Universidad de las Ciencias Informáticas

Facultad III

Título: Implementación y Pruebas del módulo Ordinario del Subsistema Procesos Penales del proyecto Sistema de Gestión Fiscal.

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autor(es): Yamilé Stivens Portela
Maurice Manuel Velázquez Torres

Tutor(es): Ing. Yarisleidis Fernández Rivera
Ing. Ariel Exabie Pérez

La Habana, Cuba

Junio 2011


El futuro de nuestra Patria tiene que ser
necesariamente un futuro de hombres de ciencia.

Declaración de Autoría

Declaro que soy el único autor de este trabajo y autorizo al <nombre área> 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 _____.

Yamilé Stivens Portela

_____

Maurice Manuel Velázquez Torres

_____

Ing. Yarisleidis Fernández Rivera

_____

Ing. Ariel Exabie Peréz

_____

Datos de Contacto

Tutor: Ing. Yarisleidis Fernández Rivera

Correo electrónico: yfernandezr@uci.cu

Síntesis: Profesora con 4 años de experiencia. Ingeniería en Ciencias Informáticas. Actualmente se desempeña como Analista de Procesos Penales del proyecto Sistema de Gestión Fiscal.

Categoría Docente: Instructor

Tutor: Ing. Ariel Exabie Pérez

Correo electrónico: aexabie@uci.cu

Síntesis: Profesor recién graduado de Ingeniería en Ciencias Informáticas. Actualmente se desempeña como Jefe del módulo Ordinario del proyecto Sistema de Gestión Fiscal.

Categoría Docente: Adiestrado

Agradecimientos

A mi mamita, por depositar en mí su confianza. Por ser un ejemplo de abnegación, sacrificio y excelente madre. Te quiero mucho.

A mi papá, por transmitirme siempre seguridad, por brindarme su apoyo e incondicionalidad.

A mi hermana por ser mi consejera con sus salmos y frases del día. Gracias ale, te quiero mucho.

A mi chiquitico, mi novio y amigo por aguantarme todos estos años, por amarme y estar junto a mí en las buenas y las malas. Yo también te amo mucho.

A mis suegros por todo su cariño y apoyo. Los quiero de corazón. Gracias por aceptarme en su familia.

A mis cuñaditas *Gleysin y Juli por su preocupación y atención conmigo. Las quiero mucho.*

A mis tios(as), primos(as) y abuelos(as) por estar ahí cuando necesité su apoyo y comprensión.

A mi gordita, gracias kathy por estar siempre dispuesta a ayudarme, eres mi amiguita del alma.

A mis amigos Diamy, Ari, Daniel y Martín, siempre habrá un recuerdo de ustedes en mi corazón.

A mi team del dominó, Yunet, Dailín, Semi, La flaca, Maylín y a mi tocaya, gracias por alegrarme las tardes.

A todos mis compañeros de estudio y profesores por haber aportado un granito de arena en mi formación.

A mis tutores, gracias por estar siempre ahí cuando los necesité, al igual que el equipo de proyecto de Fiscalía.

Yami

A mis compañeros de trabajo y de convivencia en esta universidad por soportarme y ayudarme en todo momento.

A mi tutor por el apoyo brindado durante la realización de esta investigación.

A todos los que de una forma u otra han contribuido a nuestra formación profesional durante estos años de estudio.

A mis amigos de aquí y de allá por estar siempre presentes incondicionalmente.

Mauro

Dedicatoria

A nuestro Comandante en Jefe Fidel Castro Ruz, por ser nuestro guía y nuestra luz e iniciador de la Universidad de las Ciencias Informáticas.

A mis padres por su entrega y dedicación.

Yami

A mi padre y mi madre, por todo lo que se han sacrificado por sus hijos durante toda su vida, sabiendo balancear entre familia y profesión. Por ser excelentes padres y aún mejores personas.

A mi hermana, por su apoyo incondicional durante estos 5 años de largos estudios y separación.

A Dalia por soportarme, apoyarme y estar conmigo siempre aún en la distancia.

A mi familia, gracias por la educación y el ejemplo que me han brindado.

Mauro

Resumen

Como consecuencia de los grandes avances tecnológicos y del gran volumen de información que se genera actualmente en las entidades cubanas, éstas han tratado de nutrirse al máximo con el uso de la informática para su desarrollo interno y el mejoramiento en los procesos que se gestionan en las mismas. En este caso se encuentra la Fiscalía General de la República, que ha encaminado parte de sus esfuerzos a informatizar los procesos que se llevan a cabo en las fiscalías del país, para de esa manera brindar un mejor servicio a la población, incrementar el control sobre el trabajo de los fiscales y viabilizar sus procesos.

El presente trabajo de diploma contribuirá a agilizar los Procesos Ordinarios que se desarrollan en las fiscalías, a través de la implementación del módulo Ordinario del proyecto Sistema de Gestión Fiscal, cumpliendo con los requerimientos analizados y priorizados en etapas precedentes del proyecto. Estos procesos comprenden aquellos delitos sancionables por más de un año de privación de libertad, multas mayores de 300 cuotas o ambos.

PALABRAS CLAVE
Fiscalía General de la República, Procesos Penales, Procesos Ordinarios.

Índice de contenido

Índice de tablas IX
Índice de figuras X
Introducción 1
Capítulo 1 6
1.1. Introducción 6
1.2. Valoración del estado del arte 6
1.2.1. Situación de los Sistemas de Gestión Fiscal en el mundo 6
1.2.2. Situación en Cuba 7
1.2.3. Conclusión acerca de los Sistemas de Gestión Fiscal. 8
1.3. Técnicas de Programación. 8
1.3.1. Programación estructurada 8
1.3.2. Programación Orientada a Objetos 8
1.3.3. Justificación de la técnica de programación seleccionada. 9
1.4. Lenguajes de programación 9
1.4.1. Tecnologías del lado del servidor 9
1.4.2. Tecnologías del lado del cliente. 11
1.5. Herramientas 12
1.5.1. Marco de trabajo 12
1.5.2. Servidor WEB 15
1.5.3. Sistema de Gestión de Base de Datos 16
1.5.4. Entornos de Desarrollo Integrado 18
1.5.5. Sistemas de control de versiones 19
1.5.6. Pruebas de calidad 20
Capítulo 2 24
2.1. Introducción 24
2.2. Flujo del proceso 24
2.2.1. Valoración crítica de los artefactos propuestos por los analistas. 24

2.2.2. Propuesta del sistema 27
2.3. Descripción de la implementación 29
2.3.1. Modelo de Datos 29
2.3.2. Estándares de codificación 30
2.3.3. Diagrama de despliegue 33
2.3.4. Diagramas de componentes 35
2.3.5. Patrones de diseño aplicados 37
2.3.6. Patrón arquitectónico aplicado 40
2.3.7. Estrategia para la captura de errores. 42
2.3.8. Seguridad en la aplicación. 44
2.4. Descripción de clases y operaciones 45
2.4.1. Clases controladoras 45
2.4.2. Clases del modelo 47
2.4.3. Clases de la presentación 48
2.5. Conclusiones parciales 49
Capítulo 3 50
3.1. Introducción 50
3.2. Pruebas de software 50
3.2.1. Objetivos de las pruebas 50
3.3. Pruebas de caja negra 51
3.3.1. Objetivos 51
3.3.2. Aplicación de las pruebas de caja negra 51
3.4. Evaluación de la calidad utilizando métricas 52
3.4.1. Atributos de calidad que se abarcan 52
3.4.2. Tamaño Operacional de las Clases 52
3.4.3. Relaciones entre clases 55
3.5. Validación de los estándares de codificación utilizados. 56
3.6. Conclusiones parciales 57
Conclusiones Generales 58
Recomendaciones 59

Referencias bibliográficas 60
Anexos 62
Glosario de Términos 72

Índice de tablas

Tabla 1 Descripción de la clase crearExpedienteActions 47
Tabla 2 Descripción de la clase crearIndicacionesActions 47
Tabla 3 Descripción de la clase crearExpedientePeer 48
Tabla 4 Descripción de la clase de la presentación de crear expediente 49
Tabla 5 Rango de valores para la evaluación técnica de los atributos de calidad 69
Tabla 6 Resultados de la evaluación de la métrica TOC 70
Tabla 7 Rango de valores para la evaluación técnica de los atributos de calidad 71
Tabla 8 Resultados de la evaluación de la métrica RC 71
Tabla 9 Caso de pruebas correspondiente al CU Crear Expediente. Escenario Crear Expediente ..... **¡Error! Marcador no definido.**
Tabla 10 Caso de pruebas correspondiente al CU Crear Expediente. Escenario Crear Acusado ..... **¡Error! Marcador no definido.**
Tabla 11 Caso de pruebas correspondiente al CU Crear Expediente. Descripción de las variables ... **¡Error! Marcador no definido.**

Índice de figuras

Figura 1 Interfaz principal de Procesos Penales Ordinarios. 29
Figura 2 Diagrama de despliegue de la FGR 34
Figura 3 Diagrama de despliegue en las fiscalías provinciales 34
Figura 4 Diagrama de despliegue de las fiscalías municipales 35
Figura 5 Diagrama de componentes del caso de uso Crear Expediente 36
Figura 6 Arquitectura Modelo Vista Controlador implementado por Symfony 40
Figura 7 Flujo de trabajo de Symfony según el patrón MVC 42
Figura 8 Interfaz de error mostrada por el navegador de página no encontrada. 43
Figura 9 Ejemplo de validación con JavaScript 44
Figura 10 Representación de la incidencia de los resultados de la evaluación 53
Figura 11 Representación de la incidencia de los resultados de la evaluación 54
Figura 12 Representación de la incidencia de los resultados de la evaluación 54
Figura 13 Representación en % de los resultados obtenidos en el instrumento 55
Figura 14 Representación de la incidencia de los resultados de la evaluación 55
Figura 15 Representación de la incidencia de los resultados de la evaluación 56
Figura 16 Representación de la incidencia de los resultados de la evaluación 56
Figura 17 Resultados de la aplicación de la herramienta PHP CodeSniffer. 57
Figura 18 Interfaz principal de la aplicación. 62
Figura 19 Interfaz del escenario Crear Expediente Ordinario. 63
Figura 20 Interfaz del escenario Buscar Expediente Ordinario 64
Figura 21 Interfaz del escenario Decisiones del fiscal 65
Figura 22 Interfaz del escenario Realizar Indicaciones. 66
Figura 23 Interfaz del escenario Historial del Expediente Ordinario 67
Figura 24 Interfaz del escenario Realizar reportes 68
Figura 25 Gráfica de los resultados de la evaluación de la métrica TOC 70
Figura 26 Gráfica de los resultados de la evaluación de la métrica RC. 71

Introducción

Desde la aparición del hombre, las personas han podido comunicar e intercambiar información de una u otra manera, ya sea por medio de señales, sonidos, gestos o movimientos, que le permiten al individuo formar parte de una sociedad y acogerse a ella. Con el inicio de la era de la tecnología de la información, el hombre ha logrado valiosos avances en cuanto a la rapidez de gestionar la información.

El vertiginoso avance de las Tecnologías de la Información y las Comunicaciones (TIC) día tras día afecta todos los campos de la sociedad de la información, desempeñando un papel fundamental en el proceso de transformación de la vida de las personas, de la economía, de la cultura, de la política y de la competitividad de cualquier país.

Por tales razones, desde hace varios años, Cuba ha destinado parte de sus esfuerzos hacia la informatización de su sociedad, con el propósito de insertarse en el marco tecnológico mundial y alcanzar mayor eficiencia en los procesos que se realizan en todas las esferas del país.

Como materialización de esta idea, surge la UCI[1]. Este instituto de nuevo tipo, tiene un modelo de formación de profesionales comprometidos con su país, que combina el estudio con la producción y la investigación. Entre sus propósitos se encuentra la informatización de la sociedad, motivo por el cual se le dio la tarea de informatizar el sector jurídico, específicamente la FGR[2], surgiendo así el proyecto Sistema de Gestión Fiscal.

La Fiscalía General de la República, de acuerdo con al artículo 127, capítulo XIII, de la Constitución de la República de Cuba, es el órgano del Estado al que corresponde, como objetivos fundamentales, el control y la preservación de la legalidad, sobre la base de la vigilancia del estricto cumplimiento de la Constitución, las leyes y demás disposiciones legales, por los organismos del Estado, entidades económicas y sociales y por los ciudadanos; y la promoción y el ejercicio de la acción penal pública en representación del Estado. (1)

La FGR es una organización de prestigio nacional que necesita perfeccionar la gestión de sus procesos y de toma de decisiones, eficiencia en el trabajo de los fiscales y mayor conformidad por parte de los

[1] ***UCI**: Universidad de las Ciencias Informáticas.*
[2] ***FGR**: Fiscalía General de la República.*

ciudadanos cubanos. El SGF[3] es precisamente un proyecto que está dedicado a informatizar todos los procesos que se desarrollan en las fiscalías del país. Está dividido en varios subsistemas, los cuales están estrechamente vinculados a los procesos que se ejercen actualmente en las fiscalías. Uno de ellos lo constituye el subsistema de los Procesos Penales, el cual pretende informatizar el proceso de la promoción y el ejercicio de la acción penal pública en representación del Estado. Este a su vez está conformado por varios módulos, dentro de los cuales se encuentra Ordinario, que comprende los delitos sancionables por más de 1 año de privación de libertad o multas mayores de 300 cuotas o ambas. Éste es uno de los procesos más complejos que se realiza en las fiscalías, por lo que genera grandes volúmenes de información sobre los casos que se investigan. Toda esta información es archivada como constancia del delito cometido y porque posteriormente puede ser utilizada como base para otras investigaciones. En ocasiones el acceso a la misma se torna un poco difícil debido a la forma en que es almacenada, por lo que requiere de un tiempo para su búsqueda que puede dilatar el proceso de investigación.

Existe un gran déficit de personal en las áreas de las fiscalías, por lo que cada fiscal, ente fundamental del trabajo de las fiscalías, atiende más de un caso a la vez y en consecuencia, debe estar pendiente de los plazos de vencimiento establecidos para cada uno de los procesos. La mayor parte del trabajo se realiza de manera manual, provocando menor control y supervisión por parte del nivel superior de la tramitación de los procesos en los órganos provinciales, municipales y central.

Actualmente, la FGR cuenta con un sistema informático que mejora la labor de los fiscales, pero no es suficiente para la gestión de toda la información que se maneja; por ello la necesidad de dotarlos de un sistema informático que les brinde una mayor ayuda en la toma de decisiones y en la organización del trabajo en la entidad.

Teniendo en cuenta lo expuesto anteriormente, se plantea el siguiente **problema a resolver**: Los procesos ordinarios que se desarrollan actualmente en la FGR se ven afectados en cuanto a control, seguimiento y tiempo, provocando una demora en la tramitación de los expedientes generados en las investigaciones, de aquí que el **objeto de estudio** sea el Proceso de Desarrollo de Software, encaminado a cumplir el **objetivo general** de desarrollar la implementación y prueba de una herramienta que facilite el control y la gestión de los procesos ordinarios, disminuyendo considerablemente el tiempo de tramitación de los expedientes. El **campo de acción** se centra en la Implementación y Pruebas del módulo Ordinario.

[3] ***SGF**: Sistema de Gestión Fiscal.*

En correspondencia con el problema y con la intención de darle cumplimiento al objetivo propuesto, se defiende la **idea** de que con la implementación y prueba del módulo Ordinario, perteneciente al subsistema Procesos Penales del proyecto Sistema de Gestión Fiscal, se agilizarán los procesos ordinarios que se llevan a cabo en la FGR, existiendo un mayor control por parte de los fiscales superiores del trabajo realizado.

Del objetivo general de la investigación se derivan los siguientes **objetivos específicos:**

- Conformar el marco teórico de la investigación.
- Implementar la solución propuesta.
- Validar los resultados correspondientes.

Para darle cumplimiento al objetivo trazado se han planteado las siguientes **tareas a realizar:**

- Búsquedas bibliográficas acerca de las herramientas, lenguajes de programación que se utilizarán para la solución del problema planteado.
- Estudio de los sistemas de gestión fiscal que existen en el Cuba y el mundo.
- Estudio de los requisitos funcionales detectados por los analistas.
- Estudio del modelo de casos de usos del sistema.
- Confección del modelo de implementación del modulo Ordinario.
- Implementación de los componentes identificados en el modelo de implementación.
- Realización de las pruebas de software correspondientes al módulo Ordinario.
- Análisis de los resultados obtenidos.

La realización de esta investigación traerá consigo impactos positivos tanto en las fiscalías del país como en los ciudadanos cubanos. Estos pueden ser:

A nivel de Fiscalía General de la república.

- Disponer de un sistema para gestionar centralmente los procesos de la FGR.
- Contribuir al aumento de la calidad de la tramitación de los procesos, reducción de términos en la realización de las actividades y diligencias practicadas.
- Contribuir a la distribución equitativa de la fuerza laboral.
- Lograr la supervisión y control en tiempo real de los procesos judiciales.

- Contribuir a la preservación de la información por un período mínimo de 30 años.

A nivel de Fiscales.

- Disponer de un sistema de ayuda a la toma de decisiones.
- Contribuir al aumento del rendimiento laboral.
- Aumento del cumplimiento de los términos establecidos.

A nivel de Trabajadores de las fiscalías.

- Rapidez en el procesamiento de la información, así como su envío y recepción para diferentes organismos y entidades.

A nivel de Ciudadano (a) cubano (a).

- Rapidez en sus tramitaciones legales.

Entre los **métodos de investigación científica** existentes fueron utilizados los **métodos teóricos:**

- ***Histórico–lógico:*** Su utilización posibilitó la realización de un estudio de las tendencias históricas y actuales a nivel nacional e internacional sobre el desarrollo de la informática y las comunicaciones, así como un análisis de las metodologías, herramientas y lenguajes utilizados en el desarrollo del sistema.
- ***Analítico-Sintético:*** Este método se utilizó para analizar y comprender los procesos ordinarios que transitan por la FGR y arribar a conclusiones sobre cómo sistematizar dichos procedimientos. Además se realiza un estudio del proyecto SGF en general, lo que posibilita una mayor comprensión del problema en concreto.
- **Sistémico:** Se tiene en cuenta el problema como un todo, con el propósito de desarrollar un sistema flexible y robusto que cumpla con los requisitos exigidos, que sea altamente configurable y adaptable.

Estructuración del contenido con una breve explicación de sus partes.

Capítulo 1: Fundamentación Teórica

Comprende el estado del arte sobre el tema tratado, haciendo énfasis en las tendencias actuales de las tecnologías de la información y las comunicaciones en el desarrollo del software. Se expone el marco

conceptual sobre el que gira la propuesta de solución. Se realiza un análisis de las herramientas y lenguajes a utilizar en el desarrollo del sistema.

CAPÍTULO 2: Desarrollo de la solución propuesta

En este capítulo se realizará la descripción y el análisis de la solución propuesta, especificando los estándares de código, el modelo de componentes, así como los patrones de arquitectura y diseño aplicados y la descripción de las principales funcionalidades a informatizar. En este capítulo se lleva a cabo la implementación.

