodríguez., Msc. Yadiel Ramos.** *ARTEFACTOS A GENERAR.* 6/05/2009.

19. **Larman, Craig.** *UML y Patrones.*

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

## Bibliografías

1. **Rodríguez, Yadiel Ramos.** *Sistema de Apoyo a la Investigación Criminalística para el Cuerpo de Investigación Científica Penales y Criminalística de Venezuela.* Habana : s.n., 2007.

2. **Muñoz, David Ruiz.** *Manual de Estadística.*

3. **reserved, Laura Chinchilla – Seguridad 2010. All rights.** Laura firme y honesta. *Sistema integrado de estadísticas policiales / Vigilancia electrónica.* [Online] 2010. http://www.lauracr.com/seguridad/2010/sistema-integrado-de-Estadísticas-policiales-vigilancia-electronica/.

4. Agenda Magna. *Sistema de denuncias policiales.* [Online] diciembre 7, 2009. http://agendamagna.wordpress.com/2009/12/07/sistema-de-denuncias-policiales/.

5. **Boehm, Barry and Turner, Richard.** *Balancing Agility and Discipline: a guide for the perplexed.* Addison Wesley : s.n., 2005. 0321186125..

6. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Pearson Education.S.A, 2000. 84-7829-036-2.

7. **Copyright © 1997-2010 Object Management Group, Inc. All Rights Reserved.** UNIFIED MODELING LANGUAGE. *UML® Resource Page.* [Online] 2007-2010. http://www.uml.org/.

8. *Visual Paradigm for UML.* [Online] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

9. **Oracle Corporation.** Java EE at a Glance. [Online] 2010. http://java.sun.com/javaee/index.jsp.

10. **Sitio Oficial JAVA. [En línea] 2009.** Sitio Oficial JAVA. [Online] 2009. http://java.com/es.

11. **www.springsource.org .** Spring Source Community. [Online] 2010. [Cited: enero 13, 2010.] http://www.springsource.org/.

12. HIBERNATE. *Relational Persistence for Java & .NET.* [Online] http://www.hibernate.org/.

13. JUnit.org Recursos para la Prueba de Desarrollo Impulsado. [Online] 2009. http://www.junit.org/.

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

15. *La flecha,tu diario de ciencia y tecnologia.* [Online] http://www.laflecha.net/canales/softlibre/noticias/red-hat-presenta-developer-studio-basado-en-eclipse.

16. **Presman, Roger S.** *Ingeniaría de Software. Un enfoque práctico.*

17. **Tamayo, Lic. Deymis.** *REQUERIMIENTOS SUPLEMENTARIOS.* Habana : s.n., 2007.

18. **Rodríguez., Msc. Yadiel Ramos.** *ARTEFACTOS A GENERAR.* 6/05/2009.

19. **Larman, Craig.** *UML y Patrones.*

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

21. **Software, Departamento de Ingeniería de.** *Conferencias de ISW. UC.* 2008-2009.

22. **ORACLE CORPORATION.** ORACLE. [Online] 2009. http://www.oracle.com/index.html.

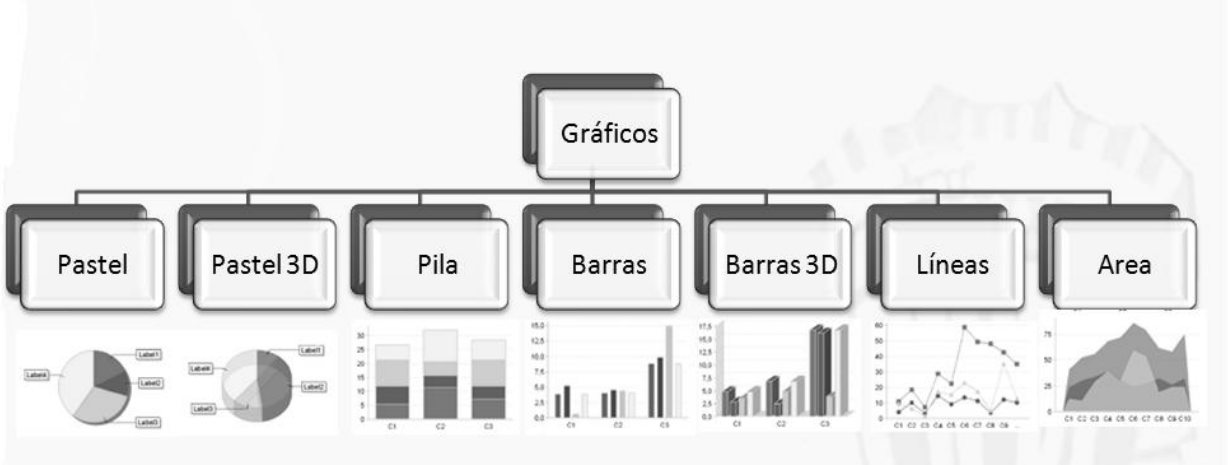
23. **Ambysoft Inc.** *El Proceso Unificado Ágil v1.1.* [Online] mayo 13, 2005-2006. http://cgi.una.ac.cr/AUP/index.html.

