Universidad de las Ciencias Informáticas

Facultad 3

Título: Implementación del Módulo Sumario del Proyecto Sistema de Informatización de la Gestión de la Fiscalía.

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

Autores: Javier Fernández Pupo

Osvaldo Suárez Nerey

Tutor: Ing. Yarisleidis Fernández Rivera

Cotutor: Ing. Yunesky del Río García

Curso 2010-2011

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los __ días del mes de _____ del año _____

Javier Fernández Pupo

_____

Firma del Autor

Osvaldo Suárez Nerey

_____

Firma del Autor

Ing. Yarisleidis Fernández Rivera

_____

Firma del Tutor

Ing. Yunesky del Río García

_____

Firma del Cotutor

Agradecimientos

Osva

A mis padres y mi hermana por siempre guiarme y confiar en mí, por tanto sacrificio, cariño y dedicación.

A mi familia que tanto se ha preocupado.

A mi tía Elia que ha sostenido mi familia durante tanto tiempo y tanto ha luchado para que me gradúe.

A mis amigos que han sido mi familia durante este tiempo y han sido mi apoyo en todo momento.

A mi novia que hizo cada momento suyo y logró en mí una mejor persona.

A Yari y el Pichi, por tener tanta calma con nosotros y ser especiales tutores.

Pupo

A mi madre por ser mi guía y principal apoyo.

A mi Tata Lisy por ayudarme tanto y quererme siempre.

A mi padre, mi abuelo Pancho y mi abuela Dela.

A toda mi familia que siempre me ha querido un mundo.

A todos los amigos y amigas que siempre han estado en las buenas y en las malas.

A Yari y el Pichi, por tener tanta calma con nosotros y ser especiales tutores.

A Nico por aguantarme este tiempo.

A mí mismo y a Osva.

Dedicatoria

Osva

A mis padres.

Pupo

A Mamí.

Resumen

Actualmente la situación general de los procesos de gestión de las entidades a escala nacional, está afectada por el control de los datos de forma manual y la existencia de sistemas informáticos que no explotan las posibilidades que brindan las nuevas tecnologías. Entre estos procesos se encuentran los Procesos Penales que se realizan en las fiscalías, los que permiten los trámites y acciones a tomar desde que una persona es denunciada en una unidad de la policía hasta que es convocado ante un tribunal quien dictaminará las medidas a cautelar.

El presente trabajo abarca la implementación del módulo Sumario del subsistema Procesos Penales, del proyecto Sistema de Informatización de la Gestión de la Fiscalía (SIGEF), lo cual es un paso importante para lograr la eficiencia y la calidad en las acciones que se realizan sobre una denuncia. Cuenta con un estudio del arte enmarcado en un problema y un objetivo general a los que se les dará respuesta mediante el cumplimiento de un conjunto de tareas específicas. Se realiza una descripción de las tecnologías y herramientas utilizadas, dando paso a la propuesta de solución. En dicha propuesta se muestran varios diagramas que permiten la descripción de los métodos empleados en la implementación de los casos de usos propuestos por los analistas, los que se han puesto a prueba por el equipo del Centro de Calidad para Soluciones Informáticas (CALISOFT) mediante el uso de pruebas de Caja Negra y se aplican Métricas de Diseño donde se evalúan diferentes atributos de calidad.

Índice

Introducción

En un mundo marcado por el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC), sin las cuales, desarrollarse económica y socialmente sería casi imposible. Cuba ha emprendido la tarea de informatizar la sociedad para de esta forma mejorar la calidad de vida de nuestro pueblo, satisfacer las necesidades de comunicación, conocimiento y mejorar considerablemente los procesos en la economía y gobierno.

En estas circunstancias surge la Universidad de las Ciencias Informáticas, con un paradigma de enseñanza distinto de los demás existentes hasta el momento y con el objetivo de formar profesionales comprometidos con la Revolución, capaces de llevar la vanguardia en este proceso de cambios. Una de las tareas asignadas es la informatización del sector jurídico, específicamente procesos llevados a cabo en la Fiscalía General de la República (FGR).

La FGR es la encargada del control y la preservación de la legalidad sobre la vigilancia del estricto cumplimiento de la Constitución, las leyes y demás disposiciones legales; por tanto se puede decir que gestiona y supervisa el Sistema Judicial y Legal de la República de Cuba, el cual se suscribe en las tradiciones y características del Derecho Continental Europeo del que tomó las correspondientes instituciones judiciales aun cuando, en su elaboración concreta y particular, tuvo en cuenta las condiciones sociales, culturales y jurídicas prevalecientes en la sociedad cubana contemporánea.

La Constitución de la República de Cuba es la norma jurídica de mayor nivel y es la que determina los órganos con capacidad legislativa y los principios y fundamentos del contenido de las leyes.

En la República de Cuba la soberanía reside en el pueblo, del cual dimana todo el poder del Estado. Ese poder es ejercido directamente o por medio de las Asambleas del Poder Popular y demás órganos del Estado que de ellas se derivan, en la forma y según las normas fijadas por la Constitución y las leyes.

Entre los principales problemas que presenta la FGR se encuentran que el sistema judicial produce grandes volúmenes de información sobre los casos que se atienden en las fiscalías del país. Toda esta información se archiva para que en el momento requerido sirva de base para otras investigaciones. Cada vez son más engorrosos los mecanismos para acceder a una información que es imprescindible y sin embargo por su aglomeración es de difícil acceso. Es muy común que se reúnan casos que van a ser procesados y que por la cantidad de estos, no se cuente con el tiempo necesario para ser estudiados a

profundidad y puede darse el caso que en ocasiones la toma de una decisión se convierta en un proceso complejo, lo que provoca que la economía procesal del trabajo de los fiscales no sea óptima.

Esto es provocado fundamentalmente por el déficit de personal en las áreas de las fiscalías y aunque se cuenta con un sistema informático que ayuda en la mejora de la labor de los fiscales, no es suficiente para almacenar toda la información que se maneja y además posee una pobre herramienta para generar documentos por lo que entorpece, en vez de ayudar, el trabajo de los fiscales.

Por lo antes mencionado, se hace evidente la necesidad de que cada fiscal atienda más de un caso a la vez, deba estar pendiente de los plazos de vencimiento establecidos para cada proceso y además, debe realizar la mayor parte del trabajo manual, provocando menor control y supervisión por parte del nivel superior de la tramitación de los procesos en los órganos provinciales.