CAPÍTULO 3: Validación de la solución propuesta

En este capítulo se validará la solución propuesta mediante pruebas de caja negra del software y se evaluará la solución haciendo un resumen de los principales resultados obtenidos. Además se aplicarán un conjunto de métricas que nos permitirán conocer las dependencias y relaciones entre clases, así como la complejidad y reutilización de las mismas.

Capítulo 1

Fundamentación Teórica

1.1. Introducción

En este capítulo se realiza un estudio del estado del arte de los sistemas de gestión fiscal que existen actualmente en el mundo y en nuestro país. Igualmente, se hace un exhaustivo estudio de las herramientas, metodologías y lenguajes que se utilizarán para el desarrollo de la solución de la propuesta.

1.2. Valoración del estado del arte

1.2.1. Situación de los Sistemas de Gestión Fiscal en el mundo

Todas las sociedades del mundo tienen que necesariamente acogerse a un conjunto de leyes arbitrarias, creadas por el propio hombre, con el objetivo de garantizar la estabilidad social y un comportamiento adecuado de la sociedad. A nivel mundial existen diferentes tipos de ordenamientos jurídicos o sistemas de derechos en dependencia de la peculiaridad de cada país. Para el desarrollo de esta investigación, se realizó un estudio de diferentes soluciones de gestión fiscal sustentadas en sistemas de derecho. Seguidamente se analizan algunas de ellas.

Lex-Doctor Estudios Jurídicos es un Sistema de Gestión Jurídica, desarrollado por Sistemas Jurídicos SRL[4], diseñado para cubrir la amplia gama de necesidades que van, desde la sencillez del esquema de un Estudio Jurídico unipersonal, hasta la compleja administración de grandes Asesorías Letradas de Bancos, Municipalidades, empresas y entidades que son susceptibles de poseer organizaciones jurídicas propias. Su tecnología de administración de datos, permite manejar grandes volúmenes de información, y organizar grupos de trabajo operando en redes de gran cantidad de terminales. (2)

Gedex se define como un software jurídico, con amplia experiencia en el sector y siendo uno de los más utilizados en España y Latinoamérica. Clasifica sus expedientes instantáneamente en función de múltiples

[4] ***SRL**: Empresa que desarrolla y comercializa exclusivamente sistemas de computación para ser aplicados a la actividad jurídica.*

aspectos (plazos, estado de apertura, etc.) Ofrece soporte para despachos que dispongan de una red local, permitiendo compartir toda la información y escritos en distintos ordenadores y delegaciones. Requiere un ordenador personal con alguno de los sistemas operativos Windows 7, Windows Vista, Windows XP, Server 2008 R2 y 2003, incluyendo versiones Home, Professional, Ultimate, Workstation, Server y Advanced Server. (3)

Este sistema está principalmente orientado hacia el manejo de los expedientes jurídicos, dejando atrás cuestiones importantes en el sector como son la toma de decisiones y una gestión más profunda de los procesos que se llevan a cabo en las fiscalías, por lo que no cubre las necesidades reales que actualmente poseen las mismas.

ABOGest es un programa para la gestión integral de despachos de abogados. Es el único software en España que ha obtenido una certificación oficial de un colegio de abogados, y ha sido diseñado, pensado y desarrollado por y para los abogados. Permite automatizar el trabajo diario controlando las citas pendientes o los vencimientos.

Con una dilatada experiencia en el mercado, es el único programa que cuenta con 10 años de vida y una evolución continua, desde la primera versión en MS-DOS con motor Paradox, hasta la última novedad con motor cliente-servidor bajo Windows XP y Server 2003. (4)

Este sistema tampoco es adecuado para la FGR, pues además de ser propietario, está más bien orientado hacia el trabajo personal de cada abogado, por lo que no permite relacionar los expedientes, imposibilitando el funcionamiento de la entidad como un todo.

1.2.2. Situación en Cuba

En Cuba se desarrolló una solución informática denominada SOFTLEX, en el seno de la Sociedad Cubana de Derecho e Informática, que da tratamiento a normativas y tiene un módulo de seguimiento de asuntos, que recoge una serie de datos útiles de los procesos.

Sin embargo hasta la fecha no se tiene conocimiento de la existencia de ninguna solución informática de envergadura y profundidad desarrollada sobre software libre con un grado de madurez elevado en la gestión de información y ayuda en la toma de decisiones a nivel nacional, por lo que se aprecia la necesidad latente de contar con un sistema informático para la gestión fiscal que de respuesta a esta problemática. (5)

1.2.3. Conclusión acerca de los Sistemas de Gestión Fiscal

Luego de realizado un estudio de los sistemas de los productos informáticos de gestión jurídica, se arribó a la conclusión que ninguno satisface las necesidades de las fiscalías de nuestro país debido a que:

- Poseen esquemas legislativos muy diferentes del esquema socialista establecido en Cuba, por lo que no es posible adaptar ninguno de estos software a nuestro sistema legislativo.
- Son privativos.
- Se centran principalmente en el almacenamiento de los expedientes y no van más allá de lo que realmente se realiza en la gestión de los procesos jurídicos.

Por tal motivo, surge la necesidad de desarrollar una aplicación que se ajuste al sistema judicial cubano.

1.3. Técnicas de Programación

1.3.1. Programación estructurada

La programación estructurada es una forma de escribir programas de ordenador (programación de computadora) de manera clara. Para ello utiliza únicamente tres estructuras: secuencia, selección e iteración; siendo innecesario el uso de la instrucción o instrucciones de transferencia incondicional (goto, exit function, exit sub o múltiples return). Los programas son fáciles de entender, ya que pueden ser leídos de forma secuencial. (6)

El principal inconveniente de este método de programación es que se obtiene un único bloque de programa, que cuando se hace demasiado grande puede resultar problemático su manejo, como es el caso del software que se desea implementar en cuanto a complejidad y tamaño del mismo.

1.3.2. Programación Orientada a Objetos

La POO[5] es un paradigma de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas informáticos. Está basado en varias técnicas, incluyendo herencia, abstracción, polimorfismo y encapsulamiento. Su uso se popularizó a principios de la década de los años 1990. Un objeto es una abstracción de algún hecho o ente del mundo real que tiene atributos que representan sus características o propiedades y métodos que representan su comportamiento o acciones que realizan.

[5] ***POO**: Programación Orientada a Objetos.*

Todas las propiedades y métodos comunes a los objetos se encapsulan o se agrupan en clases. Una clase es una plantilla o un prototipo para crear objetos, por eso se dice que los objetos son instancias de clases. Esto permite hacer los programas y módulo más fáciles de escribir, mantener y reutilizar. En la actualidad, existe variedad de lenguajes de programación que soportan la orientación a objetos. (7)

1.3.3. Justificación de la técnica de programación seleccionada

Luego de realizado un estudio de algunas de las técnicas de programación existentes en la actualidad, se selecciona para la realización de la presente investigación la POO. Esta técnica fomenta la reutilización del código, facilitando el trabajo en equipos. Además permite la creación de sistemas complejos y agiliza considerablemente el desarrollo de software.

1.4. Lenguajes de programación

1.4.1. Tecnologías del lado del servidor

1.4.1.1. Java

El lenguaje para la programación en Java, es un lenguaje orientado a objeto, de una plataforma independiente. Fue desarrollado por la compañía Sun Microsystems a principios de los años 90, con la idea original de usarlo para la creación de páginas web. Tiene muchas similitudes con el lenguaje C y C++, pero contiene 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. Java se considera un software de distribución libre y de fácil aprendizaje. (8) (32)

Independientemente a todas las potencialidades que posee Java, puede ser un lenguaje de ejecución lenta, debido al uso de la maquina virtual de Java (ésta consume muchos recursos), además el uso de un recolector de basura para eliminar de forma automática aquellos objetos no requeridos, añade una sobrecarga que puede afectar al rendimiento. Esto es en parte debido a las frecuentes declaraciones de tipos y conversiones de tipo manual.

1.4.1.2. Perl

Perl es un lenguaje diseñado por Larry Wall en 1987, que originalmente fue desarrollado para ser un lenguaje de manipulación de texto, sin embargo con el pasar de los años se ha formado una verdadera

comunidad de personas que utilizan Perl para el desarrollo de GUI[6], desarrollo de páginas web, administración de sistema, programación en red, entre otras aplicaciones. Por todo esto, Perl es un lenguaje muy utilizado en los dos campos siguientes:

- *La administración de sistemas operativos*. Debido a sus características Perl es muy potente en la creación de pequeños programas que pueden ser usados como filtros para obtener información de ficheros, realizar búsquedas, entre otros.
- *La creación de formularios en la Web.* Es decir, se utilizan para la creación de scripts CGI[7].

La principal desventaja de Perl se encuentra en el tiempo de ejecución de un programa, ya que un programa Perl es compilado cada vez que se ejecuta, por lo que puede resultar más lento que un programa similar escrito en otro lenguaje. Sin embargo, se han implementado técnicas para mejorar esta situación como guardar el compilado del programa en memoria y retrasar la compilación hasta que sea necesitada.

1.4.1.3. Active Server Pages

ASP[8] es una solución de Microsoft basada en Visual Basic. La principal ventaja de ASP es que hay un flujo constante de trabajo para estos desarrolladores. Sin embargo, se debe tomar esta información con cautela pues las tendencias actuales pronostican un decremento de los servidores de Microsoft. Además ASP es un sistema con nula portabilidad pues requiere necesariamente de un servidor Windows, con todas las implicaciones de alto costo, poca flexibilidad y escasa seguridad que estos equipos conllevan.
ASP nos provee de una gran cantidad de componentes y objetos con los cuales se puede manejar de manera sencilla la interacción del navegador y el servidor Web. (9)

1.4.1.4. PHP[9]

PHP es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. Está disponible para la mayoría de sistemas operativos existentes, desde Unix, Linux, Microsoft Windows, MAC, entre otros. Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.

---

[6] ***GUI**: Graphics Users Interfaces (en español Interfaces Gráficas de Usuario)*
[7] CGI: Common Gateway Interface (en español Interfaz de entrada común).
[8] ***ASP**: Active Server Pages*
[9] ***PHP**: Es un acrónimo recursivo que significa PHP Hypertext Pre-processor.*

El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable. Permite aplicar técnicas de programación orientada a objetos y no requiere definición de tipos de variables, lo cual beneficia al programador en la implementación. Posee una amplia documentación en su página oficial.[3] Soporta la programación orientada a objetos a partir de su versión 5. (10) (11)

Otras características del lenguaje.

- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad: Adabas D, LDAP, Microsoft SQL server, MySQL, ODBC, Oracle, PostgreSQL, Solid, Sybase.
- Soportado por una gran comunidad de desarrolladores, como producto de código abierto.
- Integración con varias bibliotecas externas, permite generar documentos en PDF (documentos de Acrobat Reader) hasta analizar código XML.

1.4.1.5. Consideraciones de la selección del lenguaje de programación

Teniendo en cuenta las características de los lenguajes de programación descritos anteriormente y de la aplicación que se desea desarrollar, se ha decidido utilizar PHP en su versión 5.2.5. Esta elección está sujeta a que el desarrollo de un proyecto en PHP será siempre mucho menor en coste y tiempo que utilizando Java, debido su Máquina Virtual. A esto se debe también que la velocidad de ejecución es mucha más rápida y sencilla en el caso de PHP. Es sumamente escalable, considerando *escalable* como la capacidad de un sistema de aumentar el número de sus usuarios aumentando sus recursos y sin perder ninguna de sus ventajas.

1.4.2. Tecnologías del lado del cliente

1.4.2.1. HTML[10]

HTML es un lenguaje de composición de documentos y especificación de ligas de hipertexto que define la sintaxis y coloca instrucciones especiales que no muestra el navegador, aunque si le indica cómo desplegar el contenido del documento, incluyendo texto, imágenes y otros medios soportados.

[10] **HTML**: Hyper Text Markup Language (en español Lenguaje para Marcación Hipertexto)

Una de las características esenciales de este lenguaje es la universalidad, y significa que prácticamente cualquier ordenador, independientemente del sistema operativo, puede leer o interpretar una página web. Sin embargo, el aspecto de dichas páginas depende del tipo de ordenador, del monitor, la velocidad de la conexión de Internet y, por último del navegador. Aunque no todos los navegadores muestran una misma página web de la misma forma. (12)

1.4.2.2. JavaScript

JavaScript es un lenguaje de programación ligero y orientado a objetos. El corazón del lenguaje se embebe en los navegadores para interpretar los códigos (scripts) que escribimos en las páginas. Con este lenguaje script se pueden generar páginas dinámicamente en función de las preferencias del usuario, validar los datos introducidos en un formulario o modificar dinámicamente el contenido de la página. Fue creado por Netscape para su navegador 2.0 con la versión 1.0 de JavaScript y ya marcha por la 1.5; está basado en objetos. JavaScript es manejado por eventos, independiente de cualquier plataforma, permite desarrollo rápido y muy fácil de aprender. No incluye complejas reglas sintácticas. (14)

1.4.2.3. CSS[11]

CSS es un lenguaje de hojas de estilos creado para controlar la presentación de los documentos electrónicos definidos con HTML. Es la mejor forma de separar los contenidos y su presentación y es imprescindible para la creación de páginas web complejas. Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página. (13)

Todos los lenguajes descritos en este acápite serán utilizados para la realización de la aplicación, HTML para la composición de la información en la página, JavaScript para las validaciones y CSS para el estilo que se les dará a la misma.

1.5. Herramientas

1.5.1. Marco de trabajo

Un framework o marco de trabajo es una estructura de soporte definido, mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas,

[11] ***CSS:*** *Cascade Style Sheet (en español Hoja de Estilo en Cascada)*

bibliotecas y un lenguaje interpretado para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Ofrecen una infraestructura que permite a los desarrolladores tener un código más ordenado, limpio y fácil de actualizar, un código más seguro y robusto y mucho más eficiente.

1.5.1.1. CodeIgniter

Es un entorno de desarrollo abierto que permite crear webs dinámicas con PHP, ayudando a realizar proyectos de forma mucho más rápida, sin tener que crear toda la estructura desde cero. Esto se debe a que dispone de un conjunto bastante amplio de librerías para realizar tareas comunes, así como una interface simple y una estructura lógica sencilla.

Es un entorno muy simple. El núcleo del sistema requiere muy pocas librerías para funcionar adecuadamente. Las librerías adicionales que se necesiten se cargan de forma dinámica, con lo cual el sistema es muy sencillo y muy rápido. (15)

1.5.1.2. Zend Frameworks

Zend Framework es un framework de código abierto para desarrollar aplicaciones web y servicios web con PHP 5. La estructura de sus componentes es algo único; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado. Entre estos componentes se encuentran: Zend_Config para temas de configuración de aplicaciones web, Zend_Db para tratar con bases de datos, Zend_Search o Zend_Feed entre otros.

Zend Framework ofrece un gran rendimiento y una robusta implementación del patrón Modelo Vista Controlador, una abstracción de base de datos fácil de usar, validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos. (16)

1.5.1.3. CakePHP

CakePHP es un marco de desarrollo rápido para PHP, libre, de código abierto. Se trata de una estructura que sirve de base a los programadores para que éstos puedan crear aplicaciones Web. El desarrollo web con CakePHP ofrece las herramientas para empezar a escribir el código que realmente se necesita la lógica específica de la aplicación. Consume muchos recursos de la base de datos. (17)

Algunas características del framework:

- Licencia flexible.
- Compatible con PHP4 y PHP5.
- Generación de código.
- Arquitectura Modelo Vista Controlador.
- Componentes de Email, Cookie, Seguridad, Sesión y Manejo de solicitudes.
- Listas de control de acceso flexibles.

1.5.1.4. Symfony

Symfony es un completo framework diseñado para optimizar gracias a sus características, el desarrollo de las aplicaciones web. Separa la lógica del negocio, la lógica del servidor y la presentación de la aplicación web según el patrón Modelo Vista Controlador. Symfony ha tomado las mejores ideas del framework Rails y de muchos otros más, ha incorporado ideas propias y el resultado es un marco de trabajo elegante, estable, productivo y muy bien documentado.

Symfony se diseñó para que se ajustara a los siguientes requisitos:

- Fácil de instalar y configurar en la mayoría de plataformas y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares.
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de los casos pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de “convenir en vez de configurar”, en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de las “mejores prácticas” y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros. (18) (19)

1.5.1.5. Justificación del framework seleccionado

El framework seleccionado fue Symfony en su versión 1.3, pues es el que más se ajusta a las características y condiciones de la aplicación que se desea desarrollar. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Permite la rápida creación de aplicaciones web, con una alta fidelidad en cuanto a seguridad en las mismas, la cual puede ser a nivel de aplicación, de módulos o de acciones específicas. Este framework trae implementado algunos patrones de diseño como es el Decorador usado para las plantillas y el patrón arquitectónico MVC[12]. Automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

1.5.2. Servidor WEB

1.5.2.1. Internet Information Server

IIS[13] es un servidor web y un conjunto de servicios para el sistema operativo Microsoft Windows. Originalmente era parte del Option Pack para Windows NT. Luego fue integrado en otros sistemas operativos de Microsoft destinados a ofrecer servicios, como Windows 2000 o Windows Server 2003. Windows XP Profesional incluye una versión limitada de IIS. Los servicios que ofrece son: FTP, SMTP, NNTP y HTTP/HTTPS.

Este servicio convierte a un computador en un servidor web para Internet o una intranet, es decir que en las computadoras que tienen este servicio instalado se pueden publicar páginas web tanto local como remotamente. Los servicios de IIS proporcionan las herramientas y funciones necesarias para administrar de forma sencilla un servidor web seguro. El servidor web se basa en varios módulos que le dan capacidad para procesar distintos tipos de páginas. Por ejemplo, Microsoft incluye los de ASP y ASP.NET. También pueden ser incluidos los de otros fabricantes, como PHP o Perl.

1.5.2.2. Apache

Es un servidor Web potente, flexible y disponible para distintas plataformas y entornos. Es altamente configurable y de diseño modular, lo que posibilita que los administradores de los sitios Web puedan elegir los módulos que serán incluidos y ejecutados en el servidor. (20)

Características:

[12] ***MVC**: Model View Controller (en español Modelo Vista Controlador).*
[13] ***IIS**: Internet Information Server.*

- Apache es una tecnología gratuita de código fuente abierta. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto.
- Apache trabaja con Perl, PHP y otros lenguajes de script. También trabaja con Java y páginas jsp, teniendo todo el soporte que se necesita para tener páginas dinámicas.
- Apache te permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.
- Es altamente configurable en la creación y gestión de logs. Apache permite la creación de ficheros de log a medida del administrador, de este modo puedes tener un mayor control sobre lo que sucede en tu servidor. (21)

1.5.2.3. Justificación de la selección del servidor web

Debido a potencialidades que brinda el servidor web Apache, y su arquitectura modular que permite construir un servidor hecho a la medida, se decidió que sería el servidor web que se utilizará para la realización de la aplicación. Éste, a diferencia del ISS, se encuentra bajo la licencia de GNU Linux, lo cual lo hace mas adaptable y configurable. En cuanto a la administración los archivos de configuración de Apache están en ASCII, por lo que tiene un formato simple, y pueden ser editados tan solo con un editor de texto. Es uno de los servidores más utilizados en el mundo puesto que es tan potente como flexible.