24. Documento de Especificación Requerimiento No funcionales del Proyecto Mejoramiento de procesos Analsisis y Diseño del Sistema de Información. [Online] junio 7, 2006. http://www.minproteccionsocial.gov.co/vbecontent/library/documents/DocNewsNo16758DocumentNo5401.PDF.

## Anexos

### Anexo 1: Tipos de Estadísticas


### Anexo 2: Tipos de Gráficos



**Anexo 3: Estilos para generar reportes.**

## Anexo 4: Tarjetas CRC para la jerarquía del Módulo Estadística

**ReporteEstadisticoCasoService**

**Description:** Servicio encargado de gestionar las estadística de Actas Procesales

Responsibilities:

| Name | Collaborator |
|---|---|
| Tipos de diligencias más hecha | |
| Tipos de diligencias menos hechas | |
| Funcionarios con más casos asignados | DependenciaInterna<br>AdministracionFacade |
| Funcionarios con menos casos asignados | DependenciaInterna<br>AdministracionFacade |
| Funcionarios con casos sin cerrar | DependenciaInterna<br>AdministracionFacade |
| Casos con mayor tiempo aperturados | DependenciaInterna<br>AdministracionFacade |
| Funcionarios que más rápido han cerrado o resuelto casos | DependenciaInterna<br>AdministracionFacade |
| Funcionarios que más se demoran en cerrar o resolver casos | DependenciaInterna<br>AdministracionFacade |
| Casos de mayor importancia | DependenciaInterna<br>AdministracionFacade |

**EstadisticasUtil**

**Description:** Servicio encargado de manejar datos comun a todas las estadísticas

Responsibilities:

| Name | Collaborator |
|---|---|
| Formar Cadena | CalendarUtils |
| Formar Cadena Caracter | |
| Cantidad de Delegaciones | ActaProcesal<br>Diligencia |
| Cantidad de Tipos | ActaProcesal<br>Diligencia |
| Cantidad de Tipos Específicos | Solicitud<br>Informe |
| Cantidad de Tipos de Experticias | |

**ReporteEstadisticoDelitoService**

**Description:** Servicio encargado de gestionar las estadísticas de Delitos

Responsibilities:

| Name | Collaborator |
|---|---|
| Principales delitos | Persona |
| Delitos totales | Persona |
| Delitos personalizados | Persona<br>EstadisticasUtil |
| Días de mayor índice delictivo | |
| Zonas de mayor índice delictivo | |
| Horas de mayor índice delictivo | |
| Delitos más frecuentes | |
| Homicidios | Persona |
| Robos de autos | Persona |
| Zonas de robo o hurto de vehiculos | |
| Zonas de homicidios | |
| Promedio de delitos diarios | |

**ReporteEstadisticoExpedienteDisciplinarioService**

**Description:** Servicio encargado de gestionar las estadística de Expedientes Disciplinarios

Responsibilities:

| Name | Collaborator |
|---|---|
| Expedientes Disciplinarios concluidos | Persona |
| Expedientes Disciplinarios en curso | Persona |
| Expedientes Disciplinarios iniciados | Persona |
| Expedientes Disciplinarios remitidos | Persona |
| Expedientes Disciplinarios de mayor importancia | Persona |
| Diligencias practicadas | |
| Diligencias practicadas en curso | |
| Expedientes Disciplinarios personalizados | EstadisticasUtil<br>Persona |
| Faltas más frecuentes | DependenciaInterna<br>AdministracionFacade |
| Diligencias más practicadas por inspectoría | |
| Diligencias menos practicadas por inspectoría | |
| Faltas Disciplinarias personalizadas | EstadisticasUtil<br>Persona |
| Faltas Disciplinarias por fechas | |

**ReporteEstadisticoExperticiaCriminalistica**

**Description:** Servicio encargado de gestionar las estadísticas de Experticias Criminalísticas

Responsibilities:

| Name | Collaborator |
|---|---|
| Listado de funcionarios experticias | |
| Listado de dependencias experticias | |
| Listado de estudios experticias | |
| Experticias Criminalísticas personalizadas | EstadisticasUtil<br>Diligencia |
| Obtener solicitudes e informes por tipo de experticia y estudio | Diligencia<br>Comunicacion<br>FuncionarioDiligencia |
| Análisis físico-comparativos atendiendo a solicitud en fecha o fuera de fecha | InformePericial |
| Análisis físico-comparativos remitidos fuera de fecha | InformePericial |
| Análisis físico-comparativos practicados fuera de fecha | InformePericial |