Para la informatización se ha tomado como política dividir su desarrollo según los procesos fundamentales que se trabajan en la FGR. Producto de esta división surgen ocho subsistemas de desarrollo y formando parte de estos se encuentra el de los Procesos Penales, que tiene como objetivo fundamental la promoción y el ejercicio de la acción penal pública en representación del Estado cuando un hecho es constitutivo de delito. Dicho subsistema está compuesto por módulos, incluyendo el Sumario que comprende los delitos sancionables hasta 1 año de privación de libertad o multa hasta 300 cuotas o ambas. El mismo tiene una gran importancia ya que se encargará del control y la gestión en la toma de decisiones, apelaciones y demás actividades en este proceso.

Teniendo en cuenta lo expuesto anteriormente se plantea el siguiente **problema a resolver**: El desarrollo manual del proceso Sumario dificulta la calidad, control y seguimiento de los procesos que se desarrollan en el área Procesos Penales de las fiscalías.

En consecuencia, el **objeto de estudio** es el Proceso de Desarrollo de Software, y específicamente el **campo de acción** se centra en la Implementación del módulo Sumario del Proyecto Sistema de Gestión Fiscal.

Para ello se ha planteado el **objetivo general**: Desarrollar la implementación y pruebas a un módulo para los procesos sumarios de las fiscalías con el objetivo de mejorar la calidad, el control y seguimiento de los casos que en esta se desarrollan**.** Con los siguientes **objetivos específicos:**

1. Construir un marco teórico.
2. Implementar los requerimientos planteados por la Fiscalía General de la República.

3. Validar la solución propuesta.

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

1. Elaborar el marco teórico de la investigación.
2. Estudiar la descripción de los casos de usos del sistema.
3. Analizar el diseño de clase.
4. Realizar el estudio del modelo de datos.
5. Implementar el módulo Sumario.
6. Validar los resultados.

CAPÍTULO 1

En este capítulo se realizará un estudio del estado del arte haciendo un análisis de las principales tecnologías de desarrollo de software en la actualidad, del cual se espera obtener las principales opciones en cuanto a técnicas de programación (programación estructurada y programación orientada a objetos), herramientas, tecnologías del lado del cliente y del servidor, marcos de trabajo, patrones de arquitectura existentes, gestores de base de datos a utilizar, etc.

CAPÍTULO 2

Se presentará la descripción y el análisis realizado de la solución propuesta a través de la valoración de los artefactos propuesto por los analistas, estándares de código a utilizar, modelo de la Base de Datos (BD), diagramas de clases del diseño y la descripción de las principales funcionalidades a automatizar.

CAPÍTULO 3

En este capítulo se realizarán pruebas de software según los test de unidad con el objetivo de evaluar y validar la solución propuesta haciendo un resumen de los principales resultados obtenidos.

Capítulo 1

1.1. Introducción

La población mundial crece, las sociedades crecen, las necesidades crecen, el mundo como conjunto no se queda atrás, y por supuesto que las tecnologías productos de los avances y estudios del hombre están presentes cada día más en nuestro desarrollo como ente social. ¿Bueno o malo? Eso depende del uso que se le dé. Con todo este crecimiento y el aumento a un acelerado paso de las necesidades de calidad en el trabajo, los avances tecnológicos nos brindan opciones impensables de mejoría en el quehacer diario.

Para lograr este objetivo, se ha hecho necesaria la realización de un profundo análisis del negocio donde se desarrollan los procesos a automatizar y el estudio de las principales tecnologías y herramientas que serán usadas para la implementación de los módulos. A partir de la comparación de las diferentes características que estas presentan, teniendo en cuenta el hecho de que son libres e independientes de plataformas y respondiendo a la necesidad actual de migración a software libre que se lleva a cabo en nuestro país, han sido escogidas con anterioridad por la directiva del proyecto, el arquitecto y los clientes, bajo un riguroso trabajo de selección, basándose en que las mismas se encuentran en constante evolución, por lo que se hace necesario tener un conocimiento avanzado y actualizado de estas a la hora de comenzar a desarrollar aplicaciones informáticas.

1.2. Desarrollo

1.2.1. Tendencias actuales de la tecnología en el desarrollo de software

Después del análisis de las necesidades encontradas en las fiscalías del país, se realizó un estudio de las tendencias y tecnologías actuales posibles a emplear para adoptar la que aporte mayores ventajas en la elaboración de la solución.

Se desarrollará una **Aplicación Web,** sistema informático que los usuarios utilizan accediendo a un servidor Web a través de Internet o de una intranet y sus ventajas más significativas son:

1. **Compatibilidad Multiplataforma**: una misma versión de la aplicación puede correr sin problemas en múltiples plataformas como Windows, Linux, Mac, etc.

2. **Actualización:** las aplicaciones web siempre se mantienen actualizadas y no requieren que el usuario deba descargar actualizaciones y realizar tareas de instalación.
3. **Acceso inmediato y desde cualquier lugar:** las aplicaciones basadas en tecnologías web no necesitan ser descargadas, instaladas y configuradas. Además, pueden ser accedidas desde cualquier computadora conectada a la red en donde se accede a la aplicación.
4. **Menos requisitos de hardware:** este tipo de aplicación no consume o consume muy poco espacio en disco y también es mínimo el consumo de memoria RAM en comparación con los programas instalados localmente. Tampoco es necesario disponer de computadoras con poderosos procesadores ya que la mayor parte del trabajo se realiza en el servidor en donde reside la aplicación.

Seguridad en los datos: los datos se alojan en servidores con sistemas de almacenamiento altamente fiables y se ven libres de problemas que comúnmente sufren los ordenadores de usuarios comunes como virus y roturas de disco.

1.2.2. Técnicas de Programación

1.2.2.1. El paradigma de la programación orientada a objetos.

La programación orientada a objetos (POO) aporta un nuevo enfoque a los retos que se plantean en la programación estructurada cuando los problemas a resolver son complejos. Al contrario que la programación procedimental que enfatiza en los algoritmos, la POO enfatiza en los datos. En lugar de intentar ajustar un problema al enfoque procedimental de un lenguaje, POO intenta ajustar el lenguaje al problema. La idea es diseñar formatos de datos que se correspondan con las características esenciales de un problema. Los lenguajes orientados a objetos combinan en una única unidad o módulo, tanto los datos como las funciones que operan sobre esos datos. Tal unidad se llama **objeto**. Si se desea modificar los datos de un objeto, hay que realizarlo mediante las funciones miembros del objeto. Ninguna otra función puede acceder a los datos. Esto simplifica la escritura, depuración y mantenimiento del programa.

El elemento fundamental de la POO es, como su nombre lo indica, el objeto. Podemos definir un objeto como un conjunto complejo de datos y programas que poseen estructura y forman parte de una organización.