1.5.3. Sistema de Gestión de Base de Datos

1.5.3.1. MySQL[14]

MySQL es un sistema de administración de bases de datos para bases de datos relacionales. Así, MySQL no es más que una aplicación que permite gestionar archivos llamados de bases de datos. Es un sistema de gestión de base de datos relacional, multihilo y multiusuario. (22)

Las principales características de este gestor de bases de datos son las siguientes:.

- Soporta gran cantidad de tipos de datos para las columnas.
- Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc).
- Gran portabilidad entre sistemas.
- Soporta hasta 32 índices por tabla.

[14] ***MySQL:*** *Del inglés My Structured Query Language (en español Lenguaje de Consulta Estructurado)*

- Gestión de usuarios y contraseñas, manteniendo un muy buen nivel de seguridad en los datos. (33)

1.5.3.2. PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD[15] y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado

Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema (23)

Características principales:

- Es una base de datos completamente ACID[16] Complaint.
  - *Atomicidad*: es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.
  - *Consistencia*: es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.
  - *Aislamiento*: es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que la realización de dos transacciones sobre la misma información sean independientes y no generen ningún tipo de error.
  - *Durabilidad*: es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.
- Posibilita la integridad referencial, lo cual garantiza la coherencia de datos entre relaciones aparejadas.
- Documentación muy bien organizada, pública y libre, con comentarios de los propios usuarios.

[15] ***BSD**: (Berkeley Software Distribution). Es la licencia de software otorgada principalmente para los sistemas BSD. La licencia BSD permite el uso del código fuente en software no libre.*
[16] ***ACID**: Atomicity, Consistency, Isolation and Durability.*

- Numerosos tipos de datos y posibilidad de definir nuevos tipos. Además de los tipos estándares en cualquier base de datos, tenemos disponibles, entre otros, tipos geométricos, de direcciones de red y de cadenas binarias entre otras.
- Soporta el almacenamiento de objetos binarios grandes (gráficos, videos, sonido,...). (31)

1.5.3.3. Justificación del gestor de base de datos seleccionado

Resulta indiscutible las grandes ventajas que presentan los gestores de base de datos descritos anteriormente, pero aun así, se ha decidido utilizar PostgreSQL en su versión 8.4, pues este ofrece una garantía de integridad de datos mucho más fuerte que MySQL y se ajusta más a aquellos escenarios en los cuales no se puede permitir que se corrompa o se pierda ni un solo registro, como es el caso de la aplicación que se va a desarrollar. Aunque PostgreSQL sea más lento respondiendo a una única consulta, presenta una mejor escalabilidad y rendimiento bajo grandes cargas de trabajo. La principal ventaja que tiene éste gestor sobre MySQL es que es totalmente gratuito bajo las licencias de GNU. Será gestionado por una aplicación gráfica llamada PGAdmin III que es una de las más completas y populares con licencia Open Source. Es capaz de gestionar versiones a partir de PostgreSQL 7.3, ejecutándose en cualquier plataforma. Posee una interfaz gráfica que soporta todas las características de PostgreSQL y facilita enormemente la administración. Es considerada la base de datos de código abierto más avanzada y potente del mundo. Además la UCI forma parte de la sociedad latinoamericana de este gestor, lo cual permite la socialización del conocimiento sobre el mismo.

1.5.4. Entornos de Desarrollo Integrado

IDE[17] es un editor de código que además puede servir para depurar y facilitar las diferentes tareas necesarias en el desarrollo de cualquier tipo de aplicación, para de esta manera agilizar el trabajo del desarrollador.

1.5.4.1. Zend Studio

Se trata de un programa de la casa Zend, orientado a desarrollar aplicaciones web en PHP. El programa, además de servir de editor de texto para páginas PHP, proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración de código. Su licencia es cara. (24)

[17] ***IDE**: Integrated Development Environment (Entorno de Desarrollo Integrado en español)*

1.5.4.2. Eclipse PDT

Es un excelente IDE para PHP, basado en el popular IDE Eclipse, Este es uno de los primeros OpenSource para PHP. El proyecto PDT proporciona un marco de herramientas de desarrollo de PHP para la plataforma Eclipse. Este proyecto abarca todos los componentes de desarrollo necesarios para el desarrollo de PHP y facilitar la extensibilidad. Es independiente de la plataforma.

1.5.4.3. Netbeans

El IDE NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. Es un producto libre y gratuito sin restricciones de uso, de código abierto escrito completamente en Java usando la plataforma NetBeans. A partir de su versión 6.8, añade nuevas funcionalidades específicamente para PHP como es el potente debugger integrado y soporte para Symfony, un gran Framework MVC y para AJAX. (25)

1.5.4.4. Justificación del IDE seleccionado

De los IDE analizados anteriormente, Zend Studio queda descartado por lo caro que resulta la compra de sus licencias. Sin embargo entre Eclipse PDT y Netbeans, ambos libres y gratuitos, es más conveniente el uso del Netbeans en su versión 6.9 debido a que el mismo se integra con Symfony, el framework seleccionado anteriormente para el desarrollo de la aplicación, por lo que les facilitaría el trabajo a los programadores, que se evitarían tener que trabajar desde la consola para ejecutar comandos de Symfony propiamente dicho y la configuración que conlleva el mismo.

1.5.5. Sistemas de control de versiones

Un sistema de control de versiones es un software que administra el acceso a un conjunto de ficheros, y mantiene un historial de cambios realizados. El control de versiones es útil para guardar cualquier documento que cambie con frecuencia, como una novela, o el código fuente de un programa. Normalmente consiste en una copia maestra en un repositorio central, y un programa cliente con el que cada usuario sincroniza su copia local.

1.5.5.1. CVS[18]

CVS es un sistema de control de versiones que tiene como objetivo almacenar el registro de todo el trabajo y los cambios en los ficheros (código fuente principalmente) que forman un proyecto y permitir que distintos desarrolladores (inclusive situados a gran distancia) colaboren. CVS se ha hecho popular en el mundo del software libre. Sus desarrolladores difunden el sistema bajo la licencia GPL. Utiliza una arquitectura cliente-servidor: un servidor guarda la(s) versión(es) actual(es) del proyecto y su historial. Los clientes se conectan al servidor para sacar una copia completa del proyecto. Presenta el inconveniente que los archivos en el repositorio sobre la plataforma CVS no pueden ser renombrados, estos deben ser agregados con otro nombre y luego eliminados.

1.5.5.2. Subversion

Subversion (SVN) es un sistema de control de versiones que maneja los archivos y las carpetas de un proyecto y sus modificaciones en el transcurso del tiempo. Fue creado en el año 2000, con el objetivo de ser una alternativa real y mejorada a CVS. Al igual que otros proyectos exitosos de renombre, SVN es libre y Open Source, ya que está distribuido bajo la licencia Apache. Trabaja replicando el modelo cliente/servidor, es decir uno o más clientes se conectan a un servidor central que tiene la última copia del proyecto, como también copias de sus versiones anteriores. (26)

1.5.5.3. Justificación del control de versiones seleccionado

Subversion en su versión 1.5.1 posee grandes ventajas sobre CVS. El primero presenta una fuerte integración con Apache, servidor web seleccionado para la realización de la aplicación y una total transparencia al eliminar o cambiar nombres de archivos. Además, Subversión es capaz de mantener un control diferencial sobre cualquier archivo binario del depósito así reduciendo el consumo de espacio, esto contrastado con CVS que requiere archivar copias completas de un archivo binario cada vez que éste cambia.

1.5.6. Pruebas de calidad

1.5.6.1. Pruebas de Caja blanca

La prueba de caja blanca, denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de

[18] **CVS:** Del inglés Concurrent Versions System

prueba. Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que:

- Garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo;
- Ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa.
- Ejecuten todos los bucles en sus límites y con sus límites operacionales.
- Ejerciten las estructuras internas de datos para asegurar su validez.

Una de las técnicas de prueba de caja blanca es:

- *Pruebas de caminos básicos*: Es una técnica propuesta inicialmente por Tom McCabe, la cual le permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizarán que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. Algunos elementos y conceptos utilizados alrededor de éste método son los siguientes:
  - *Grafo de flujo o grafo del programa:* Representa el flujo de control lógico de un programa y se utiliza para trazar más fácilmente los caminos de éste. (Cada nodo representa una o más sentencias procedimentales y cada arista representa el flujo de control)
  - *Complejidad ciclomática:* Es una métrica de software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto de las pruebas, el cálculo de la complejidad ciclomática representa el número de caminos independientes del conjunto básico de un programa. Esta medida ofrece al probador de software un límite superior para el número de pruebas que debe realizar para garantizar que se ejecutan por lo menos una vez cada sentencia.
  - *Camino independiente:* Cualquier camino del programa que introduce, por lo menos, un nuevo conjunto de sentencias de proceso o una nueva condición.
  - *Pruebas de flujo de datos:* Selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.

    (27)

1.5.6.2. Pruebas de Caja Negra

Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software; o sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de caja negra no es una alternativa a las técnicas de prueba de caja blanca. Más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores que los métodos de caja blanca no detectan. La prueba de caja negra intenta encontrar errores de las siguientes categorías:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.

Algunas de las técnicas empleados en las pruebas de caja negra son:

- *Métodos de prueba basados en grafos*: En este método se debe entender los objetos (objetos de datos, objetos de programa tales como módulos o colecciones de sentencias del lenguaje de programación) que se modelan en el software y las relaciones que conectan a estos objetos. Una vez que se ha llevado a cabo esto, el siguiente paso es definir una serie de pruebas que verifiquen que todos los objetos tienen entre ellos las relaciones esperadas. En este método:
  - Se crea un grafo de objetos importantes y sus relaciones.
  - Se diseña una serie de pruebas que cubran el grafo de manera que se ejerciten todos los objetos y sus relaciones para descubrir error.
- *Partición equivalente:* Presenta la partición equivalente como un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. Una clase de equivalencia representa un conjunto de estados válidos o no válidos para condiciones de entrada. Típicamente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica.

- Análisis de Valores Límites: Esta técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
  (27)

1.5.6.3. Justificación de la selección de la prueba de calidad

Para la validación del sistema que se obtendrá, se decidió realizar pruebas de Caja Negra, ya que las mismas se centran en los requisitos funcionales que debe cumplir el software, asegurando de esa manera el correcto funcionamiento de la aplicación, aplicando la técnica de partición de equivalencia. Para complementar estas pruebas se aplicaran además algunas métricas orientadas al tamaño de las clases y las relaciones entre estas para medir atributos de calidad como reutilización, complejidad de mantenimiento, responsabilidad entre otros.

1.6. Conclusiones parciales

Con el estudio de varios sistemas informáticos dedicados a la Gestión Jurídica en Cuba y el mundo se llega a la conclusión de que ninguno de los software analizados anteriormente satisface a plenitud las necesidades que actualmente tienen las fiscalías del país, razón por la cual se decidió desarrollar una aplicación que brinde facilidades a los usuarios y que gestione los procesos que se desarrollan en las mismas. En la confección de la misma se utilizará:

- Como lenguaje de programación PHP 5.2.5
- Framework de desarrollo Symfony en su versión 1.3
- Como servidor Web Apache 2.2
- Sistema Gestor de Base de Datos PostgreSQL 8.4.
- Netbeans 6.9 como IDE de desarrollo.
- Para el control de versiones el Subversion 1.5.1

Capítulo 2
Características del Sistema

2.1. Introducción

En este capítulo se tratan los elementos y características relaciones con la implementación del sistema. Se realiza una valoración de la propuesta del análisis del sistema exponiendo los requisitos funcionales y no funcionales detectados durante la fase de levantamientos de requisitos. Se muestra como está concebida la arquitectura y las posibilidades proporcionados por el marco de trabajo, con el objetivo de brindar una mayor comprensión del funcionamiento de los componentes implementados. Además se realizan las descripciones de clases y operaciones utilizadas y por último se analizan la complejidad de algoritmos no triviales estimados entre los más importantes dentro del sistema.

2.2. Flujo del proceso

2.2.1. Valoración crítica de los artefactos propuestos por los analistas

En el proceso de desarrollo de software, la descripción de los requisitos del sistema es una cuestión de gran importancia pues estos facilitan un mayor entendimiento de las necesidades del cliente final y posibilitan una mejor identificación de las clases y funcionalidades que serán implementadas.

Un requerimiento según la IEEE Standard Glossary of Software Engineering Terminology se puede definir como una:

1. Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
2. Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.
3. Una representación documentada de una condición o capacidad como en 1 o 2.

Los requisitos se pueden clasificar en funcionales y no funcionales.

- *Requerimientos funcionales*: Son capacidades o condiciones que el sistema debe cumplir.
- *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 y/o confiable.

2.2.1.1. Requisitos funcionales

Los requisitos funcionales descritos por los analistas y que posibilitaron la identificación de las clases y las funcionalidades a implementar son un total de 55. De ellos 39 pertenecen a casos de uso con prioridad Crítico y el resto Secundario. En los artefactos adjuntos al documento de tesis, se puede encontrar la planilla de Especificación de Requisitos del módulo Ordinario.

2.2.1.2. Requisitos no funcionales

Las cualidades que debe presentar el software que se desea implementar son:

1. **Usabilidad**
   - El software brindará una ayuda para cualquier tipo de usuario, lo que le permitirá un avance considerable en la explotación de la aplicación en todas sus funcionalidades.
   - Existirán servidores locales con capacidad necesaria para el procesamiento de las solicitudes del conjunto de aplicaciones de las diferentes oficinas.
   - Las aplicaciones siempre solicitarán los datos a través del servidor local.
   - Desde cada servidor local se establecerá la conexión con servidores centrales para mantener la actualización de los datos en ambos sentidos.
2. **Fiabilidad**
   - El sistema estará disponible 24 horas al día, 7 días a la semana.
   - Disponibilidad de los casos asignados desde cualquier parte del país.
3. **Eficiencia**
   - Tiempo de respuesta promedio de las peticiones que se realizan al servidor no deberá ser mayor de 3 segundos.
4. **Soporte**
   - Soporte para grandes volúmenes de datos y velocidad de procesamiento.
   - Tiempo de respuesta rápido en accesos concurrentes.
   - El sistema debe ser multiplataforma.
5. **Requisitos de Apariencia o Interfaz Externa**
   - Diseño sencillo, con pocas entradas, permitiendo que no sea necesario mucho entrenamiento para utilizar el sistema.
   - El sistema tiene que ofrecer una interfaz amigable y fácil de operar.

- El sistema tiene que mantener la línea de diseño establecida para la institución que mantiene la uniformidad y representatividad de la misma.

6. Requisitos de Seguridad

- Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.
- El sistema debe mantener en todo momento la seguridad de la información asegurando la autenticidad para acceder a la misma.
- La seguridad se establecerá por roles que se le asignarán a los usuarios que interactúen con el sistema.
- El software brindará solamente aquellas funcionalidades que competen a la Unidad Ejecutora donde esté implantado.
- El sistema mantendrá en todo momento las trazas que se corresponden con las diferentes situaciones críticas que se puedan ocurrir.

7. Confiabilidad

- La herramienta de implementación a utilizar tiene soporte para recuperación ante fallos y errores.

8. Funcionalidad

- Mínima cantidad de páginas para ejecutar todas las funciones posibles (preferentemente que estén relacionadas).

9. Implantación

- Entregar toda la documentación asociada al proyecto.
- Organizar el adiestramiento de los usuarios.

10. Hardware

- **PC Cliente**
  - Computadora cliente de 128 Mb de memoria RAM o superior.
  - Computadora cliente de 40 Gb de disco duro o superior.
  - Pentium a 200 MHz de velocidad de procesamiento o superior.
  - Tarjeta de red.
  - El sistema tiene que interactuar con dispositivos de impresión.
  - El sistema tiene que interactuar con dispositivos de escaneo.
- **Servidores**

- Se debe contar con un servidor que brinde las siguientes funciones integradas: DNS, DHCP, Firewall.
- Se debe contar con un servidor de correo independiente.
- Se debe contar con un servidor independiente LDAP.
- Se debe contar con un servidor PostgreSQL y FTP.

11. Software

- **Cliente**
  - Debe poseer un cliente de servidor.
  - Debe tener el sistema operativo Debian o Windows.
- **Servidor**
  - Debe poseer servicio de directorio activo.

2.2.2. Propuesta del sistema

Para llevar a cabo los procesos ordinarios de las fiscalías, el sistema inicialmente debe permitir crear un expediente ordinario, especificando los datos del mismo que lo identificarán en la aplicación. Una vez introducidos estos valores, se puede ejecutar distintas acciones sobre dicho expediente en dependencia de los permisos y privilegios del usuario autenticado, partiendo de realizar una búsqueda sobre el mismo, donde se podrán especificar diferentes criterios que pueden ser número del expediente, provincia, municipio, fiscal que atiende el caso, procedencia del expediente[19] y/o algunos datos específicos del (de los) acusado(s), que arrojarán como resultado todos los documentos que coincidan con los parámetros especificados.

Una de las acciones que se pueden ejecutar, y que solo estará permitida para los fiscales superiores, es realizar una indicación, donde el sistema permite hacerle alguna observación al fiscal que atiende el proceso. En otro caso el fiscal superior puede reasignar el expediente a otro fiscal si fuese necesario.

Otra acción importante que se realiza en el sistema son las decisiones del fiscal, las cuales constituyen el grueso y la constancia del proceso de investigación llevado a cabo por el fiscal. Son en su totalidad 37 documentos que puede emitir el fiscal, de los cuales ya se tiene predefinido el formato y los datos

[19] *Procedencia del expediente se refiere al Órgano de Instrucción o Unidad de la PNR a la que pertenece el mismo.*

generales son cargados automáticamente, teniendo el fiscal sólo que detallar la información especifica del caso en cuestión. Para todos los documentos se brinda la posibilidad de impresión una vez creados.

Otra de las opciones que el sistema permite realizar es mostrar el historial de acciones que ha tenido lugar sobre el expediente desde su creación, dígase indicaciones, decisiones del fiscal, prórrogas, incluyendo también los datos generales del mismo como son número de expediente, procedencia, fecha de vencimiento, provincia y municipio al que pertenece, datos de los acusados y el fiscal actuante sobre dichas acciones.

Una vez concluido el plazo de vencimiento del expediente y en caso de que el fiscal no haya concluido las investigaciones, el sistema le permitirá solicitar prórroga, la cual podrá ser autorizada por su fiscal jefe, posponiendo entonces la fecha de cierre del mismo.

El sistema también permitirá obtener reportes acerca de la información almacenada en el sistema, la cual le servirá principalmente al fiscal para la toma de decisiones.

Todas las acciones descritas anteriormente podrán ser llevadas a cabo en dependencia de los permisos que tenga el fiscal que se encuentre autenticado en la aplicación. Desde el momento en punto en que cierre un expediente, sólo se podrá visualizar el historial del mismo.