**ReporteEstadisticoExperticiaForenseService**

**Description:** Servicio encargado de gestionar las estadísticas de Experticias Forense

Responsibilities:

| Name | Collaborator |
|---|---|
| Estadísticas forenses por grupos | |
| Relación de cadáveres por muerte violenta | |
| Relación de cadáveres levantados en la morgue | |
| Relación de cadáveres levantados en el sitio del suceso | |
| Relación de cadáveres recibidos | |
| Experticias forenses personalizadas | Diligencia<br>EstadisticasUtil |
| Experticias forenses realizadas por fecha | Diligencia |

**ReporteEstadisticoEvidenciaCriminalisticaService**

**Description:** Servicio encargado de gestionar las estadísticas de Evidencias Criminalísticas

Responsibilities:

| Name | Collaborator |
|---|---|
| Armas más recibidas | |
| Armas menos recibidas | |
| Armas más procesadas | |
| Armas menos procesadas | |
| Vehículos más estudiados | |
| Vehículos menos estudiados | |
| Evidencias atendiendo a criterios personalizados | EstadisticasUtil |
| Evidencias por tipo | Evidencia |
| Vehículos por tipo | |
| | |

**ReporteEstadisticoFuncionariosCriminalisticaService**

**Description:** Servicio encargado de gestionar las estadísticas de Funcionarios de Criminalística

Responsibilities:

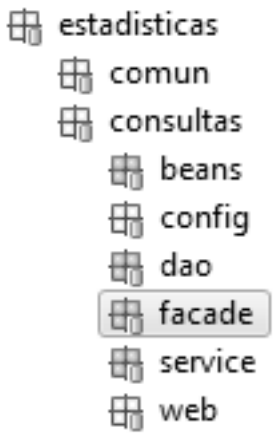
| Name | Collaborator |
|---|---|
| Actuaciones Periciales realizadas por funcionarios | |
| Solicitudes de avalúo recibidas por funcionarios | |
| Experticias físico-comparativas realizadas por funcionarios | |
| Experticias audiovisuales realizadas por funcionarios | |
| Remisiones realizadas por funcionario | |
| Emisiones realizadas por funcionario | |
| Funcionarios personalizados | EstadisticasUtil |
| Funcionarios comparativos | |
| Cantidad de experticias realizadas por funcionario | |

| ReporteEstadisticoCTACService | |
|---|---|
| **Description:** Servicio encargado de gestionar las estadísticas de Notificaciones Telefónicas de Vehículos | |
| Responsibilities: | |
| Name | Collaborator |
| Notificaciones Telefonicas por criterio | |
| Notificaciones de robo de vehículo | |
| Notificaciones de hurto de vehículo | |
| Vehículos recuperados que fueron denunciados | |
| Vehículos recuperados que no fueron denunciados | |
| Notificaciones cuantitativas de vehículos | |
| Notificaciones de vehículo que no han sido denunciados ni recuperados | |
| Notificaciones Personalizadas | EstadisticasUtil<br>AnalisisInformacionFacade |

| ReporteEstadisticoTransmisionesService | |
|---|---|
| **Description:** Servicio encargado de gestionar las estadísticas de | |
| Responsibilities: | |
| Name | Collaborator |
| transmisionesFuncionariosMuerto | |
| transmisionesGenerales | |
| transmisionesPersonasMuertas | |
| transmisionesRoboHurto | |
| transmisionesVerificacionInforma | |
| transmisionesPersonalizadas | EstadisticasUtil |
| dameCantidadesTiempoTransmisi | ReporteEstadisticoComparativo |

| ReporteEstadisticoAccesoService | |
|---|---|
| **Description:** Servicio encargado de gestionar las estadística de Acc | |
| Responsibilities: | |
| Name | Collaborator |
| pasesVisitanteSede | RegistroAcceso |
| solicitudesSistema | ReporteEstadisticoComparativo |
| dameCantidadesTiempoAcceso | ReporteEstadisticoComparativo |
| cantidad | |
| accesoPersonalizadas | EstadisticasUtil,EstadisticasAcceso |

**Anexo 5: Patrón Facade**



**Anexo 6: Patrón Interface**

**Glosario**

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

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

**API:** es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Usados generalmente en las bibliotecas.

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

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

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

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

**JDK:** es un software que provee herramientas de desarrollo para la creación de programas en java. Puede instalarse en una computadora local o en una unidad de red.

**MVC:** Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web.

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

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

**Servlet:** Es un objeto que se ejecuta en un servidor o contenedor JEE, fue especialmente diseñado para ofrecer contenido dinámico desde un servidor web.

**Singletons:** El patrón de diseño (instancia única) está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto.

**SDK:** es generalmente un conjunto de herramientas de desarrollo que le permite a un programador crear aplicaciones para un sistema concreto, por ejemplo ciertos paquetes de

software, frameworks, plataformas de hardware, computadoras, videoconsolas, sistemas operativos, etc.

**SIIPOL:** Sistema de Investigación e Información Policial.

**Spring:** Framework de la capa de lógica de negocio para Java.