Un objeto puede considerarse como una especie de cápsula dividida en tres partes:

- Relaciones
- Propiedades
- Métodos

Cada uno de estos componentes desempeña un papel totalmente independiente:

Las relaciones permiten que el objeto se inserte en la organización y están formadas esencialmente por punteros a otros objetos.

Las propiedades distinguen un objeto determinado de los restantes que forman parte de la misma organización y tiene valores que dependen de la propiedad de que se trate. Las propiedades de un objeto pueden ser heredadas a sus descendientes en la organización.

Los métodos son las operaciones que pueden realizarse sobre el objeto, que normalmente estarán incorporados en forma de programas (código) que el objeto es capaz de ejecutar y que también pone a disposición de sus descendientes a través de la herencia. (McGraw-Hill)

1.2.2.2. Programación estructurada

La programación estructurada consiste en escribir un programa de acuerdo con unas reglas y un conjunto de técnicas. Las reglas son: el programa tiene un diseño modular, los módulos son diseñados descendentemente, cada módulo de programa se codifica usando tres estructuras de control (secuencia, selección e iteración); es el conjunto de técnicas que han de incorporar: recursos abstractos; diseño descendente y estructuras básicas de control.

Descomponer un programa en términos de recursos abstractos consiste en descomponer acciones complejas en términos de acciones más simples capaces de ser ejecutadas en una computadora.

El diseño descendente se encarga de resolver un problema realizando una descomposición en otros más sencillos mediante módulos jerárquicos. El resultado de esta jerarquía de módulos es que cada módulo se

refina por los de nivel más bajo que resuelven problemas más pequeños y contienen más detalles sobre los mismos.

Las estructuras básicas de control sirven para especificar el orden en que se ejecutarán las distintas instrucciones de un algoritmo. Este orden de ejecución determina el flujo de control del programa.

La programación estructurada significa:

- El programa completo tiene un diseño modular.
- Los módulos se diseñan con metodología descendente (puede hacerse también ascendente).
- Cada módulo se codifica utilizando las tres estructuras de control básicas: secuenciales, selectivas y repetitivas
- Ausencia total de sentencias **ir** → **a** (*goto*).
- Estructuración y modularidad son conceptos complementarios (se solapan).

Esta técnica incorpora:

Diseño descendente (top-down): el problema se descompone en etapas o estructuras jerárquicas.

Recursos abstractos (simplicidad): consiste en descomponer las acciones complejas en otras más simples capaces de ser resueltas con mayor facilidad.

Estructuras básicas

Estructuras secuénciales: cada acción sigue a otra acción secuencialmente. La salida de una acción es la entrada de otra.

Estructuras selectivas: en estas estructuras se evalúan las condiciones y en función del resultado de las mismas se realizan unas acciones u otras. Se utilizan expresiones lógicas.

Estructuras repetitivas: son secuencias de instrucciones que se repiten un número determinado de veces.

Las principales **ventajas** de la programación estructurada son:

1. Los programas son más fáciles de entender
2. Se reduce la complejidad de las pruebas
3. Aumenta la productividad del programador.

4. Los programas queden mejor documentados internamente. (McGraw-Hill)

Se puede afirmar que el tipo de programación más idóneo para la implementación de los módulos en cuestión, es la POO, brindándonos la ventaja de reutilización de código, ya que un objeto es más fácil de reutilizarse en tanto su responsabilidad sea mejor definida y más concreta, siendo este paradigma el que más se adapta a las condiciones que son necesarias para el cumplimiento del objetivo de trabajo: la construcción de un software que satisfaga las condiciones exigidas por el cliente.

1.2.3. Tecnologías y plataformas de desarrollo

1.2.3.1. Tecnologías del lado del servidor

1.2.3.1.1. Java

Hoy en día, en un mundo altamente competitivo, Java se ha convertido en una de las tecnologías más seguras para el desarrollo de sitios web y de programas. Las primeras ventajas de Java son independencia de la plataforma y disponibilidad fácil para los usuarios pues es una fuente abierta.

Java es un lenguaje de programación orientada a objeto y fue pensado para servir como nueva manera de manejar complejidad de software. Java refiere a un número de productos y de especificaciones de los programas informáticos de Sun Microsystems que juntos proporcionan un sistema para el despliegue y uso de software. Java se utiliza en una gran variedad de plataformas computacionales para diferentes dispositivos, teléfonos móviles, servidores web, etc. (Alvarez, 2001)

Ventajas de Java:

- Es una fuente abierta, así que los usuarios no tienen que luchar con los impuestos sobre patentes pasado cada año.
- Independiente de la plataforma.
- El poder de Java API sea alcanzada fácilmente por los reveladores.
- Java realiza la colección de basura de las ayudas, así que la gerencia de memoria es automática.
- Desarrollo de aplicaciones web dinámicas.
- Permite la creación de programas modulares y códigos reutilizables. (Alvarez, 2001)

1.2.3.1.2. PHP

1. **Velocidad**: Velocidad de ejecución, la cual es importante, sino además no proporcionar demoras en la máquina. Por esta razón no debe requerir demasiados recursos de sistema. PHP se integra muy bien junto a otros softwares, especialmente bajo ambientes Unix, cuando se configura como módulo de Apache, está listo para ser utilizado.
2. **Estabilidad**: La velocidad no sirve de mucho si el sistema pierde su conexión cada cierta cantidad de ejecuciones. Ninguna aplicación es 100% libre de bugs, pero teniendo de respaldo una increíble comunidad de programadores y usuarios es mucho más difícil para lo bugs sobrevivir. PHP utiliza su propio sistema de administración de recursos y dispone de un sofisticado método de manejo de variables, conformando un sistema robusto y estable.
3. **Seguridad**: El sistema debe poseer protecciones contra ataques. PHP provee diferentes niveles de seguridad, estos pueden ser configurados desde el archivo *.ini*.
4. **Simplicidad**: Se les debe permitir a los programadores generar código productivamente en el menor tiempo posible. Usuarios con experiencia en C y C++ podrán utilizar PHP rápidamente. (Marley)

Otra característica a tener en cuenta es la conectividad. PHP dispone de una amplia gama de librerías, y agregarle extensiones es muy fácil. Esto le permite al PHP ser utilizado en muchas áreas diferentes, tales como encriptado, gráficos, XML y otras.