A continuación se muestran la interfaz principal del módulo Ordinario. A partir de la misma es que se accede a todas las funcionalidades que brinda la aplicación. Para ver las demás interfaces del sistema ver del Anexo # I al Anexo #VII.

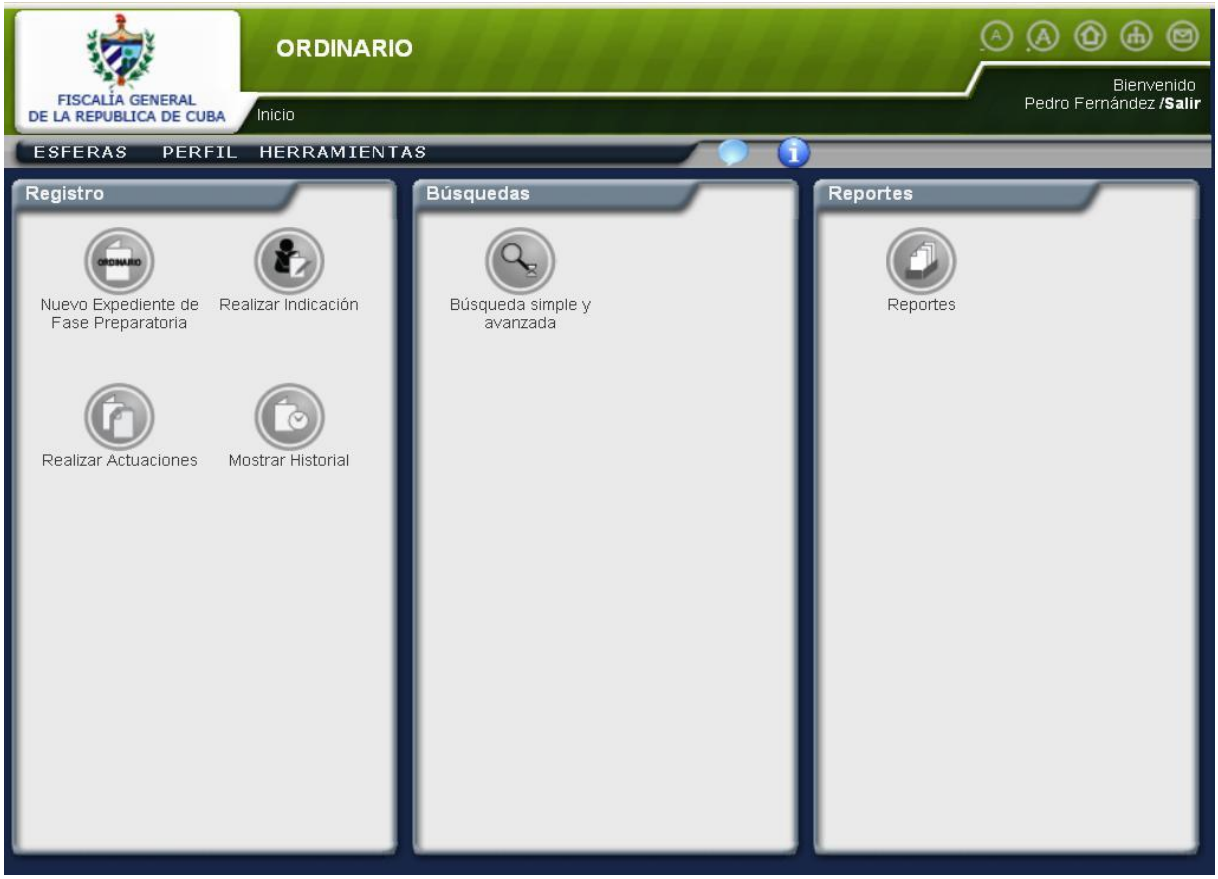

Figura 1 Interfaz principal de Procesos Penales Ordinarios.

2.3. Descripción de la implementación

2.3.1. Modelo de Datos

Un modelo de datos es una definición lógica, independiente y abstracta de los objetos, operadores y demás que en conjunto constituyen la máquina abstracta con la que interactúan los usuarios. Los objetos nos permiten modelar la estructura de los datos. Los operadores nos permiten modelar su comportamiento. (28)

El modelo de datos correspondiente al módulo consta con un total de 63 tablas, de ellas 12 nomencladores y las 51 restantes para el almacenamiento de la información correspondiente al flujo de trabajo del módulo. Para su construcción se tuvo en cuenta la reducción a la mínima expresión de los campos nulos y la persistencia de campos resúmenes para agilizar recuperaciones frecuentes de algunos

datos que son complejos de calcular. El diagrama correspondiente se encuentra en los artefactos adjuntos al documento.

2.3.2. Estándares de codificación

2.3.2.1. Estándares de codificación para PHP

- La indentación debe ser a cuatro espacios sin caracteres de tabulación. Esto es debido a que ciertos IDE's de desarrollo introducen caracteres de tabulación cuando indentan un texto automáticamente.
- Las estructuras de control deben tener un espacio entre el **keyword** de la estructura y el signo de apertura de paréntesis para distinguir entre las llamadas de las funciones y el signo de llaves debe estar sobre la línea de la estructura.
- Las funciones deben ser llamadas sin espacios entre el nombre de la función, el signo de paréntesis y el primer parámetro; espacios entre cada coma por parámetro y sin espacios entre el ultimo paréntesis, el signo de paréntesis cerrado y el signo de punto y coma (**;**).
- El estilo de los comentarios debe ser como el estilo de comentarios para C (/* */ ó //), no debe de utilizarse el estilo de comentarios de Perl (#).
- Cuando se incluya un archivo de dependencia incondicionalmente utilice **require_once** y cuando sea condicionalmente, utilice **include_once**.
- Siempre utilice las etiquetas <?php?> para abrir un bloque de código. No utilice el método de etiquetas cortas, porque esto depende de las directivas de configuración en el archivo *php.ini* y hace que el script no sea tan portable.
- Los nombres de las clases deben de iniciar con letra mayúscula. Los nombres de las variables y de las funciones pueden iniciar con letra minúscula, pero si estas tienen más de una palabra, cada nueva palabra debe iniciar con letra mayúscula (el nombre puede escribirse separado por signos de guión mayor). Si una función, en una clase, es privada; deberá comenzar con el signo de guión mayor para una fácil identificación. Las constantes deben de escribirse siempre en mayúsculas y tanto estas como las variables globales deben de tener como prefijo el nombre de la clase a la que pertenecen.
- Los archivos con código PHP, deben de ser guardados en formato ASCII utilizando la codificación ISO-8859-1. El formato ASCII con codificación ISO-8859-1, es el formato en que

se guardan los archivos de texto plano (.txt). La razón de este estándar es que determinados editores HTML (en especial Dreamweaver), agregan códigos de carácter extraño de salto de línea (como si se tratara de un archivo binario) y esto puede ocasionar que el intérprete de PHP, encuentre problemas a la hora de leer el script.

2.3.2.2. El sangrado

El uso de la sangría debe ser de 4 espacios sin los caracteres de la etiqueta. Deben configurarse editores para tratar las etiquetas como los espacios para prevenir inyección de caracteres de la etiqueta en el código de la fuente.

2.3.2.3. La longitud de la línea máxima

La longitud de la línea designada es de 85 caracteres. Sin embargo, las líneas más largas son aceptables hasta un máximo de 120 caracteres.

2.3.2.4. La terminación de la línea

La terminación de la línea es de manera normal para los Unix de archivos de texto. Las líneas no deben contener arrastrado de espacios. Para facilitar esta convención, la mayoría de los editores pueden configurarse para despojar el arrastrado de los espacios, como un ahorro del funcionamiento.

2.3.2.5. Para mostrar una cadena

Debe estar dentro de comillas dobles o simples (ejemplo: "Hola Mundo", 'Lo que quiero mostrar'). Cabe destacar que si se desea mostrar el símbolo " o ' debe encerrarse en el otro tipo de comillas ("...'...", '..."...') o usarse un escape (\', \"). Toda línea de instrucción siempre termina en un punto y coma (;), al igual que el lenguaje C.

2.3.2.6. Insertar un comentario

Para insertar un comentario de una línea debe empezar por // o por #. El resto de la línea es tratado entonces como un comentario. Para insertar un bloque de comentario, de una o más líneas, se utiliza la combinación /* y */, por ejemplo: /* <COMENTARIOS> */

2.3.2.7. Definiciones de clases

Los nombres de las clases pueden contener sólo caracteres alfanuméricos. Se permiten los números en los nombres de la clase pero se descorazonan. Subrayar sólo se permite en lugar del separador del camino. Por ejemplo, el filename que "Zend/Db/Table.php" debe trazar al nombre de la clase "Zend_Db_Table."

Si el nombre de la clase se compone de más de una palabra, la primera letra de cada nueva palabra debe capitalizarse. No se permiten las letras capitalizadas sucesivas; por ejemplo, una clase que "Zend_PDF" no se permite, mientras "Zend_Pdf" es aceptable.

Cualquier código dentro de una clase debe dentarse la sangría normal de cuatro espacios.

Sólo se permite una clase por el archivo de PHP.

Las declaraciones de las clases tienen su abrazadera de la apertura en una nueva línea:

```
<?php
  <?
class Articulo{
    var $titulo;
    var $anno;
}
?>
```

2.3.2.8. Las clases objeto disponen de getters para los registros de las columnas:

```
$articulo = new Articulo();
//...
$titulo = $articulo->getTitulo();
```

ArticuloPeer es una clase de tipo “peer”; o sea, que tiene métodos estáticos para trabajar con las tablas de la base de datos. La misma proporciona los medios necesarios para obtener los registros de las tablas.

2.3.2.9. Llamadas de funciones

Deben llamarse las funciones sin los espacios entre el nombre de la función, el paréntesis de la apertura, y el primer parámetro; los espacios entre las comas y cada parámetro, y ningún espacial entre el último parámetro, el paréntesis del cierre, y el punto y coma. Aquí es un ejemplo:

```
<? php  $variable=getNombreArticulo($anno, $autor);  ?>
```

2.3.2.10. Las estructuras de mando

Éstos incluyen; si, para, mientras, cambie, etc. Este es un ejemplo:

```
<?php
 If ((condition1) ||(condition2)) {
      action1;
 }
else if ((condition3) && (condition4)) {
      action2;
}
else {
     default action;
}
?>
```

Las declaraciones del mando deben tener un espacio entre la palabra clave del mando y el paréntesis abriendo, distinguirlos de las llamadas de la función.

2.3.3. Diagrama de despliegue

El modelo de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los links de comunicación entre ellos, y las instancias de los componentes y objetos que residen en ellos. El propósito del modelo de despliegue es capturar la configuración de los elementos de procesamiento, y las conexiones entre estos elementos en el sistema. El modelo consiste en uno o más nodos (elementos de procesamiento con al menos un procesador, memoria, y posiblemente otros dispositivos), dispositivos (nodos estereotipados con una capacidad de procesamiento en el nivel modelado de abstracción), y conectores, entre nodos, y entre nodos y dispositivos. A continuación se muestra el diagrama de despliegue correspondiente a la aplicación. El mismo ha sido dividido en tres, para la FGR, las fiscalías provinciales y un último diagrama para las fiscalías municipales, particularizando en cada caso debido a las características especificas de cada una de estas entidades.

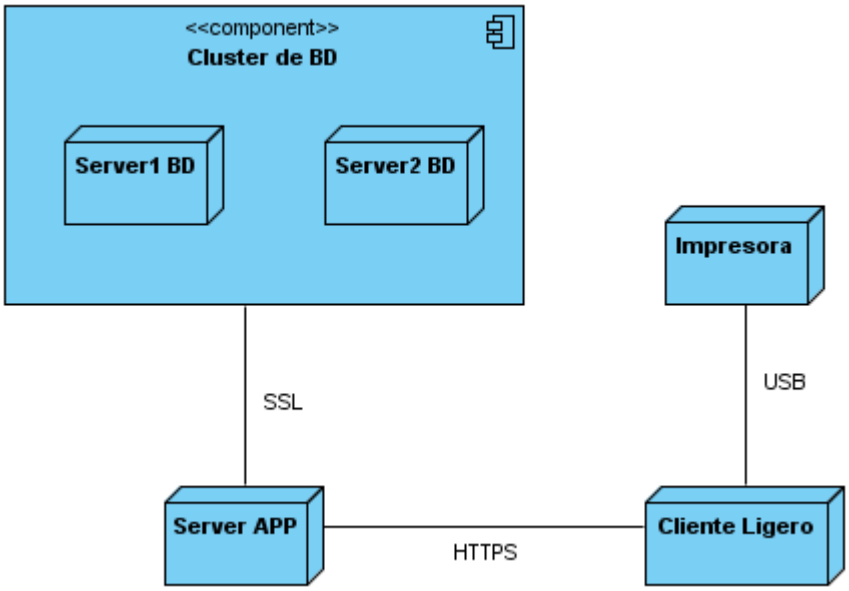

Figura 2 Diagrama de despliegue de la FGR

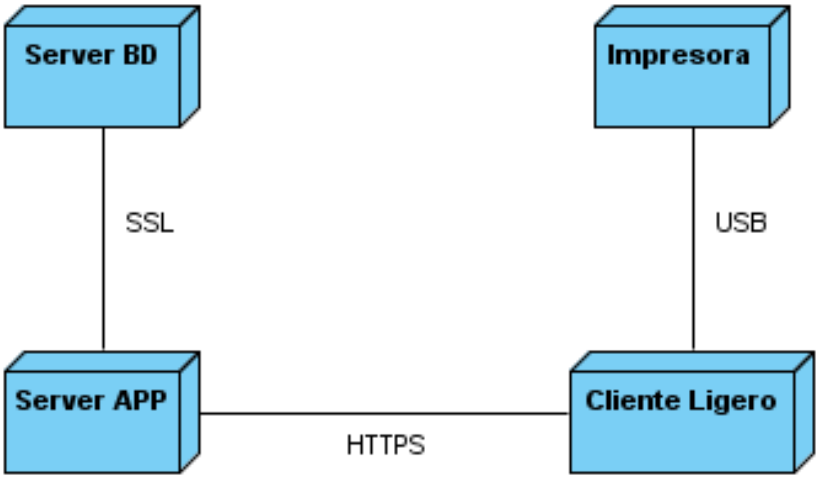

Figura 3 Diagrama de despliegue en las fiscalías provinciales

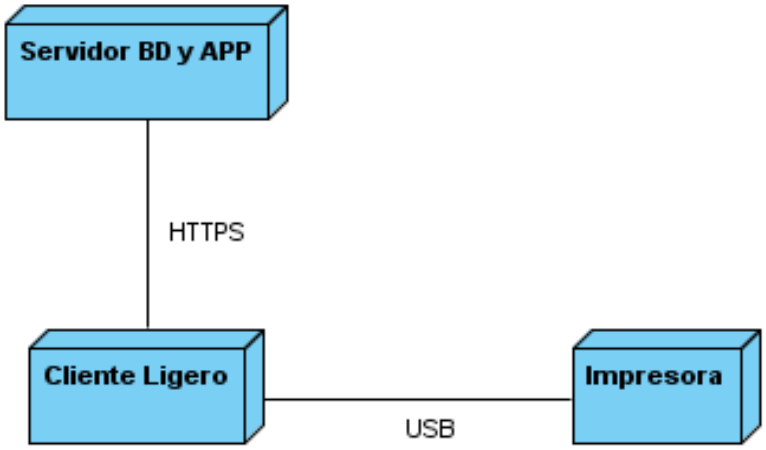

Figura 4 Diagrama de despliegue de las fiscalías municipales

2.3.4. Diagramas de componentes

Los diagramas de componentes modelan la vista estática de un sistema. Se representa como un grafo de componentes de software unidos por medio de relaciones de dependencia (compilación, ejecución), pudiendo mostrarse las interfaces que estos soporten.

En la figura 5 se muestra el diagrama de componentes correspondiente al caso de uso Crear Expediente. Cada paquete en el diagrama representa una división física del sistema. En la imagen se puede apreciar además el uso de los patrones de diseño como el Decorador, Experto, Bajo acoplamiento entre otros que serán explicados en el epígrafe siguiente.

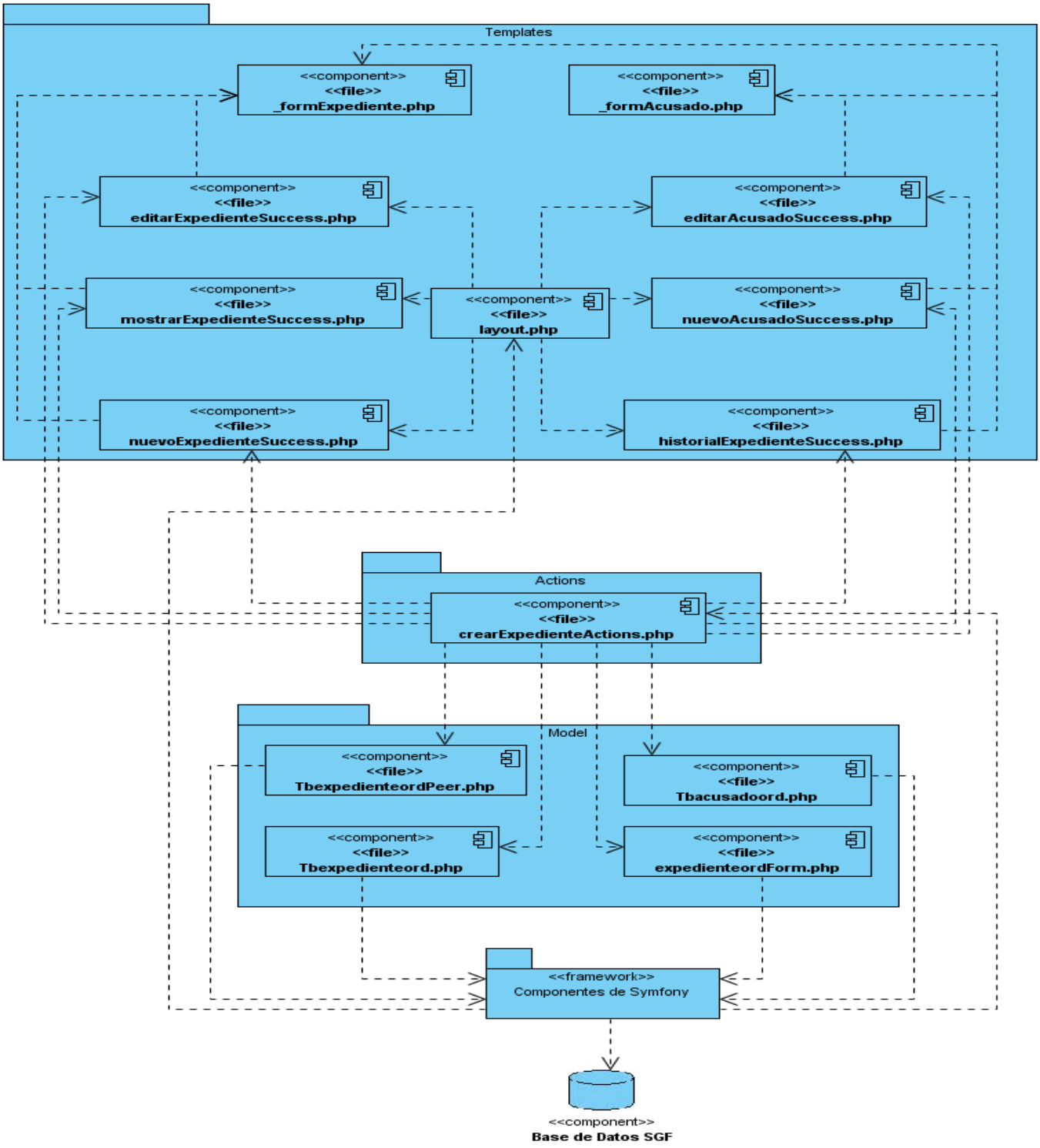

Figura 5 Diagrama de componentes del caso de uso Crear Expediente

2.3.5. Patrones de diseño aplicados

Un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones sobre sus compromisos. Un patrón es una descripción de un problema bien conocido que suele incluir:

- Descripción.
- Escenario de Uso.
- Solución concreta.
- Las consecuencias de utilizar este patrón.
- Ejemplos de implementación.
- Lista de patrones relacionados.