Ventajas adicionales de PHP

1. PHP corre en (casi) cualquier plataforma utilizando el mismo código fuente, pudiendo ser compilado y ejecutado en algo así como 25 plataformas, incluyendo diferentes versiones de Unix, Windows (95,98,NT,ME,2000,XP,Win7) y Macs. Como en todos los sistemas se utiliza el mismo código base, los scripts pueden ser ejecutados de manera independiente al sistema operativo. (Marley)
2. La sintaxis de PHP es similar a la del C, por esto cualquiera con experiencia en lenguajes del estilo C podrá entender rápidamente PHP. (Marley)
3. PHP es completamente expandible. Está compuesto de un sistema principal (escrito por Zend), un conjunto de módulos y una variedad de extensiones de código. (Marley)

4. Puede interactuar con muchos motores de bases de datos tales como MySQL, MS SQL, Oracle, Informix, PostgreSQL, y otros muchos. (Marley)
5. Una gran variedad de módulos cuando un programador PHP necesite una interfaz para una librería en particular, fácilmente podrá crear una API para esta. Algunas de las que ya vienen implementadas permiten manejo de gráficos, archivos PDF, Flash, Cybercash, calendarios, XML, IMAP, POP, etc. (Marley)
6. Rapidez. PHP generalmente es utilizado como modulo de Apache, lo que lo hace extremadamente veloz. Está completamente escrito en C, así que se ejecuta rápidamente utilizando poca memoria. (Marley)
7. PHP es Open Source, lo cual significa que el usuario no depende de una compañía específica para arreglar cosas que no funcionan, además el usuario no está forzado a pagar actualizaciones anuales para tener una versión que funcione. (Marley)

Teniendo en cuenta las características de los lenguajes de programación estudiados, se puede concluir que con el uso del lenguaje Java en aplicaciones web el acceso a las páginas es más lento debido a la sobrecarga que genera la máquina virtual, se necesita mejoras de recursos de hardware y es imposible para el país acceder a la última versión de la máquina virtual por estar bloqueados. Por lo antes expuesto se puede asegurar que la elección realizada por la directiva del proyecto Sistema de Informatización de la Gestión de la Fiscalía ha sido la correcta ya que el lenguaje PHP es el que más se ajusta a las necesidades del proyecto. Es el más adecuado para el desarrollo de aplicaciones web porque además de ser libre, es muy rápido, contiene una biblioteca nativa de funciones sumamente amplia y no requiere definición de tipos de variables lo que beneficia al desarrollador en la implementación.

1.2.4. Tecnologías del lado del cliente

1.2.4.1. HTML

Es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). HTML

puede describir hasta un cierto punto la apariencia de un documento, y puede incluir un *script* (por ejemplo JavaScript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML. Es usado para referirse al contenido del tipo de MIME text/html o todavía más ampliamente como un término genérico para el HTML, ya sea en forma descendida del XML (como XHTML 1.0 y posteriores) o en forma descendida directamente de SGML. (Valdés, 2007)

Ventajas:

- Sencillo que permite describir hipertexto.
- Texto presentado de forma estructurada y agradable.
- Archivos pequeños.
- Despliegue rápido.
- Lenguaje de fácil aprendizaje.
- Lo admiten todos los exploradores. (Valdés, 2007)

Desventajas:

- Lenguaje estático.
- La interpretación de cada navegador puede ser diferente.
- Guarda muchas etiquetas que pueden convertirse en “basura” y dificultan la corrección.
- El diseño es más lento.
- Las etiquetas son muy limitadas. (Valdés, 2007)

1.2.4.2. Tecnología AJAX

AJAX no es una tecnología. Es realmente muchas tecnologías, cada una floreciendo por su propio mérito, uniéndose en poderosas nuevas formas. AJAX incorpora:

- Presentación basada en estándares usando XHTML y CSS.
- Exhibición e interacción dinámicas usando el Document Object Model.
- Intercambio y manipulación de datos usando XML y XSLT.
- Recuperación de datos asincrónica usando XML.
- Uso de JavaScript. (TECOnSITE)

El modelo clásico de aplicaciones Web funciona de esta forma: la mayoría de las acciones del usuario en la interfaz disparan un requerimiento HTTP al servidor web. El servidor efectúa un proceso (recopila información, procesa números), y devuelve una página HTML al cliente. Este es un modelo adaptado del uso original de la Web como un medio perfecto para el hipertexto, y no necesariamente buena para las aplicaciones de software. (TECOnSITE)

A pesar de que todas las tecnologías antes expuestas son adecuadas para el desarrollo de las páginas clientes, se ha decidido por parte de la directiva del proyecto SIGEF usar la Tecnología Ajax ya que reúne gran parte o la totalidad de las características de las principales tecnologías del lado del cliente: HTML, XML, CSS y JavaScript. Es el que más se ajusta a las condiciones impuestas por los clientes ya que es de fácil manejo y sencillo de aprender. Ajax es completamente compatible con PHP y los frameworks que a él pertenecen y no sobrecarga las aplicaciones web; lo que permite dar mayor velocidad a la ejecución de las mismas. Esta será gestionada mediante el Prototype que es una librería de JavaScript muy completa y a su vez, amplía las posibilidades del lenguaje de programación, añade nuevas funcionalidades y ofrece nuevos mecanismos para la manipulación de los elementos DOM. Los helpers de Ajax dependen de Prototype, por lo que la librería del mismo, se incluye automáticamente cada vez que se utiliza cualquiera de ellos.

1.2.5. Herramientas CASE

1.2.5.1. ERwin

PLATINUM ERwin es una herramienta para el diseño de base de datos, que brinda productividad en su diseño, generación, y mantenimiento de aplicaciones. Desde un modelo lógico de los requerimientos de información, hasta el modelo físico perfeccionado para las características específicas de la base de datos diseñada, además ERwin permite visualizar la estructura, los elementos importantes, y optimizar el diseño de la base de datos. Genera automáticamente las tablas y miles de líneas de procedimientos almacenados y triggers para los principales tipos de base de datos.
ERwin hace fácil el diseño de una base de datos. Los diseñadores de bases de datos sólo apuntan y pulsan un botón para crear un gráfico del modelo E-R (Entidad _ relación) de todos sus requerimientos de datos y capturar las reglas de negocio en un modelo lógico, mostrando todas las entidades, atributos, relaciones, y llaves importantes. La migración automática garantiza la integridad referencial de la base de

datos. ERwin establece una conexión entre una base de datos diseñada y una base de datos, permitiendo transferencia entre ambas y la aplicación de ingeniería reversa. Usando esta conexión, ERwin genera automáticamente tablas, vistas, índices, reglas de integridad referencial (llaves primarias, llaves foráneas), valores por defecto y restricciones de campos y dominios. (Sub-Jefatura de Informática, 1999)

ERwin soporta principalmente bases de datos relacionales SQL y bases de datos que incluyen Oracle, Microsoft SQL Server, Sybase. El mismo modelo puede ser usado para generar múltiples bases de datos, o convertir una aplicación de una plataforma de base de datos a otra. (Sub-Jefatura de Informática, 1999)

1.2.5.2. Visual Paradigm

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. También proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Presenta licencia gratuita y comercial. Es fácil de instalar y actualizar y compatible entre ediciones. (Visual Paradigm)

Características principales:

1. Soporte de UML versión 2.1.
2. Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
3. Modelado colaborativo con CVS y Subversion (control de versiones).
4. Ingeniería inversa Java, C++, Esquemas XML, XML, NET exe/dll, CORBA IDL.
5. Generación de código - Modelo a código, diagrama a código.
6. Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
7. Diagramas EJB - Visualización de sistemas EJB.

8. Generación de código y despliegue de EJB - Generación de beans para el desarrollo y despliegue de aplicaciones.
9. Diagramas de flujo de datos.
10. Soporte ORM - Generación de objetos Java desde la base de datos.
11. Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
12. Generador de informes.
13. Importación y exportación de ficheros XMI.
14. Integración con Visio - Dibujo de diagramas UML con plantillas de Microsoft Visio. (Visual Paradigm)

Aunque son increíbles las posibilidades que brinda ERwin en cuanto al diseño de las tablas de la base de datos y a las comodidades en cuanto al manejo del Modelo Entidad-Relación, es evidente la completitud de las tareas que aporta Visual Paradigm, que además es compatible con el IDE seleccionado anteriormente y maneja todas las fases de los ciclos de vida de RUP; metodología usada para guiar los pasos de la construcción del software en cuestión, por lo que se ha hecho una correcta elección por parte de la directiva del proyecto SIGEF al seleccionar esta herramienta.

1.2.6. Servidor Web

1.2.6.1. Internet Information Server (IIS)

El servidor Internet Information Servers un servidor de páginas Web, FTP y Gopher que soporta los protocolos HTTP, FTP y GOPHER. Este servidor tiene varias características interesantes:

1. Es muy fácil de instalar. La instalación y puesta en marcha es cuestión de minutos y la publicación de páginas y ficheros es también igual de simple.
2. Trabaja como un servicio NT, por lo que se integra perfectamente con el resto de servicios de NT.
3. Utiliza una herramienta de configuración gráfica muy sencilla, que ayuda en las labores de administración y seguridad del servidor.
4. Permite la administración remota vía Web, utilizando para ello un navegador (por ejemplo, el Internet Explorer).

5. Permite la publicación mediante herramientas Web, como el FrontPage y similares, o mediante la simple copia de ficheros sobre la red.
6. Permite utilizar mecanismos de seguridad avanzados, como Autentificación cifrada de NT.
7. Podemos crear páginas activas y scripts, soportando Active Server Pages (ASP), protocolos CGI y ISAPI, que añaden dinamismo a nuestras páginas. (IIS)

1.2.6.2. Apache

Apache es el servidor web hecho por excelencia, su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. (Apache)

Apache es una muestra, al igual que el sistema operativo Linux, de que el trabajo voluntario y cooperativo dentro de Internet es capaz de producir aplicaciones de calidad profesional, difíciles de igualar.

1. Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal. (Apache)
2. Apache es una tecnología gratuita, open source. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto. Esto le da una transparencia a este software de manera que si se quiere ver qué es lo que se instala como servidor, se puede saber, sin ningún secreto, sin ninguna puerta trasera. (Apache)
3. Apache es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor Web Apache. Actualmente existen muchos módulos para Apache que son adaptables a este, y están ahí para que se instalen cuando se necesiten. Otra cosa importante es que cualquiera que posea una experiencia decente en la programación de C o Perl puede escribir un módulo para realizar una función determinada. (Apache)
4. Apache trabaja con gran cantidad de Perl, PHP y otros lenguajes de script. Perl destaca en el mundo del script y Apache utiliza su parte del pastel de Perl tanto con soporte CGI como con soporte mod perl. También trabaja con Java y páginas jsp. Teniendo todo el soporte que se necesita para tener páginas dinámicas. (Apache)

5. 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. (Apache)
6. Tiene una alta configurabilidad 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 el servidor. (Apache)

A pesar de las grandes ventajas que proporciona Internet Information Server (IIS) en cuanto a seguridad y servicios web; es privativo, lo que ha llevado a la directiva del proyecto SIGEF a utilizar como servidor web el Apache, que además de ser seguro, es uno de los más utilizados en el mundo puesto que es tan potente como flexible, es libre y de código abierto y además posee módulos configurables de forma tal que se pueda decidir cuáles de estos serán ejecutados es el servidor.

1.2.7. IDE de desarrollo

1.2.7.1. Zend Studio

Zend Studio es la nueva generación de Entorno de Desarrollo de Aplicaciones (IDE) PHP, con grado profesional. Ha sido diseñado para maximizar la productividad en el desarrollador, al permitirle desarrollar y mantener el código más rápido, resolver rápidamente problemas de aplicaciones y mejorar la colaboración en equipo. Características como reestructuración de código fuente, generación de código, asistencia de código y análisis semántico se combinan para permitir un rápido desarrollo de aplicaciones. (taller.web)

Zend Studio mejora la productividad con características de desarrollo orientada a equipos, tales como soporte para el manejo de configuración fuente (CVS, SVÑ) y montajes de proyectos compartidos. Estas características ayudan a mejorar la colaboración entre miembros de proyecto. (taller.web)

1.2.7.2. Eclipse PDT

Es considerado tan potente como popular. Incorpora un sinfín de utilidades para simplificar la labor de los programadores. Aparte de ser un entorno de desarrollo muy completo, una de las particularidades más interesantes para la comunidad es que es de código libre y gratuito. Posee editor sensible al contexto, el cual resalta, asiste y autocompleta código. (Eclipse)

Entre las características principales se encuentran:

- Integración con el modelo del proyecto Eclipse, que permite para inspeccionar el uso de las vistas del contorno del fichero y del proyecto, así como la nueva vista PHP Explorer.
- Soporte para el debug incremental del código de PHP
- Extensos frameworks y APIs que permiten a los desarrolladores e ISVs (vendedores de software independientes) fácilmente extender PDT para crear nuevas e interesantes herramientas orientadas al desarrollo de PHP. (Eclipse)

Aprovechando las comodidades que brinda Eclipse PDT, la directiva del proyecto de SIGEF ha decidido utilizar el mismo como entorno de desarrollo de la aplicación que se desea desarrollar. Aunque es un poco complicado configurar el debug PHP si no se tiene la receta adecuada para hacerlo funcionar, es una aplicación que brinda una gran cantidad de opciones y posibilidades. Corre en plataforma libre, es compatible con el lenguaje seleccionado y se le pueden añadir un sinfín de plugins que ayudan al desarrollo de aplicaciones web. De Zend Studio se puede decir que pese a las grandes ventajas que proporciona, requiere Licencia de pago y no incluye editor visual HTML.