(29) (30)

2.3.5.1. Patrones GRASP[20]

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (29). Existe un gran número de patrones que se sitúan dentro de este grupo. Symfony implementa alguno de ellos. A continuación se señalan.

2.3.5.1.1. *Experto*

Asigna una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Este es uno de los patrones que implementa el framework Symfony al utilizar Propel[21] para el mapeo de objetos de la base de datos. Este ORM[22] genera las clases para la gestión de las entidades según las tablas de la base de datos con las responsabilidades debidamente asignada, pues cada una de estas clases cuenta con un conjunto de funcionalidades relacionadas directamente con la entidad que representan.

2.3.5.1.2. Creador

[20] ***GRASP:*** *acrónimo en inglés de General Responsibility Assignment Software Patterns (en español Patrones Generales de Software para Asignar Responsabilidad).*
[21] ***Propel:*** *Object Relational Mapping (ORM) utilizado por Symfony para la abstracción de la base de datos.*
[22] ***ORM:*** *acrónimo en inglés de Object Relational Mapping.*

Consiste en asignarle a una clase B la responsabilidad de crear una instancia de una clase A en uno de los siguientes casos:

- B agrega los objetos A.
- B contiene los objetos A.
- B registra las instancias de los objetos A.
- B utiliza específicamente los objetos A.
- B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado (así que B es un Experto respecto a la creación de A).

Este patrón se evidencia en las clases actions que se generan de cada módulo. Por ejemplo, la clase *crearExpedienteActions*, es la que contiene las acciones del modulo Crear Expediente Ordinario, en las cuales se crean los objetos de las clases que representan las entidades.

2.3.5.1.3. Alta cohesión

Asigna una responsabilidad de modo que la cohesión siga siendo alta. La cohesión es una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. (29)
Una clase con baja cohesión hace muchas cosas no afines o un trabajo excesivo. No conviene este tipo de clases pues presentan los siguientes problemas:

- Son difíciles de comprender, reutilizar y conservar.
- Las clases con baja cohesión a menudo representan un alto grado de abstracción o han asumido responsabilidades que deberían haber delegado a otros objetos.

Una de las características principales del framework Symfony es la organización del trabajo en el mismo en cuanto a la estructura del proyecto, lo cual permite crear y trabajar con clases con una alta cohesión. Por ejemplo, la clase *Actions* contiene varias funcionalidades estrechamente relacionadas entre ellas, teniendo un sentido común y un propósito único, siendo estas las encargadas de controlar las acciones de las plantillas. Esto hace posible que el software sea flexible a cambios sustanciales con efecto mínimo.

2.3.5.1.4. *Bajo acoplamiento*

Asigna una responsabilidad para mantener bajo acoplamiento. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras.

Este patrón está evidenciado en el framework Symfony ya que dentro de la capa modelo las clases de abstracción de datos son las más reutilizables y no tienen asociaciones con las clases de la capa vista ni con el controlador.

2.3.5.1.5. *Controlador*

Asigna la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones:

- El sistema global (controlador de fachada).
- La empresa u organización global (controlador de fachada).
- Algo en el mundo real que es activo (por ejemplo, el papel de una persona) y que pueda participar en la tarea (controlador de tareas).

Symfony es un framework basado en el patrón arquitectónico Modelo-Vista-Controlador, que define una capa específica para los controladores, que son el núcleo del mismo. Este patrón se puede observar en las clases *sfFrontController, sfWebFrontController, sfContex* propias de Symfony, quien también aplica el patrón Front Controller (Controlador frontal) y tiene una estructura bien organizada de sus controladores que parte desde la página *indexSuccess.php* y se cumplimenta en la clase Actions. Cada clase de esta capa tiene su responsabilidad y es única, por ejemplo, hay controladores que se encargan de la seguridad del sistema trabajando con ficheros .yml[23].

2.3.5.2. Patrones GoF[24]

2.3.5.2.1. *Singleton*

En las acciones se usan los métodos ->getRequest (), ->getUser (), esto se debe a que, en la acción, el método getContext (), guarda una referencia a todos los objetos del núcleo de Symfony, estos métodos pueden ser accedidos desde la vista y desde el controlador, solo varía la forma de llamarlos. (18)

2.3.5.2.2. *Command*

Este patrón se evidencia cuando un objeto o sistema puede recibir varias peticiones o comandos. Reducir la responsabilidad del receptor en el manejo de los comandos, aumenta la facilidad con que pueden

[23] ***.yml:*** *Extensión utilizada por el framework symfony para sus archivos de configuración.*
[24] ***GoF:*** *acrónimo en inglés de "Gang of Four" (en español Pandilla de los Cuatro). Deben su nombre a que el libro en que por primera vez se habla de estos patrones fue realizado por cuatro autores.*

agregarse otros comandos y ofrece las bases para registrar los mismos, para formar colas de espera con ellos y para cancelarlos (deshacerlos).

Este patrón se pone de manifiesto en el método dispatch() de la clase *sfWebFrontController*, que es la encargada de determinar cual módulo y acción usar en dependencia de la petición del usuario.

2.3.5.2.3. *Decorator*

En este método de la clase abstracta sfView, padre de todas las vistas, tienen cada una un decorador para permitir añadir funcionalidades a las vistas dinámicamente. Este patrón se observa en el archivo denominado layout.php que contiene el Layout (plantilla global) de todas las páginas del registro. El mismo almacena el código HTML que es común a todas las páginas del registro, para no tener que repetirlo en cada página, por lo que el Layout decora la plantilla. (19)

2.3.6. Patrón arquitectónico aplicado

En el capítulo anterior se expuso que el framework seleccionado Symfony, hacía uso del patrón arquitectónico Model View Controller, el cual permite que las páginas web contengan mínima codificación ya que la presentación es separada del código PHP.

- El modelo representa la información con la que trabaja la aplicación, es decir, la lógica del negocio.
- La vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo y/o en la vista.

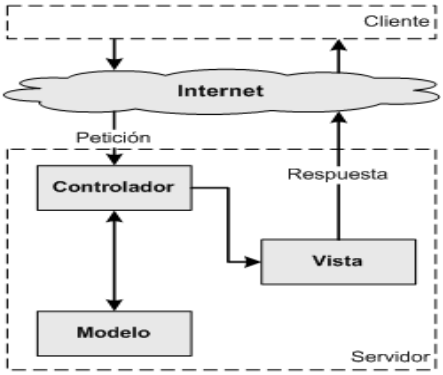

Figura 6 Arquitectura Modelo Vista Controlador implementado por Symfony

La implementación que realiza Symfony de la arquitectura MVC incluye varias clases como son:

- *sfController*: Es la clase del controlador. Se encarga de decodificar la petición y transferirla a la acción correspondiente.
- *sfRequest*: Almacena todos los elementos que forman la petición (parámetros, cookies, cabeceras, etc.)
- *sfResponse*: Contiene las cabeceras de la respuesta y los contenidos. El contenido de este objeto se transforma en la respuesta HTML que se envía al usuario.

(19)

El controlador frontal y el layout son comunes para todas las acciones de la aplicación. Se pueden tener varios controladores y varios layouts, pero solamente es obligatorio tener uno. El controlador frontal es un componente que sólo tiene código relativo al MVC, por lo que no es necesario crear uno, ya que Symfony lo genera de forma automática.

Las clases de la capa del modelo también se generan automáticamente, en función de la estructura de datos de la aplicación. La librería Propel se encarga de esta generación automática, ya que crea el esqueleto o estructura básica de las clases y genera automáticamente el código necesario. Symfony utiliza Propel como ORM[25] y Propel utiliza PDO[26] como capa de abstracción de bases de datos. Estos dos componentes externos han sido desarrollados por el equipo de Propel, y están completamente integrados en Symfony, por lo que se pueden considerar una parte más del framework.

La abstracción de la BD es completamente invisible al programador, así si se cambia el SGBD[27] en cualquier momento, no se debe reescribir ni una línea de código, ya que tan sólo es necesario modificar un parámetro en un archivo de configuración.

La lógica de la vista se puede transformar en un archivo de configuración sencillo, sin necesidad de programarla.

[25] ***ORM**: Object-Relational Mapping o Mapeo de Objetos a Bases de Datos*
[26] ***PDO**: PHP Data Objects*
[27] ***SGBD**: Sistema Gestor de Base de Datos*

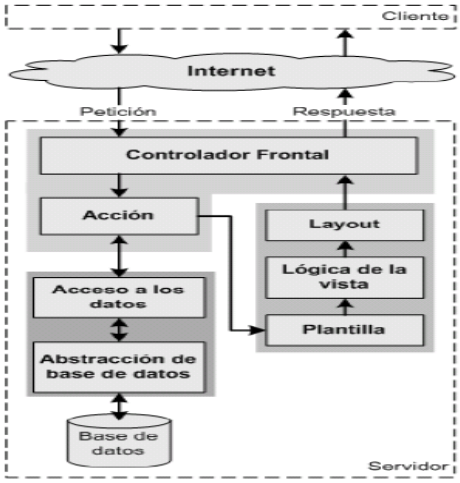

Figura 7 Flujo de trabajo de Symfony según el patrón MVC

2.3.7. Estrategia para la captura de errores

En todo sistema informático, el tratamiento de errores es un tema de suma importancia, no se puede hablar del correcto funcionamiento del mismo sin antes tener validada la aplicación. En teoría, escribimos el código y cada línea funciona como se pretende y los recursos que se emplean siempre están presentes. Sin embargo, éste no siempre es el caso en el mundo real. Los programadores pueden cometer errores, las conexiones de red pueden interrumpirse, los servidores de bases de datos pueden dejar de funcionar y los archivos de disco pueden no tener los contenidos que las aplicaciones creen que contienen. En pocas palabras, el código que se escribe tiene que ser capaz de detectar errores como éstos y responder adecuadamente.

El framework Symfony incluye protección frente a ataques de tipo XSS[28] a través de los mecanismos de escape que se aplica a todos los datos mostrados mediante las variables de las plantillas y a ataques CSRF[29]. Estas opciones se pueden configurar desde la línea de comandos o editando archivo de configuración *security.yml*.

---

[28] ***XSS:*** *del inglés Cross-Site Scripting es un tipo de inseguridad informática o agujero de seguridad basado en la explotación de vulnerabilidades del sistema de validación de HTML incrustado.*

[29] ***CSRF****: del inglés Cross-site Request Forgery o falsificación de petición en sitios cruzados. Es un tipo de exploit malicioso de un sitio web en el que comandos no autorizados son transmitidos por un usuario en el cual el sitio web confía.*

En la aplicación, para el caso específico de un error al escribir la URL[30], se muestra una página indicando el tipo de error producido como se evidencia a continuación:

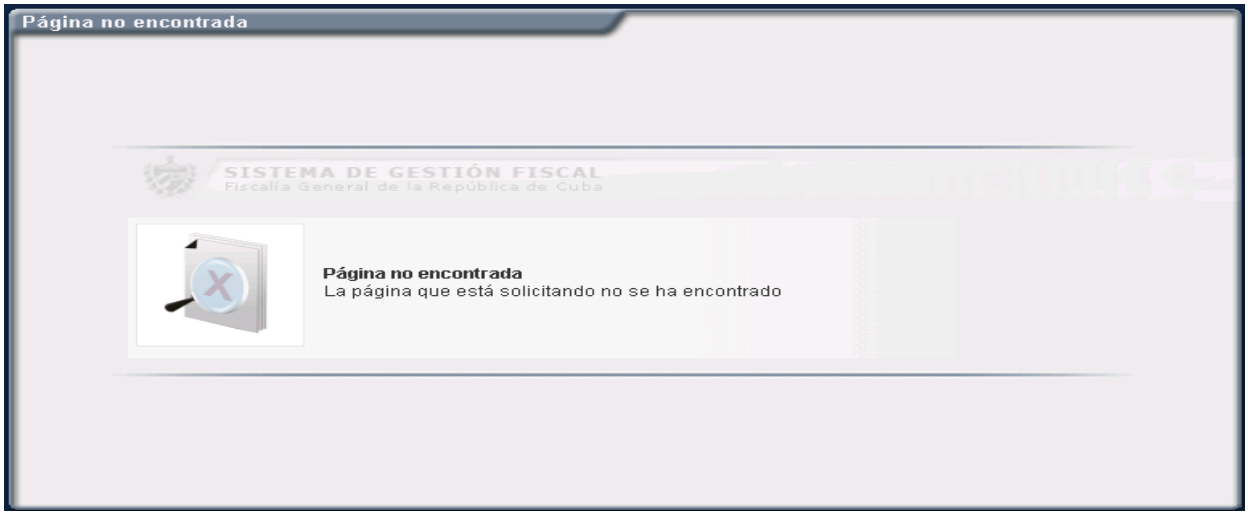

Figura 8 Interfaz de error mostrada por el navegador de página no encontrada.

En el caso en que el usuario intente acceder a una página especifica de la aplicación y no haya sido autenticado anteriormente, el sistema redirecciona al usuario hacia la página principal de la aplicación, donde debe autenticarse (ver Anexo # I).

Una vez que el usuario introduce datos en el sistema e intenta almacenarlos, se verifica primeramente el correcto estado y formato de los mismos a través de JavaScritp, evitando que los datos que se vayan a almacenar en la base de datos tengan un formato incorrecto y que la información no sea lo más fidedigna posible. En caso de existir algún error en los datos, el sistema muestra un mensaje indicándole al usuario el campo incorrecto (ver Figura 10), dándole la posibilidad de corregirlo y volver a realizar la acción de guardado. Para mayor comodidad para el usuario, se decidió que aquellos campos que sólo aceptan números, el sistema sólo permitirá que se escriban números y así mismo para aquellos que solo aceptan letras, ahorrándole tiempo de verificación a la aplicación y al cliente.

[30] ***URL:*** *Uniform Resource Locator (en español Localizador de Recurso Uniforme).*

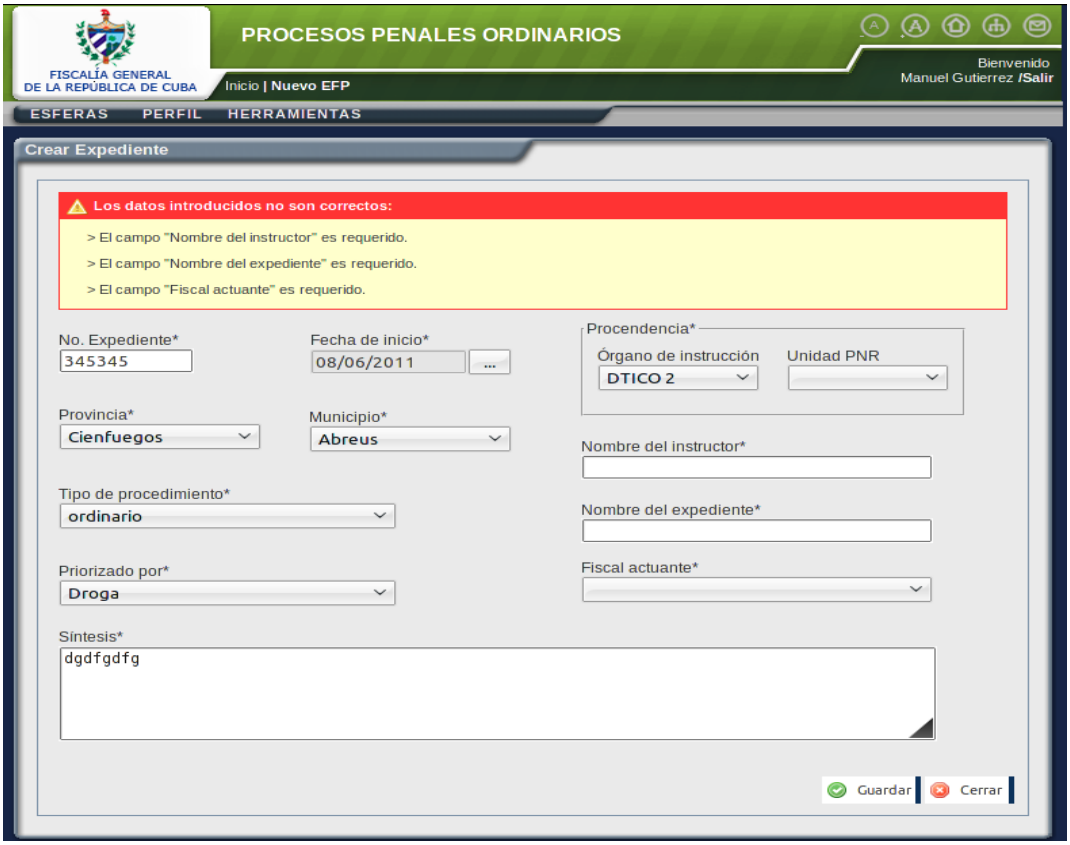

Figura 9 Ejemplo de validación con JavaScript

2.3.8. Seguridad en la aplicación

La posibilidad de ejecutar una acción puede ser restringida a usuarios con ciertos privilegios. Las herramientas proporcionadas por Symfony para este propósito permiten la creación de aplicaciones seguras, en las que los usuarios necesitan estar autenticados antes de acceder a alguna característica o a partes de la aplicación. Añadir esta seguridad a una aplicación requiere dos pasos: declarar los requerimientos de seguridad para cada acción y autenticar a los usuarios con privilegios para que puedan acceder estas acciones seguras.

Antes de ser ejecutada, cada acción pasa por un filtro especial que verifica si el usuario actual tiene privilegios de acceder a la acción requerida. En Symfony, los privilegios están compuestos por dos partes:

- Las acciones seguras requieren que los usuarios estén autenticados.

- Las credenciales son privilegios de seguridad agrupados bajo un nombre y que permiten organizar la seguridad en grupos.