1.2.8. Sistema Gestor de Base de Datos

1.2.8.1. MySQL

MySQL es un sistema de gestión de bases de datos relacional, licenciado bajo la GPL de la GNU. Su diseño multi-hilo le permite soportar una gran carga de forma muy eficiente. MySQL fue creada por la empresa sueca MySQL AB, que mantiene el copyright del código fuente del servidor SQL, así como también de la marca. (Definición ABC, 2009)

Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración. (Definición ABC, 2009)

1. Las principales características de este gestor de bases de datos son las siguientes:
2. Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multi-hilo.
3. Soporta gran cantidad de tipos de datos para las columnas.
4. Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc.).
5. Gran portabilidad entre sistemas.
6. Soporta hasta 32 índices por tabla.
7. Gestión de usuarios y passwords, manteniendo un muy buen nivel de seguridad en los datos. (Definición ABC, 2009)

1.2.8.2. PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES, de la universidad de Berkeley. El director de este proyecto es el profesor Michael Stonebraker, y fue patrocinado por Defense Advanced Research Projects Agency (DARPA), el Army Research Office (ARO), el National Science Foundation (NSF), y ESL, Inc. (ArPUG)

Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido, más tarde en otros sistemas de gestión comerciales. PostgreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto, PostgreSQL no es un sistema de gestión de bases de datos puramente orientado a objetos. (ArPUG)

A continuación se enumeran las principales características de este gestor de bases de datos:

1. Implementación del estándar SQL92/SQL99.

2. Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP...), cadenas de bits, etc. También permite la creación de tipos propios.
3. Incorpora una estructura de datos array.
4. Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.
5. Permite la declaración de funciones propias, así como la definición de disparadores.
6. Soporta el uso de índices, reglas y vistas.
7. Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
8. Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.(ArPUG)

Aunque son evidentes las ventajas de ambos gestores de bases de datos, la dirección del proyecto SIGEF se ha inclinado por PostgreSQL quien será gestionado por una aplicación gráfica llamada PGAdmin III que es una de las más completas y populares con licencia Open Source. Está escrita en C++ y utiliza la librería gráfica multiplataforma wxWidgets, permitiendo que se pueda usar en sistemas operativos como: GNU/Linux, FreeBSD, Solaris, Mac OS y Windows. 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.

1.2.9. SVN (SubVersion)

1.2.9.1. CVS

Es un sistema de control de versiones donde se puede registrar el historial de todos los archivos fuentes y documentos. Contribuye a la calidad de un buen producto y corre en plataformas libres. Mediante sentencias de comandos se puede proporcionar el registro de todas las operaciones realizadas sobre un documento en específico y logra hacer cumplir las políticas de seguridad establecidas para la aplicación. Permite que las unidades de trabajo funcionen como un solo equipo de desarrollo aunque no necesariamente trabajen como tal. El historial de todas las versiones se almacena en un único servidor de

versiones. Proporciona una base de datos de módulos flexibles y le asigna simbólicamente nombres a los componentes del software. Se ejecutan en la mayoría de las variantes de Unix y de los clientes NT/95 para Windows. (Leopoldo, 19)

1.2.9.2. TortoiseSVN

TortoiseSVN es un cliente gratuito de código abierto para el sistema de control de versiones SubVersion.

Algunas Características:

1. Integración con el shell de Windows:
   TortoiseSVN se integra perfectamente en el shell de Windows (por ejemplo, el explorador). Esto significa que puede seguir trabajando con las herramientas que ya conoce, y que no tiene que cambiar a una aplicación diferente cada vez que necesite las funciones del control de versiones.
2. Íconos sobre impresionados:
   El estado de cada carpeta y fichero versionado se indica por pequeños íconos sobre impresionados. De esta forma, puede ver fácilmente el estado en el que se encuentra su copia de trabajo.
3. Fácil acceso a los comandos de SubVersion:
   Todos los comandos de SubVersion están disponibles desde el menú contextual del explorador. TortoiseSVN añade su propio submenú allí.
4. Versionado de carpetas:
   CVS sólo controla la historia de ficheros individuales, pero SubVersion implementa un sistema "virtual" de ficheros versionados que sigue la pista de los cambios en todos los árboles de directorios en el tiempo. Los ficheros y los directorios están versionados. Como resultado, hay comandos reales en el lado del cliente como mover y copiar que operan en ficheros y directorios.
5. Manejo de datos consistente:
   SubVersion expresa las diferencias entre ficheros usando un algoritmo de diferenciación binario, que funciona exactamente igual tanto en ficheros de texto (legibles por los humanos) como en ficheros binarios (que no son legibles por nosotros). Ambos tipos de ficheros se almacenan

igualmente comprimidos en el repositorio, y las diferencias se transmiten en ambas direcciones por la red.

6. Extensibilidad:
   SubVersion no tiene lastre histórico; está implementado como una colección de librerías C compartidas con APIS bien definidas. Esto hace que SubVersion sea fácil de mantener y se pueda utilizar por otras aplicaciones y lenguajes. (Najera)

Después de un amplio estudio de los diferentes tipos de controladores de versiones o SubVersion como también suele llamarse, se corrobora la decisión tomada por la directiva del proyecto SIGEF de escoger el TortoiseSVN pues además de correr en plataforma libre posee un mejor dominio de las versiones y un mayor control del trabajo que se está realizando sobre cada uno de los elementos de configuración brindando soporte a todas las operaciones que se realizan sobre los mismos.

1.2.10. Framework

1.2.10.1. Zend Framework

Probablemente, el framework más conocido, y el más utilizado profesionalmente. Es muy desacoplado, por lo que mucha gente lo considera una librería de componentes más que un framework. Por otra parte, gracias a esto, podemos utilizarlo en conjunto con otros frameworks. Tiene detrás a Zend, una de las empresas más importantes de la comunidad PHP, y gran contribuidora a su código desde sus primeras versiones. Siempre ha habido quejas sobre su curva de aprendizaje y el tiempo necesario para comenzar un nuevo proyecto, aunque eso mejoró un poco con los componentes RAD (Rapid Application Development). Es muy recomendado para proyectos grandes. (Duque)

Ventajas

- Reduce el "time to market" de las aplicaciones, permitiendo ofrecer presupuestos más ajustados.
- Estandariza los procesos más frecuentes, dotándolos de gran robustez.
- Facilita el mantenimiento de las aplicaciones.

- Ofrece muchas facilidades para el acceso a recursos avanzados (Web services securizados, por ejemplo) que de otro modo resultan bastante más costosos de desarrollar.
- A diferencia de otros frameworks, es posible utilizarlo en modo "desacoplado", es decir, aquellas clases o componentes que sean necesarios en cada proyecto, sin arrastrar todo el framework detrás para cualquier pequeña necesidad.
- Tiene el respaldo de la propia ZEND, creadora de PHP, lo que asegura su continuidad futura tanto como la del propio lenguaje PHP. (Duque)

1.2.10.2. Symfony framework

Symfony es uno de los frameworks PHP más populares entre los usuarios y las empresas, ya que permite que los programadores sean mucho más productivos a la vez que crean código de más calidad y más fácil de mantener. Symfony es maduro, estable, profesional y está muy bien documentado. (Potencier, 2007)

Symfony simplifica al máximo el desarrollo de aplicaciones web profesionales con PHP, utilizando las mejores prácticas y los patrones de diseño más importantes. Symfony incorpora muchas de las ideas del RAD (“desarrollo rápido de aplicaciones”) para conseguir que la programación de las aplicaciones sea lo más productiva, correcta y divertida posible. Es un enorme conjunto de herramientas y utilidades que simplifican el desarrollo de las aplicaciones web. (Potencier, 2007)

Symfony emplea el tradicional patrón de diseño MVC (modelo-vista-controlador) para separar las distintas partes que forman una aplicación web. El modelo representa la información con la que trabaja la aplicación y se encarga de acceder a los datos. (Potencier, 2007)

La vista transforma la información obtenida por el modelo en las páginas web a las que acceden los usuarios. El controlador es el encargado de coordinar todos los demás elementos y transformar las peticiones del usuario en operaciones sobre el modelo y la vista. (Potencier, 2007)

Algunos datos importantes sobre Symfony: la primera versión se publicó en Octubre de 2005; su licencia es de tipo software libre; ha sido desarrollado por una empresa francesa llamada Sensio Labs. (Potencier, 2007)

Después de haberse realizado un estudio de algunos de los principales framework para PHP, la directiva del proyecto SIGEF se ha inclinado por la utilización de Symfony, ya que este es independiente de plataforma, corre en el IDE de desarrollo seleccionado anteriormente, es compatible con la mayoría de los gestores de bases de datos y posee una amplia bibliografía y foros en español a los que se les puede hacer referencias en cualquier situación o duda.

1.2.11. Patrones de arquitectura

1.2.11.1. Estilo Basado en Capas

El Patrón de arquitectura por capas es una de las técnicas más comunes que los arquitectos de software utilizan para dividir sistemas de software complicados. Al pensar en un sistema en términos de capas, se imaginan los principales subsistemas de software ubicados de la misma forma que las capas de un pastel, donde cada capa descansa sobre la inferior. En este esquema la capa más alta utiliza varios servicios definidos por la inferior, pero la última es inconsciente de la superior. Además, normalmente, cada capa oculta las capas inferiores de las siguientes superiores a esta. (Arevalo, 2010)

1. Los beneficios de trabajar un sistema en capas son:

   - Se puede entender una capa como un todo, sin considerar las otras.
   - Las capas se pueden sustituir con implementaciones alternativas de los mismos servicios básicos.
   - Se minimizan dependencias entre capas.
   - Las capas posibilitan la estandarización de servicios.
   - Luego de tener una capa construida, puede ser utilizada por muchos servicios de mayor nivel. (Arevalo, 2010)

1.2.11.2. Patrón Modelo-Vista-Controlador (MVC)

Modelo Vista Controlador (MVC). Es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El estilo de llamada

y retorno MVC, se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista. (Potencier, 2007)

- El Modelo es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo. (Potencier, 2007)
- La Vista es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa con el Modelo a través de una referencia al propio Modelo. (Potencier, 2007)
- El Controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo. (Potencier, 2007)

1.2.12. Pruebas de calidad

1.2.12.1. Prueba de Caja Blanca

Consiste en realizar pruebas para verificar qué líneas específicas de código funcionan tal como está definido. También se le conoce como prueba de caja-transparente.

La prueba de la caja blanca es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivar los casos de prueba. (Huanca)

Las pruebas de caja blanca intentan garantizar que:

- Se ejecutan al menos una vez todos los caminos independientes de cada módulo.
- Se utilizan las decisiones en su parte verdadera y en su parte falsa.

- Se ejecuten todos los bucles en sus límites.
- Se utilizan todas las estructuras de datos internas.
- Para esta prueba se consideran tres importantes puntos.
- Conocer el desarrollo interno del programa, determinante en el análisis de coherencia y consistencia del código.
- Considerar las reglas predefinidas por cada algoritmo.
- Comparar el desarrollo del programa en su código con la documentación pertinente. (Huanca)
- La prueba se divide en dos partes que son:

El análisis estático.

- ✓ Análisis estático Manual.
- ✓ Inspección: Determina si el código esta completo y correcto, como también las especificaciones.
- ✓ Walkthrough: Interrelación informal entre testers, creadores y usuarios del sistema.
- ✓ Análisis estático Automático.
- ✓ Verificación estática: Compara los valores generados por el programa con los rangos de valores predefinidos haciendo una descripción del funcionamiento de los procedimientos en términos booleanos determinando los puntos de falla.
- ✓ Ejecución simbólica: Hace un seguimiento de la comunicación entre funciones, módulos, aplicaciones, luego de que todas las partes hayan sido verificadas por separado. (Huanca)

El análisis dinámico

Para esto se cuenta con diferente tipo de herramientas.

- ✓ Análisis de cobertura: Examina las extensiones del código, haciendo una caja blanca por modulo.
- ✓ Trafico: Sigue todos los caminos de comunicación entre módulos guardando los valores de las variables en cada uno de ellos.
- ✓ Simulador: Simula partes del sistema para el cual el hardware no está habilitado.
- ✓ Sintonía: Testea los recursos utilizados durante la ejecución del programa.

- ✓ Prueba de certeza: Examina las construcciones lógicas del programa. (Huanca)

1.2.12.2. Prueba de Caja Negra

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

Los casos de prueba de la caja negra pretenden demostrar que:

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

A continuación se derivan conjuntos de condiciones de entrada que utilicen todos los requisitos funcionales de un programa.