Para restringir el acceso a una acción se crea y se edita un archivo de configuración YAML llamado security.yml en el directorio config/ del módulo. En este archivo, se pueden especificar los requerimientos de seguridad que los usuarios deberán satisfacer para cada acción o para todas las acciones. (18)

El sistema garantiza su seguridad a través diferentes funciones propias del framework Symfony como son:

La clase myUser

A traves de esta clase el usuario puede autenticarse y al mismo tiempo recibir credenciales que se convierten en permisos para acceder a las diferentes partes de la aplicación según el nivel de acceso que tenga el usuario. Estos permisos son gestionados mediante el archivo *security.yml* que trabaja en complemento con la clase en cuestión y posee los atributos y métodos necesarios para el control de acceso y la asignación de privilegios.

El sistema de enrutamiento

Permite simplificar el aspecto de las direcciones URL y brindar mayor seguridad a la aplicación evitando los ataques de injección SQL. El mismo funciona ocultando al usuario la URI[31] interna y mostrando una dirección URL externa más agradable y segura. Todas las funciones que se realizan se gestionan desde un archivo de configuración especial, llamado routing.yml, en el que se pueden definir las reglas de enrutamiento.

2.4. Descripción de clases y operaciones

2.4.1. Clases controladoras

Nombre: *crearExpedienteActions*	
Tipo de clase: Controladora	
Nombre función:	**Descripción**
executeNuevoExpediente	Crea instancias de las clases *Tbexpedienteord* y *Tbdenuncia* y las envía hacia la vista.
executeCrearExpediente	Crea una instancia de la clase *Tbexpedienteord* y almacena los datos en la base de datos de un nuevo

[31] ***URI:*** Uniform Resource Identifier (en español Identificador Uniforme de Recurso). Es una cadena de caracteres corta que identifica inequívocamente un recurso, dígase servicio, página, documento, dirección de correo electrónico o enciclopedia.

	expediente ordinario.
Nombre: *crearExpedienteActions*	
Tipo de clase: Controladora	
Nombre función:	**Descripción**
executeActualizarExpediente	Crea instancias de las clases TbexpedienteordPeer y *Tbdenuncia* y las envía hacia la vista con los resultados del expediente ordinario previamente creado.
executeCrearDenuncia	Crea una instancia de la clase *TbexpedienteordPeer* y actualiza y almacena los datos en la base de datos del expediente ordinario previamente creado.
validateCrearExpediente	Crea una instancia de la clase *Tbdenuncia* y almacena los datos de una nueva denuncia en la base de datos y envía hacia la vista los cambios realizados.
validateActualizarExpediente	Valida si existen errores al crear un nuevo expediente ordinario y los envía hacia la vista.
executeNuevoAcusado	Valida si existen errores al editar los datos del expediente ordinario previamente creado y los envía hacia la vista.
executeEditarAcusado	Valida si existen errores al crear una nueva denuncia y los envía hacia la vista.
executeActualizarAcusado	Crea instancias de las clases *Tbacusadoord*, *Tbpersonanatural*, *Tbdireccionord* y *TbexpedienteordPeer* y las envía hacia la vista.
executeMostrarExpediente	Crea instancias de las clases *Tbacusadoord*, *Tbpersonanatural*, *Tbdireccionord* y *TbexpedienteordPeer* y almacena y relaciona los datos en la base de datos de un nuevo acusado con un expediente ordinario.
executeActualizarMostrarExpediente	Crea instancias de las clases *Tbacusadoord*, *Tbpersonanatural*, *Tbdireccionord* y *TbexpedienteordPeer* y las envía hacia la vista con los resultados del acusado previamente creado.

Nombre: *crearExpedienteActions*	
Tipo de clase: Controladora	
validateActualizarMostrarExpediente	Crea instancias de las clases *Tbacusadoord*, *Tbpersonanatural*, *Tbdireccionord* y *TbexpedienteordPeer* y actualiza y almacena los datos en la base de datos del acusado previamente creado.
executeHistorialExpediente	Valida si existen errores al crear un nuevo acusado y los envía hacia la vista.
executeExportarPDF	Valida si existen errores al editar los datos del acusado previamente creado y los envía hacia la vista.
executeMypdf	Crea instancias de las clases *TbexpedienteordPeer*, *Tbprorogaord*, *Tbdevoluciontribunal* y *Tbdevolucioninstructor* y las envía hacia la vista.

Tabla 1 Descripción de la clase crearExpedienteActions

Nombre: *crearIndicacionesActions*	
Tipo de clase: Controladora	
Nombre función:	**Descripción**
executeNuevaIndicacion	Crea una instancia de la clase *TbindicacionesordForm* y la envía hacia la vista.
executeCrearIndicacion	Crea una instancia de *Tbindicacionesord* y almacena los datos en la base de datos una indicación.
executeMostrarIndicacion	Crea una instancia de la clase *TbindicacionesordForm* y muestra como quedaron los cambios realizados.

Tabla 2 Descripción de la clase crearIndicacionesActions

2.4.2. Clases del modelo

Nombre: crearExpedientePeer	
Tipo de clase: Modelo	
Nombre función:	**Descripción**

getBusquedaExpedienteOrd(parámetros)	Obtiene los expedientes que coincidan con los parámetros especificados.
getDenuncias($noExpediente)	Obtiene todas las denuncias del expediente que se especifica en el parámetro.
getFiscalActuante($idcuadro)	Devuelve el nombre y apellidos del cuadro que se especifica en el parámetro.
getProrrogas($idfiscal, $idexpediente)	Devuelve la cantidad de días de cada prórroga dado un fiscal y un expediente.
getAcusadosExpediente($idexpediente)	Devuelve todos los acusados dado un expediente dado.
getDevolucionTribunal($idexpediente)	Devuelve todas las devoluciones efectuadas por el tribunal a la fiscalía.

Tabla 3 Descripción de la clase crearExpedientePeer

2.4.3. Clases de la presentación

Nombre: Crear Expediente	
Tipo de clase: Presentación	
Nombre del archivo:	**Descripción**
_formExpediente.php	Contiene la estructura del formulario de un expediente.
nuevoExpediente.php	Contiene un formulario vacío para la entrada de datos.
editarExpediente.php	Contiene un formulario con los datos del expediente para ser modificados.
mostrarExpediente.php	Muestra el estado final del documento creado, posibilitando la opción de imprimir el mismo.
_formAcusado.php	Contiene la estructura del formulario de un acusado.
nuevoAcusado.php	Contiene un formulario vacío de un acusado para la entrada de datod.
editarAcusado.php	Contiene un formulario con los datos de un acusado para ser modificados.

Tabla 4 Descripción de la clase de la presentación de crear expediente

2.5. Conclusiones parciales

Con la realización de este capítulo, se arribó a la conclusión, que los artefactos propuestos por los analistas permitieron adquirir una comprensión de todo lo relacionado con las funcionalidades a implementar. Se definieron los estándares de codificación que se tendrían en cuenta para la implementación del sistema con el objetivo de obtener un código más legible y entendible, ayudando a la comunicación entre desarrolladores y se obtuvo un modelo del diagrama de despliegue para la FGR, las fiscalías provinciales y las municipales, especificando en cada caso la comunicación entre sus nodos. Además se realizaron un conjunto de descripciones de las principales clases y funcionalidades que fueron utilizadas.

Se obtuvo una propuesta de solución tangible que gestionará los procesos ordinarios que se realizan en las fiscalías y ayudará a los fiscales en la toma de decisiones y en el control y supervisión del trabajo.

Capítulo 3
Validación de la solución propuesta

3.1. Introducción

La aparición de errores en el desarrollo de software tiene una alta probabilidad, pues el factor humano aun está lejos de lograr la perfección, por lo que de alguna manera debe asegurarse la calidad del producto, y aquí es precisamente donde entran a jugar su papel las pruebas y validaciones al software pues estas son un elemento esencial para el aseguramiento de la calidad del mismo. Estos errores pueden estar dados por la mala especificación de los requisitos del software, uso indebido de las estructuras de datos, errores al integrar módulos, entre otras causas.

En el presente capítulo trata acerca de la validación de la solución propuesta anteriormente, a través de pruebas de software que garanticen el correcto funcionamiento y totalidad de las funcionalidades especificadas por el cliente.

3.2. Pruebas de software

De acuerdo a la IEEE el concepto de prueba se define como: "*Una actividad en la cual un sistema o componente es ejecutado bajo condiciones especificas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente*". De aquí la importancia que tiene realizarle pruebas al software antes de ser entregado al usuario final, garantizando así que el mismo tenga la menor cantidad de errores y mejor calidad posible.

3.2.1. Objetivos de las pruebas

Las pruebas de software, al contrario de lo que normalmente se considera, tienen como objetivos detectar errores no encontrados hasta el momento en la aplicación. Se puede hablar entonces del éxito de las pruebas siempre y cuando se hallen errores en el software. Con las pruebas se puede además observar hasta que punto el software parece funcionar en concordancia con los requisitos funcionales descritos para el sistema; aunque no pueden asegurar la ausencia de defectos, sólo puede mostrar que existen. (27)

3.3. Pruebas de caja negra

3.3.1. Objetivos

El principal objetivo de realizar este tipo de prueba al sistema es para detectar el incorrecto o incompleto funcionamiento del mismo, así como los errores de interfaces, errores en estructuras de datos o en accesos a bases de datos externas, rendimiento y errores de inicialización y terminación.

3.3.2. Aplicación de las pruebas de caja negra

Entre las técnicas de caja negra que existen, se utilizó la prueba de partición de equivalencia. Esta técnica divide el campo de entrada de un programa en clases de datos de las cuales se puede obtener los casos de pruebas correspondientes. Mediante la aplicación de esta se examinó los valores válidos e inválidos de las entradas existentes en el software, descubriendo de forma inmediata clase de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

Para definir las clases de equivalencia se tuvieron en cuenta un conjunto de reglas:

- Si una condición de entrada especifica un rango, entonces se confeccionan una clase de equivalencia válida y dos inválidas.
- Si una condición de entrada especifica la cantidad de valores, se identifica una clase de equivalencia válida y dos inválidas.
- Si una condición de entrada especifica un conjunto de valores de entrada y existen razones para creer que el programa trata en forma diferente a cada uno de ellos, se identifica una clase válida para cada uno de ellos y una clase inválida.
- Si una condición de entrada especifica una situación de tipo “debe ser”, se identifica una clase válida y una inválida

Luego de tener las clases válidas e inválidas definidas, se procedió a definir los casos de pruebas, para todos los casos de uso. Cada uno de estos casos de pruebas se definió teniendo en cuenta lo siguiente:

- Escribir un nuevo caso que cubra tantas clases de equivalencia válidas no cubiertas como sea posible hasta que todas las clases de equivalencia hayan sido cubiertas por casos de prueba.
- Escribir un nuevo caso de prueba que cubra una y solo una clase de equivalencia inválida hasta que todas las clases de equivalencias inválidas hayan sido cubiertas por casos de pruebas.

Para ver los datos casos de prueba correspondiente al caso de uso Crear expediente, ir al Anexo # X. Los restantes casos de pruebas se encuentran anexados a los artefactos generados de la presente

investigación.

Dichas pruebas fueron aplicadas por el centro de Calisoft de la universidad. Durante las mismas se detectaron un total de 22 no conformidades entre errores ortográficos, de validación y funcionales. Todos fueron satisfactoriamente corregidos para lograr la liberación de la aplicación por dicho centro.

3.4. Evaluación de la calidad utilizando métricas

Las métricas tienen por objetivo medir la calidad de los productos intermedios generados en un proyecto de software. En este epígrafe se validará la solución propuesta a través de métricas que medirán el estado de ciertos atributos de calidad que permitirán emitir un juicio valorativo de la calidad del software.

3.4.1. Atributos de calidad que se abarcan

- *Responsabilidad o Cohesión:* Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- Complejidad *de implementación:* Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- *Reutilización*: Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- *Acoplamiento:* Consiste en el grado de dependencia o interconexión de una clase o estructura de clase con otras, está muy ligada a la característica de Reutilización.
- *Complejidad del mantenimiento:* Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.
- *Cantidad de pruebas:* Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad del producto (Componente, módulo, clase, conjunto de clases, etc.) diseñado.

3.4.2. Tamaño Operacional de las Clases

Las métricas orientadas al tamaño de las clases OO[32] se centran en el recuento de atributos y operaciones para cada clase individual, y los valores promedio para el sistema OO como un todo.

El tamaño general de una clase puede medirse teniendo en cuenta los siguientes aspectos:

[32] ***OO**: Orientada a Objetos.*

- El total de operaciones (operaciones tanto heredadas como privadas de la instancia), que se encapsulan dentro de la clase.
- El número de atributos (atributos tanto heredados como privados de la instancia), encapsulados por la clase.

En este caso se aplicará la métrica con respecto al total de operaciones. Se debe tener en cuenta que los valores grandes de TOC[33] demuestran que una clase puede tener demasiada responsabilidad, lo cual reducirá la reutilización de la clase y complicará la implementación y la comprobación, por otra parte cuanto menor sea el valor medio para el tamaño, más probable es que las clases existentes dentro del sistema se puedan reutilizar.

Luego de aplicada la métrica a un conjunto de clases de la aplicación, la misma arrojó como resultado:

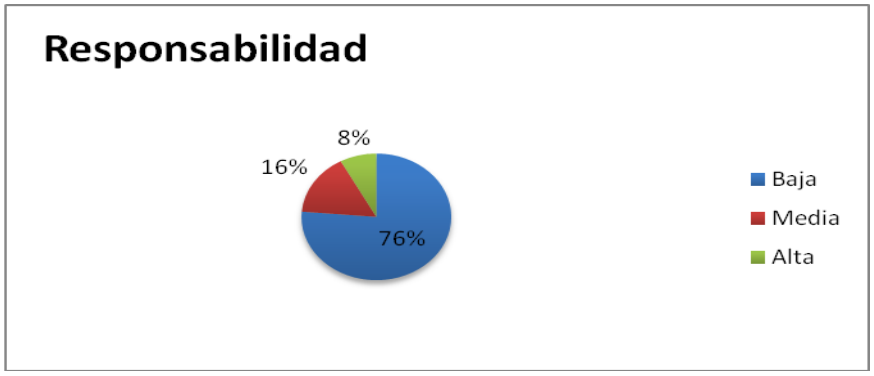

Figura 10 Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad

[33] ***TOC**: Tamaño Operacional de las Clases*

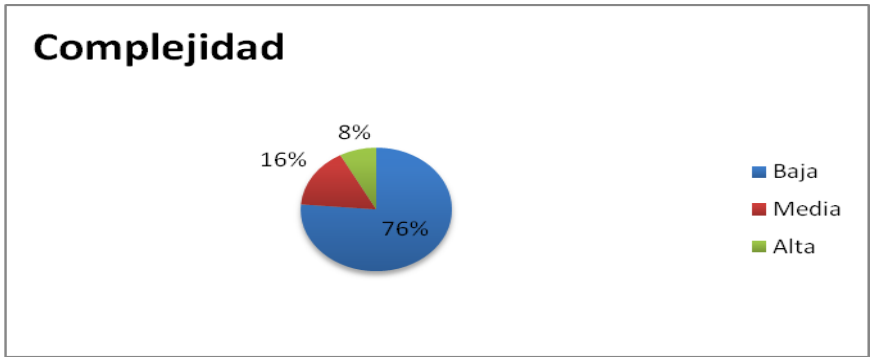

Figura 11 Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación.

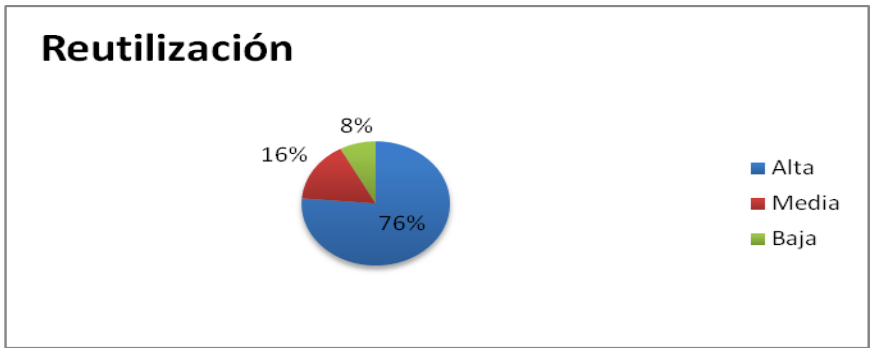

Figura 12 Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica TOC, se puede concluir que la implementación del modulo Ordinario tiene una calidad buena teniendo en cuenta que existe una responsabilidad baja de las clases en un 76% de lo que se puede inferir que un cambio en el sistema afectaría a pocas clases. Además el 76% de las mismas poseen evaluaciones positivas en los atributos de calidad complejidad de implementación y reutilización lo cual garantiza una solidez en el sistema puesto que habrá gran reutilización. Para más información ver instrumentos y tabla de resultados en Anexo # VIII.

3.4.3. Relaciones entre clases

Esta métrica está dada por el número de relaciones de uso de una clase. Los atributos que afecta son el Acoplamiento, la Complejidad de mantenimiento, la Reutilización y la Cantidad de pruebas. De manera que mientras mayor sean las relaciones entre las clases mayor será el Acoplamiento, la Complejidad de mantenimiento y la Cantidad de pruebas, mientras que su Reutilización disminuye.

Al aplicar esta métrica se obtuvieron resultados satisfactorios en cuanto a los atributos de calidad acoplamiento, complejidad de mantenimiento y cantidad de pruebas.

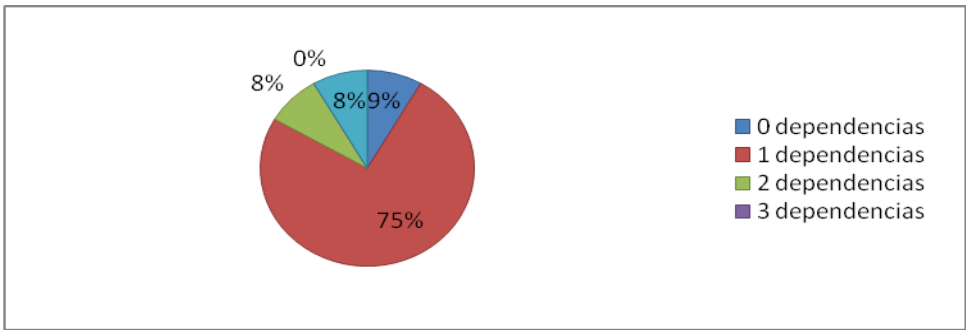

Figura 13 Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos

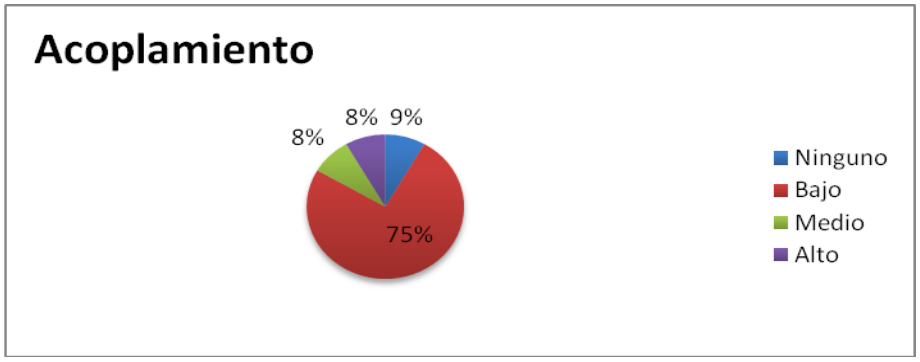

Figura 14 Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento.

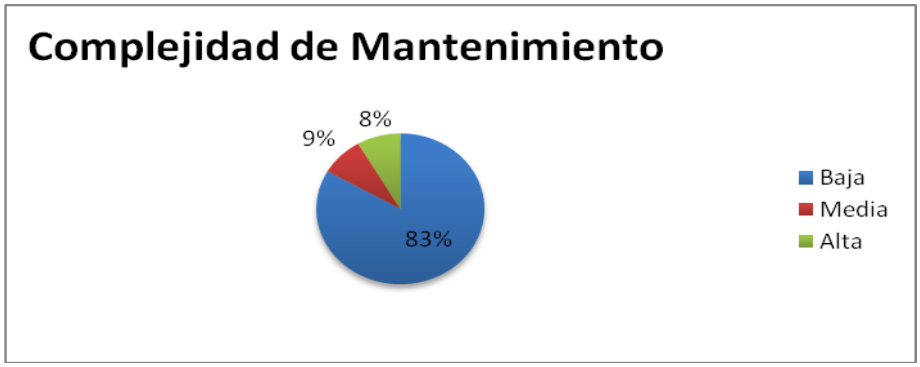

Figura 15 Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento.

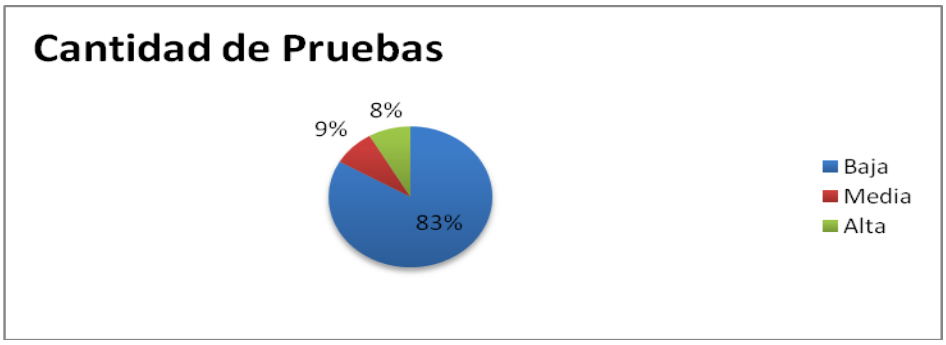

Figura 16 Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas.

La aplicación de esta métrica demostró la baja dependencia que existe entre las clases con un 75 % de ellas dependientes de una clase. Esto favorece el hecho de que al ocurrir un cambio en alguna de las clases, la afectación a las restantes es casi nula. Además existe un bajo acoplamiento por lo que la fuerza con que se relacionan las clases es débil, posibilitando modificaciones con muy poco impacto negativo en el sistema. Los atributos complejidad de mantenimiento y cantidad de pruebas también aportan resultados satisfactorios.

3.5. Validación de los estándares de codificación utilizados

PHP CodeSniffer 1.1.0 es una herramienta perteneciente a los paquetes PEAR, que chequea los archivos PHP y detecta las violaciones que se hayan cometido en dependencia de los estándares de codificación definidos, indicando la línea y la recomendación de modificación. Se trata de una herramienta de

desarrollo esencial que garantiza que el código se mantenga limpio y consistente. También puede ayudar a prevenir algunos errores de semántica común hecho por los desarrolladores. Contiene algunos estándares definidos para Zend, Symfony, Pear, MySource, entre otros (34). En este caso se utilizaron los estándares para Symfony, definidos en el capítulo 2.

La aplicación de la herramienta arrojó una serie de errores en cuanto a la aplicación de los estándares. Los mismos se relacionan en la siguiente figura. Los errores fundamentales detectados pueden agrupar en las siguientes categorías.

- Líneas indentadas incorrectamente.
- Líneas de código que superan los 85 caracteres.
- No existen espacios después de las comas en los parámetros de llamadas a funciones.

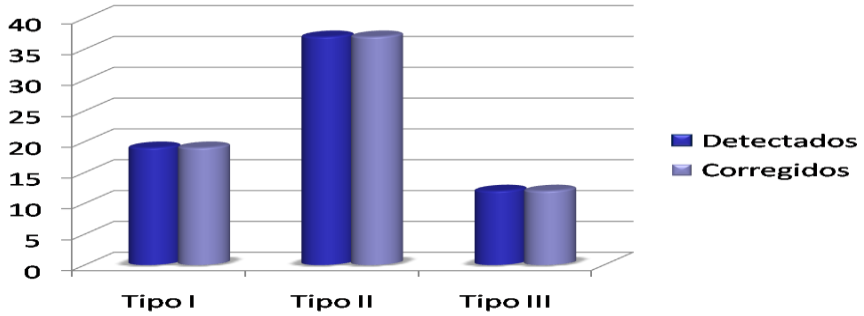

Figura 17 Resultados de la aplicación de la herramienta PHP CodeSniffer.

3.6. Conclusiones parciales

Con el desarrollo de este capítulo se pudo apreciar la importancia de la realización de las pruebas en el proceso de desarrollo de software. Se desarrollaron las pruebas de caja negra, las cuales de manera general arrojaron resultados satisfactorios, siendo corregidos todos los errores encontrados por las mismas. Además se aplicaron métricas de calidad a las clases que dieron una valoración del estado de atributos de calidad como Reutilización, Complejidad de Mantenimientos, Responsabilidad, entre otros. Estas métricas utilizadas aportaron resultados positivos por lo que se puede considerar que la aplicación tiene buena calidad. Finalmente se realizó un análisis de las violaciones cometidas en la utilización de los estándares definidos a través de la herramienta PHP CodeSniffer. Su corrección posibilitó una mayor estandarización en el código, haciéndolo mas legible y entendible.

Conclusiones Generales

La realización de la presente investigación propició un estudio de los sistemas de gestión fiscal que existen en el mundo arrojando como resultado que ninguno satisfacía las necesidades de las fiscalías y no se ajustaban al proceso judicial cubano, lo cual condujo entonces a un estudio de las herramientas y lenguajes de programación que podrían ser utilizados en la confección de una aplicación que se ajustara a los procesos que se desarrollan en las fiscalías del país y ayudara en la toma de decisiones de los fiscales, alcanzando así el objetivo general propuesto para la presente investigación de desarrollar la implementación y prueba de una herramienta que facilite el control y la gestión de los procesos ordinarios, disminuyendo considerablemente el tiempo de tramitación de los expedientes.

Se tuvo en cuenta además la aplicación de buenas prácticas en el desarrollo de software como son los estándares de codificación con el propósito de lograr un código legible y organizado y de patrones de diseño que propiciaron una optimización y disminución de errores en la aplicación.

Finalmente se realizaron un conjunto de pruebas al software para comprobar la calidad del mismo como fueron las pruebas de caja negra y la evaluación de un grupo de atributos de calidad a través de métricas que arrojaron resultados satisfactorios.

Recomendaciones

Una vez culminada la aplicación que gestionará los procesos ordinarios que se llevan a cabo en la FGR y demás sedes fiscales del país, se recomienda:

- Realizar una prueba piloto en un ambiente real de ejecución y con un mayor volumen de datos.
- Profundizar en el análisis realizado con el objetivo de encontrar nuevas funcionalidades que puedan ser incorporadas a la aplicación.
- Desarrollar una segunda versión de la aplicación donde se utilice un framework del lado cliente de con el objetivo de obtener una aplicación más amigable para el usuario final.

Referencias bibliográficas

1. **Fiscalía General de la República de Cuba.** Fiscalía General de la República de Cuba. [En línea] 2006. http://www.fgr.cu/.

2. **Sistemas Jurídicos S.R.L.** Lex-Doctor Gestión Jurídica. [En línea] 1989-2010. http://www.lex-doctor.com.

3. **BRiNDYS Software .** GEDEX. [En línea] 1996-2011. http://www.brindys.com/gedex/sammenu.html?ibl=es-mx..

4. **Estudios y Proyectos Informaticos.** ABOGest. [En línea] 2005. http://www.abogest.com..

5. **Pérez Díaz, Juan Enrique y Monferrer Pérez, Shaily.** *Análisis y Diseño del Módulo Sumario del proyecto Sistema de Gestión Fiscal.* Universidad de las Ciencias Informáticas. 2009. Tesis.

6. **Colectivo de autores.** Lenguajes de programación (PE). [En línea] 2009. http://www.lenguajes-de-programacion.com/programacion-estructurada.shtml.

7. —. Lenguajes de programación(POO). [En línea] 2009. http://www.lenguajes-de-programacion.com/programacion-orientada-a-objetos.shtml.

8. **Eckel, Bruce.** *Thinking in Java.* 2nd edition. 2000.

9. **Mercer, Dave.** *Fundamentos de programación en ASP 3.0.*

10. **Gallego Vázquez, José Antonio.** *Desarrollo web con PHP y MySQL.* s.l. : Anaya Multimedia, 2003.

11. **Ramos Manso, Martin.** *Programación PHP. Sitios web interactivos.* págs. 13-16.

12. **Kemedy, Chuck Musciano y Bill.** *HTML La guía completa.* 2. s.l. : McGRAW-HILL INTERAMERICANA EDITORES, S.A. de C.V., 1999.

13. **Pérez, Javier Eguíluz.** *Introducción a CSS.* 2008.

14. **Eguíluz Pérez, Javier.** *Introducción a Java Script.* 2008.

15. **EllisLab, Inc.** CodeIgniter. [En línea] 2001-2011. http://codeigniter.com/.

16. **Zend Technologies Ltd.** ZEND FRAMEWORK. [En línea] 2006-2011. http://zendframework.com/.

17. **Cake Software Foundation, Inc.** Cookbook. [En línea] 2010-2011. http://book.cakephp.org/es.

18. **Potencier, Fabien.** *Symfony la guía definitiva.* 2008.

19. —. *El tutorial Jobbet.* 2009.

20. **The Apache Software Foundation.** Apache. [En línea] 2011. http://www.apache.org/.

21. **Ciberaula International Training, S.L.** Ciberaula. [En línea] 2010. http://linux.ciberaula.com/articulo/linux_apache_intro.

22. **Gilfillan, Ian.** *La biblia MySQL.*

23. **Martinez, Rafael.** PostgreSQL-es. [En línea] 2009-2011. http://www.postgresql.org.es/.

24. **Zend Technologies Ltd.** Zend. [En línea] 2010. http://www.zend.com/en/products/studio/.

25. **Oracle Corporation.** Netbeans. [En línea] 2011. http://netbeans.org.

26. **Galíndez, Rodrigo.** *Control de versiones usando subversion.* 2008.

27. **Pressman, Roger S.** *Ingenería de Software, un enfoque práctico.* 5ta edición. 2005. págs. 281-303.

28. **Date, C.J.** *Introducción a los Sistemas de Base de Datos.* 7ma edición. s.l. : Pearson Educación, 2001. pág. 14.

29. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 1ra edición. 1999. págs. 189-424.

30. **Pressman, Roger S.** *Ingenería de Software, un enfoque práctico.* 5ta edición. 2005. págs. 390-392.

31. **Worsley, John C. y Drake, Joshua D.** *Practical PostgreSQL.* 1ra Edición. s.l. : O´Reilly, 2002.

32. **Stefanov, Stoyan.** *Object Oriented Java.* s.l. : Packt Publishing Ltd, 2008. 978-1-847194-14-5.

33. **Oracle Corporation.** MySQL. [En línea] 2010. http://www.mysql.com/.

34. **The PHP Group.** Pear. [En línea] 2001-2011. http://pear.php.net/package/PHP_CodeSniffer.

Anexos

Anexo #I. Interfaz principal de acceso a la aplicación

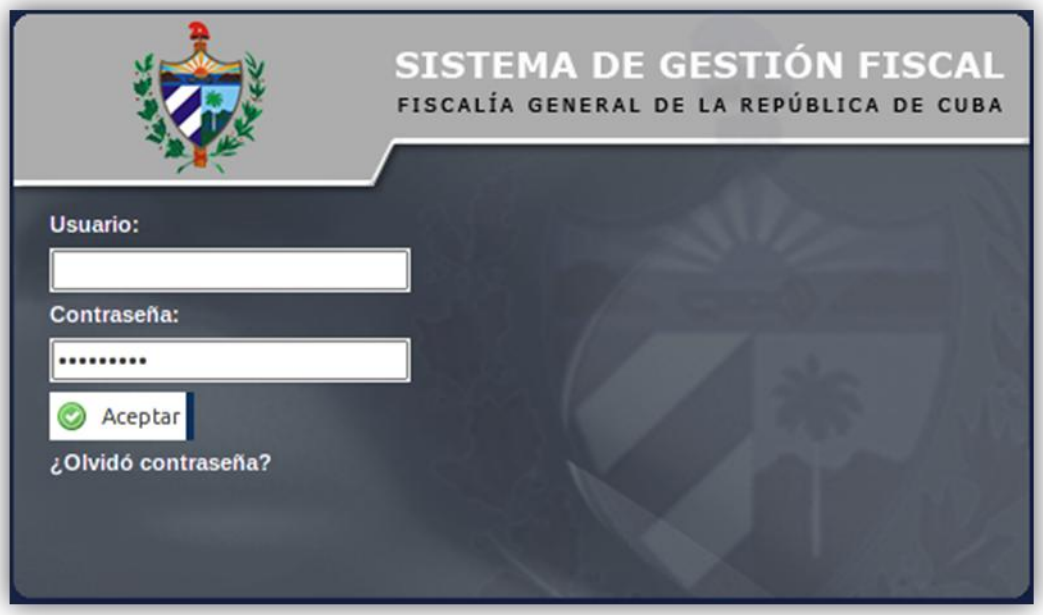

Figura 18 Interfaz principal de la aplicación.

Anexo #II. Interfaz crear Expediente de Fase Preparatoria

Figura 19 Interfaz del escenario Crear Expediente Ordinario.

Anexo #III. Interfaz de búsquedas

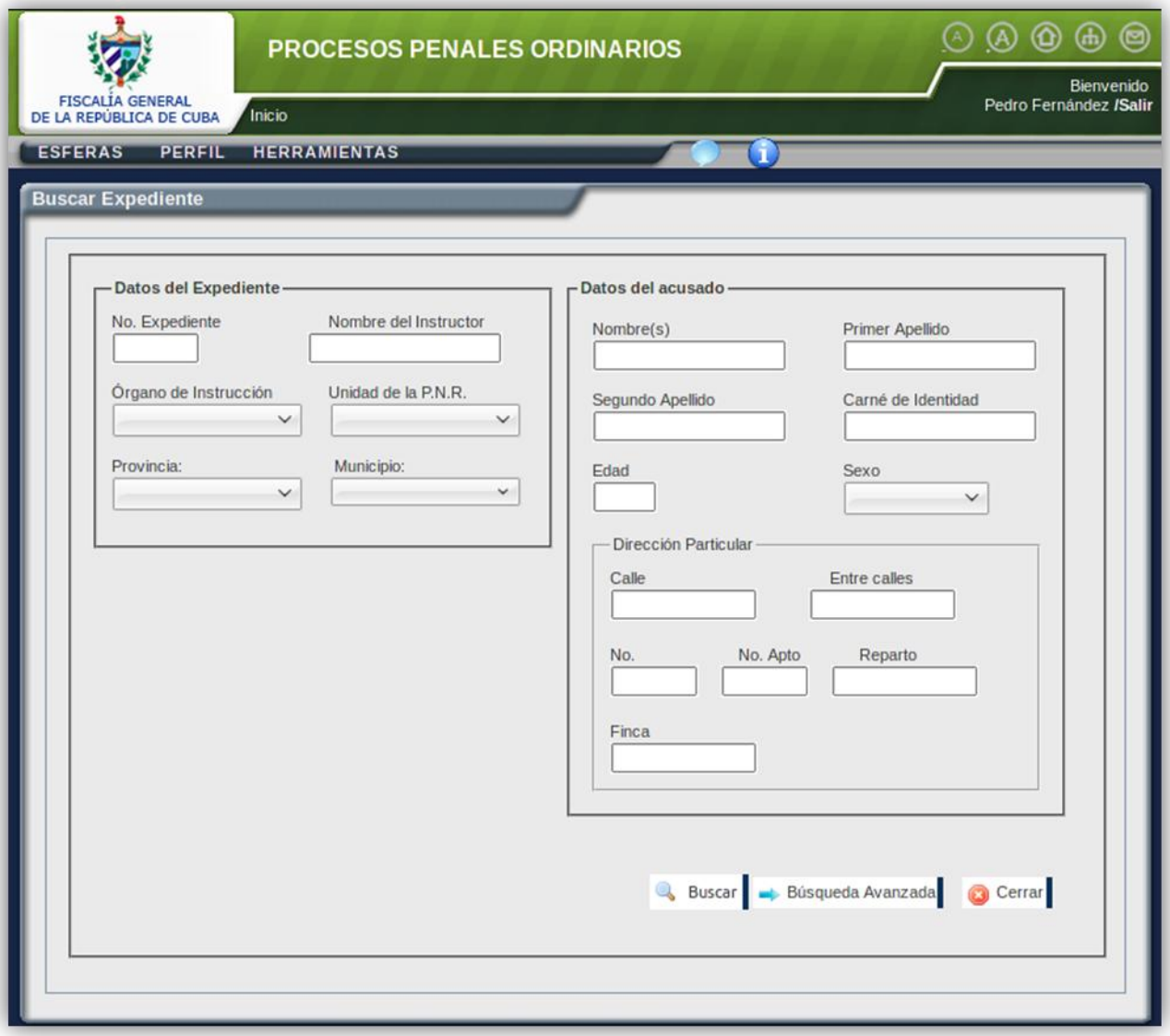

Figura 20 Interfaz del escenario Buscar Expediente Ordinario

Anexo #IV. Interfaz decisiones del fiscal

Figura 21 Interfaz del escenario Decisiones del fiscal

Anexo #V. Interfaz realizar indicación

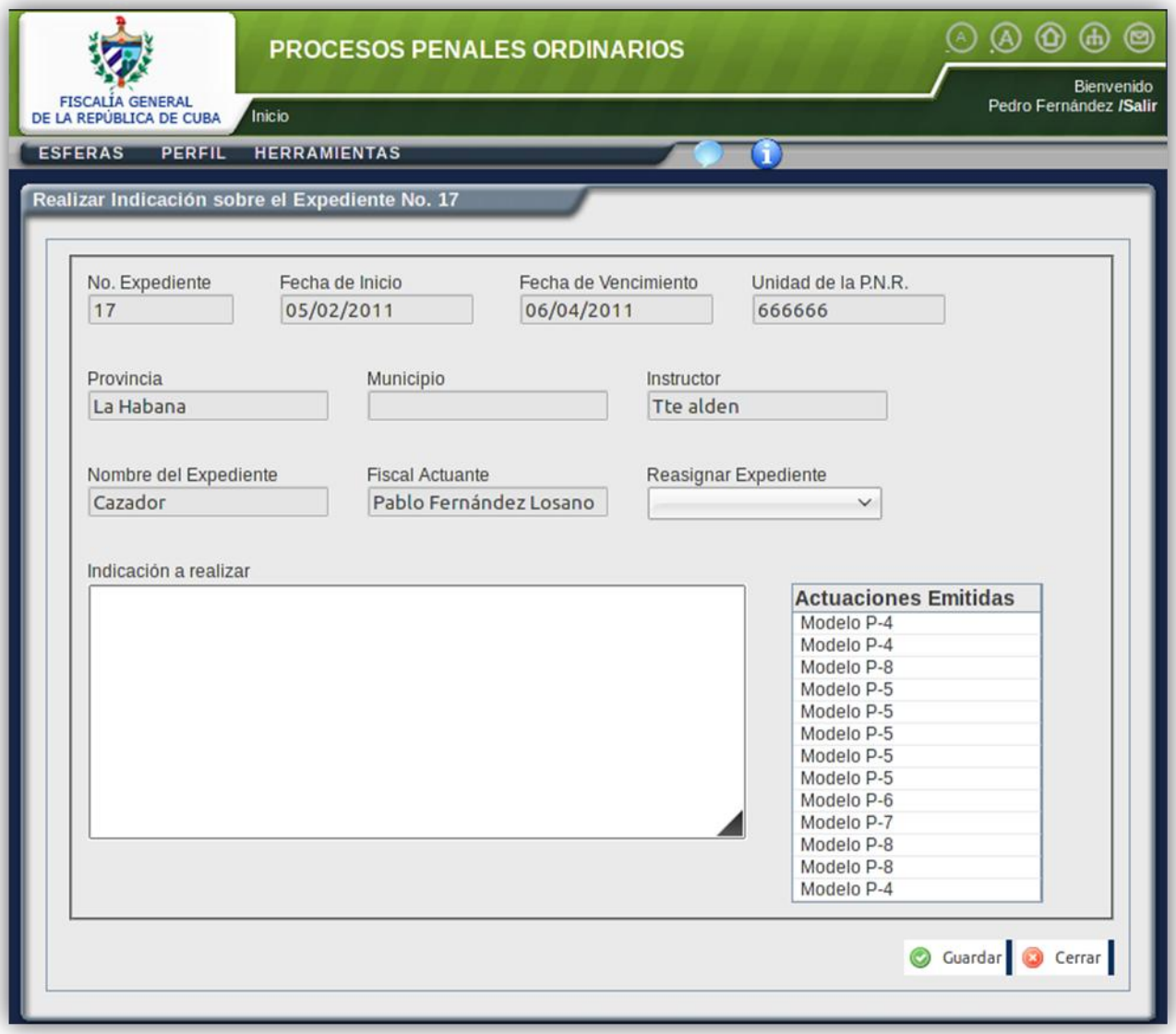

Figura 22 Interfaz del escenario Realizar Indicaciones.

Anexo #VI. Interfaz mostrar historial del expediente

Figura 23 Interfaz del escenario Historial del Expediente Ordinario

Interfaz reportes

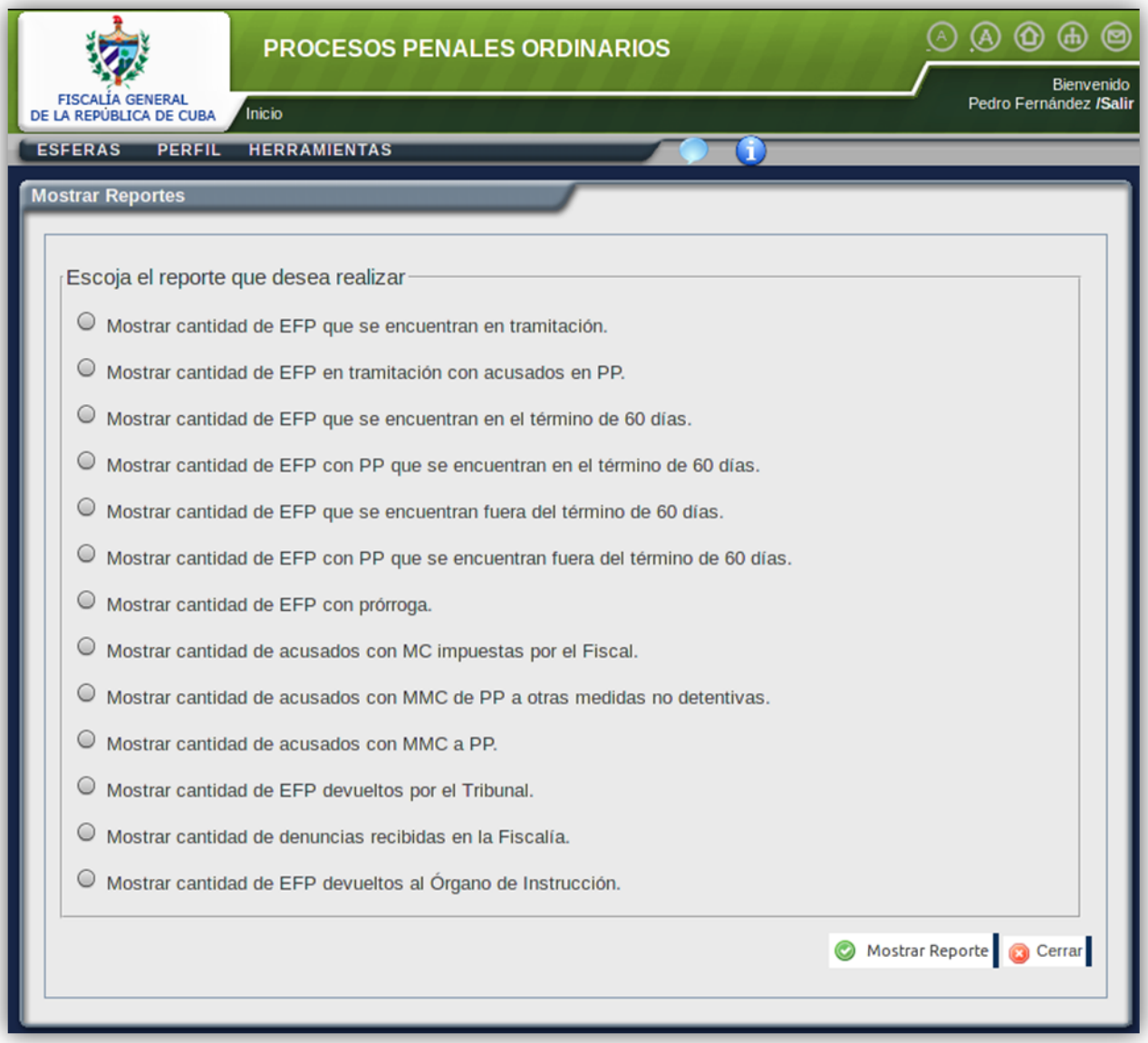

Figura 24 Interfaz del escenario Realizar reportes

Anexo #VII. Instrumento de medición de la métrica Tamaño Operacional de las Clases

Atributo de calidad	Categoría	Criterio
Responsabilidad	Baja	< =Promedio.
	Media	Entre Promedio y 2* Promedio.
	Alta	> 2* Promedio.
Complejidad implementación	Baja	< = Promedio.
	Media	Entre Promedio y 2* Promedio.
	Alta	> 2* Promedio.
Reutilización	Baja	> 2* Promedio.
	Media	Entre Promedio y 2* Promedio.
	Alta	<= Promedio.

Tabla 5 Rango de valores para la evaluación técnica de los atributos de calidad

No	Módulo	Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
1	Crear Expediente	Tbexpedienteord	2	Baja	Baja	Alta
2	Crear Expediente	TbexpedienteordPeer	32	Alta	Alta	Baja
3	Crear Expediente	TbexpedienteordForm	1	Baja	Baja	Alta
4	Crear Expediente	BaseTbexpedienteordForm	7	Alta	Alta	Baja
5	Crear Expediente	TbexpedienteordActions	17	Alta	Alta	Baja
6	Busquedas	BusquedasForm	1	Baja	Baja	Alta
7	Busquedas	busquedasActions	5	Media	Media	Media
8	Crear Indicaciones	Tbindicacionesord	0	Baja	Baja	Alta
9	Crear Indicaciones	TbindicacionesordPeer	1	Baja	Baja	Alta
10	Crear Indicaciones	TbindicacionesordForm	1	Baja	Baja	Alta
11	Crear Indicaciones	BaseTbindicacionesordForm	1	Baja	Baja	Alta
12	Crear Indicaciones	TbindicacionesordActions	3	Baja	Baja	Alta
13	CrearActuaciones	Tbactuaciones	0	Baja	Baja	Alta
14	CrearActuaciones	TbactuacionesPeer	2	Baja	Baja	Alta
15	CrearActuaciones	TbactuacionesForm	0	Baja	Baja	Alta
16	CrearActuaciones	BaseTbactuacionesForm	5	Media	Media	Media
17	CrearActuaciones	TbactuacionesActions	3	Baja	Baja	Alta
18	Mostrar Reportes	MostrarReportesActions	2	Baja	Baja	Alta
19	Modelo TbEI1	Tbei1	1	Baja	Baja	Alta
20	Modelo TbEI1	Tbei1Peer	0	Baja	Baja	Alta
21	Modelo TbEI1	Tbei1Form	1	Baja	Baja	Alta
22	Modelo TbEI1	BaseTbei1	2	Baja	Baja	Alta

23	Modelo TbEI1	Tbei1Actions	6	Media	Media	Media
24	Modelo P4	Tbp4	0	Baja	Baja	Alta
25	Modelo P4	Tbp4Peer	0	Baja	Baja	Alta
26	Modelo P4	Tbp4Form	1	Baja	Baja	Alta
27	Modelo P4	BaseTbp4Form	2	Baja	Baja	
28	Modelo P4	Tbp4Actions	6	Media	Media	Media
29	Modelo P5	Tbp5	0	Baja	Baja	Alta
30	Modelo P5	Tbp5Peer	0	Baja	Baja	Alta
31	Modelo P5	Tbp5Form	1	Baja	Baja	Alta
32	Modelo P5	BaseTbp5Form	2	Baja	Baja	Alta
33	Modelo P5	Tbp5Actions	6	Media	Media	Media
34	Modelo P6	Tbp6	0	Baja	Baja	Alta
35	Modelo P6	Tbp6Peer	0	Baja	Baja	Alta
36	Modelo P6	Tbp6Form	1	Baja	Baja	Alta
37	Modelo P6	BaseTbp6Form	2	Baja	Baja	Alta
38	Modelo P6	Tbp6Actions	6	Media	Media	Media

Tabla 6 Resultados de la evaluación de la métrica TOC

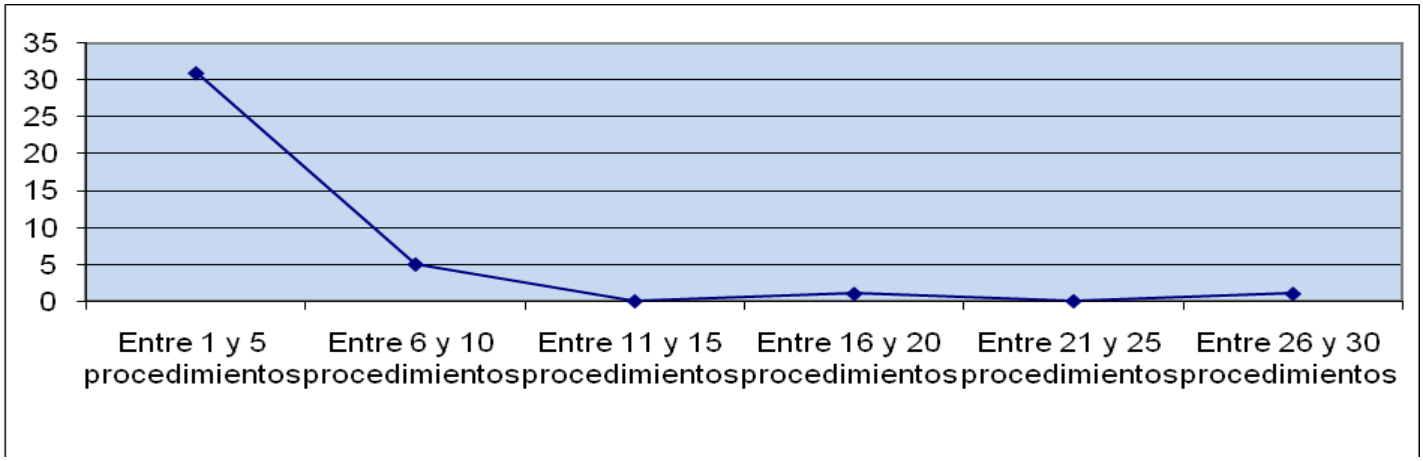

Figura 25 Gráfica de los resultados de la evaluación de la métrica TOC

Anexo #VIII. Instrumento de medición de Relaciones entre clases.

Atributos de calidad	Categoria	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad Mantenimiento	Baja	<= Promedio

	Media	Entre Promedio y 2* Promedio.
	Alta	> 2* Promedio.
Cantidad de Pruebas	Baja	<= Promedio.
	Media	Entre Promedio y 2* Promedio.
	Alta	> 2* Promedio.

Tabla 7 Rango de valores para la evaluación técnica de los atributos de calidad

No	Subsistema	Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mantenimiento	Cantidad de Pruebas
1	Crear Expediente	TbexpedienteordActions	5	Alto	Alta	Alta
2	Busquedas	busquedasActions	2	Medio	Media	Media
3	Crear Indicaciones	TbindicacionesordActions	1	Bajo	Baja	Baja
4	CrearActuaciones	TbactuacionesActions	1	Bajo	Baja	Baja
5	Mostrar Reportes	MostrarReportesActions	0	Ninguno	Baja	Baja
6	Modelo TbEI1	Tbei1Actions	1	Bajo	Baja	Baja
7	Modelo P4	Tbp4Actions	1	Bajo	Baja	Baja
8	Modelo P5	Tbp5Actions	1	Bajo	Baja	Baja
9	Modelo P6	Tbp6Actions	1	Bajo	Baja	Baja
10	Modelo P7	Tbp7Actions	1	Bajo	Baja	Baja
11	Modelo P8	Tbp8Actions	1	Bajo	Baja	Baja
12	Modelo P9	Tbp9Actions	1	Bajo	Baja	Baja

Tabla 8 Resultados de la evaluación de la métrica RC

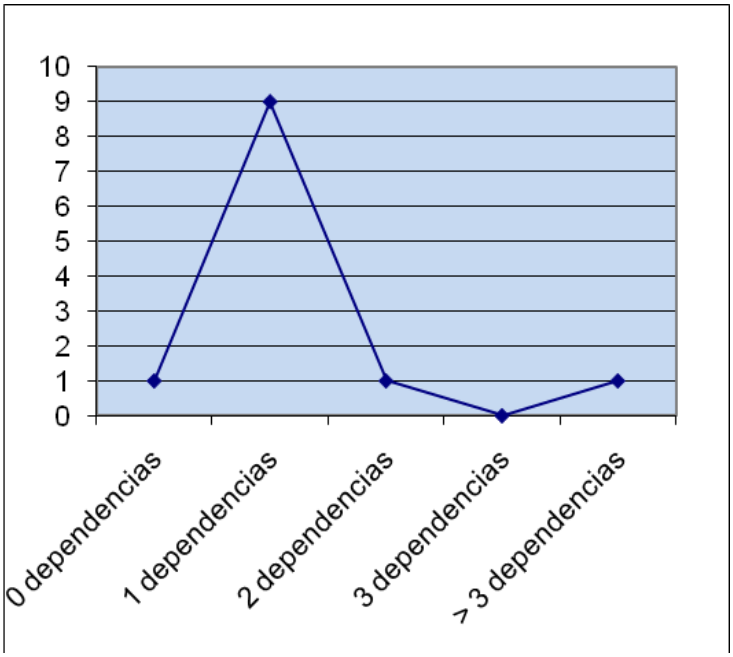

Figura 26 Gráfica de los resultados de la evaluación de la métrica RC.

Glosario de Términos

A

Arquitectura: Uno de los elementos bases del proceso de desarrollo de software es diseñar la Arquitectura de Software. Esencialmente sobre ella se sustentan todos los mecanismos de diseño y representaciones de la estructura general de la aplicación a desarrollar. De la cohesión, utilidad y flexibilidad de los componentes de la arquitectura dependerán la calidad final y la utilidad del software. La correcta definición del estilo arquitectónico a utilizar, patrones y mecanismos de diseño es la raíz de lo anteriormente descrito.

D

Día hábil: día laborable que incluye desde el lunes hasta el viernes y los sábados laborables. No incluye los sábados no laborables, los domingos ni los días feriados del año.

Día natural: son todos los días de la semana incluyendo el sábado y el domingo.

E

Expediente de fase preparatoria: Recibe esta denominación el conjunto de diligencias de instrucción dirigidas a la determinación y esclarecimiento de un hecho delictivo así como de su autor. Científicamente dirigida de forma multilateral, completa y objetiva.

F

Fiscal: Lo perteneciente al físico, y en este sentido se dice leyes fiscales y derechos fiscales, acción fiscal, entre otros. Encargado de promover los intereses del fisco. El funcionario generalmente letrado, que representa y ejerce el ministerio público en los tribunales, defendiendo judicialmente los intereses de la sociedad y del estado.

Fiscal actuante: Es aquel que a los efectos de la tramitación de las impugnaciones tiene la capacidad legal para emitir el pronunciamiento que corresponda.

I

Indentación: Significa mover un bloque de texto hacia la derecha insertando espacios o tabuladores para separarlo del texto adyacente, lo que en el ámbito de la imprenta se ha denominado siempre como sangrado o sangría. Se utiliza para mejorar la legibilidad del código fuente por parte de los programadores y su uso es determinado por los diferentes lenguajes de implementación que lo usan.

P

Plataforma: sistema operativo o sistemas complejos, ya sea de hardware o software, sobre el cual un programa pueda ejecutarse. Ejemplos típicos incluyen: arquitectura de hardware, lenguajes de programación y sus librerías de tiempo de ejecución. Ejemplos de plataformas son PC (Windows) y Macintosh (Mac).

S

Sobreseer: Cesar en una instrucción sumarial y dejar sin curso ulterior un procedimiento. Cesar en el cumplimiento de una obligación.

Sobreseimiento: Acción y efecto de sobreseer. Suspensión por parte de un juez o de un tribunal de un procedimiento judicial, por falta de pruebas o por otra causa.

Sobreseimiento libre: El que por ser evidente la inexistencia de delito o la irresponsabilidad del inculpado, pone término al proceso con efectos análogos a los de la sentencia absolutoria.

Sobreseimiento provisional: El que por deficiencia de pruebas paraliza la causa.

T

Tribunal: Entiéndase por tribunal no solo el lugar en que se administra justicia, sino también por los jueces que componen el órgano colegiado.