Las pruebas de caja negra pretenden encontrar estos tipos de errores:

- Funciones incorrectas o ausentes.
- Errores en la interfaz.
- Errores en estructuras de datos o en accesos a bases de datos. (Gill)

Externas.

- Errores de rendimiento.
- Errores de inicialización y de terminación. (Gill)

Para la validación de los resultados que se obtendrán después de haber desarrollado el software, la directiva del proyecto ha decidido optar por la liberación del software por parte de CALISOFT mediante la realización de pruebas de Caja Negra ya que así se asegurará la completitud de los requisitos funcionales,

parte fundamental en el desarrollo de un software. Se ha determinado que es un poco dificultoso el desarrollo de pruebas de Caja Blanca ya que el framework que se está utilizando es poco conocido por las personas de calidad y no es muy entendible la ejecución del código, mientras que usando las pruebas de Caja Negra, se harían una serie de comparaciones entre los resultados obtenidos y haciendo uso de los datos entrados al sistema y la respuesta del mismo.

1.3. Conclusiones parciales

En este capítulo se ha introducido toda la información necesaria para la comprensión de los pasos seguidos por la directiva del proyecto SIGEF, el arquitecto y los clientes, en cuanto a la selección de las tecnologías y herramientas a utilizar. Se han mencionado los principales candidatos a utilizar y el motivo del por qué fueron escogidos en cada caso. Las herramientas y tecnologías escogidas fueron POO, PHP, AJAX, Visual Paradigm, Apache, Eclipse, PostgreSQL, TortoiseSVN, Symfony y las Pruebas de Caja Negra para validar la implementación. Esta comparativa sirvió para comprender la importancia de realizar una buena selección evitando así el uso de herramientas obsoletas que resten prestigio al trabajo realizado. Después de estos pasos, se han madurado las condiciones para dar respuesta a la problemática anteriormente planteada y cumplir así el objetivo trazado.

Capítulo 2

2.1. Introducción

En este capítulo se delimitan varios puntos importantes en la implementación de los componentes. Se comienza exponiendo los requisitos funcionales identificados por los analistas. Se muestra cómo está concebida la arquitectura y las posibilidades que proporcionan los marcos de trabajo utilizados en la programación de la aplicación, con el objetivo de facilitar la comprensión del funcionamiento de los componentes implementados. Se determina la complejidad a un algoritmo no trivial, se hace una descripción de clases y se muestran algunas de las operaciones utilizadas.

2.2. Desarrollo

2.2.1. Valoración de los artefactos propuestos por los analistas

En un proceso de desarrollo de software es de gran importancia la descripción de los requisitos funcionales, los que son brindados por el analista de sistema y facilitan un mejor entendimiento de los procesos a desarrollar, permitiendo comprender con profundidad el problema en cuestión y facilitando una mejor identificación de las clases y funcionalidades que serán implementadas. La especificación de requisitos, es la base que permite verificar si se alcanzaron o no los objetivos establecidos en el proyecto debido a que son un reflejo detallado de las necesidades de los clientes o usuarios del sistema. Un requisito según el Standard Glossary of Software Engineering Terminology de la IEEE8 se puede definir como una:

- Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
- 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.
- Una representación documentada de una condición o capacidad como en los casos anteriores.

(Álvarez, 2009)

2.2.1.1. Requisitos Funcionales

Los **requisitos funcionales** propuestos por los analistas posibilitaron crear una entrada apropiada y un punto de partida para las actividades de implementación:

1. Crear el Libro de Control.

El sistema permite crear el libro de control de denuncias con sus campos determinados.

2. Actualizar automáticamente el Libro de Control por entrada inicial de atestado.

El sistema permite actualizar automáticamente en el libro de control de denuncias los datos requeridos de cualquier atestado que sea creado en el sistema.

3. Mostrar el Libro de Control.

El sistema permite mostrar el libro de control de denuncias.

4. Crear el Atestado.

El sistema permite crear un atestado recibido de la PNR.

5. Mostrar Atestado.

El sistema permite visualizar los datos del atestado.

6. Adjuntar Diligencia escaneada al Atestado.

El sistema permite adjuntar las actuaciones a un atestado dado.

7. Distribuir automáticamente el Atestado al Fiscal correspondiente.

El sistema permite distribuir automáticamente por parte del sistema el atestado que se acaba de crear al fiscal que atiende la unidad de la PNR de la que proviene.

8. Actualizar en la interfaz Proceso Sumario el atestado con acusados detenidos.

El sistema permite actualizar en la interfaz Proceso Sumario el nuevo atestado con detenidos, es decir, se muestra el atestado para conocimiento del fiscal dentro de ese menú.

9. Actualizar en la interfaz Proceso Sumario el atestado sin acusados detenidos.

El sistema permite actualizar en la interfaz Proceso Sumario el nuevo atestado sin detenidos, es decir, se muestra el atestado para conocimiento del fiscal dentro de ese menú.

10. Crear Historial del Atestado.

El sistema permite crear automáticamente el historial del atestado.

11. Mostrar Historial de un Atestado.

El sistema permite visualizar el historial de un atestado.

12. Actualizar automáticamente datos del Historial de un Atestado por creación de Factura para salida de la Fiscalía.

El sistema permite actualizar el historial del atestado al que se haga referencia en una Factura creada ya sea por salida del mismo a la PNR o al Tribunal, siempre aclarando el destino del atestado correspondiente, es decir, si se emite concretamente un documento Factura, se verifica a que atestado(s) hace refencia, y se le(s) actualiza su historial con la salida correpondiente ya sea para el Tribunal o la PNR.

13. Actualizar automáticamente datos del Historial de un Atestado por recibo de factura de entrada a la Fiscalía.

El sistema permite actualizar el historial del atestado al que se haga referencia en una factura recibida ya sea por atestados devueltos de la PNR o del Tribunal, siempre aclarando el origen del atestado correspondiente, es decir, si se recibe un documento Factura, se verifica a qué atestado(s) hace referencia y se le(s) actualiza su historial con la entrada correpondiente ya sea del Tribunal o de la PNR.

14. Crear Documentación.

El sistema permite los siguientes documentos:

- M-1, M2, M-3, M-4, M-5, M-6, M-7. M-9, P-3, P-14, P-16, P-18, P-18.1, P-20

15. Mostrar Documentación.

El sistema permite mostrar la documentación para una denuncia dada.

16. Registrar cantidad de 8.2 aplicados.

El sistema permite registrar la cantidad de medidas administrativas aplicadas por el artículo 8.2 por la PNR en el periodo de tiempo suministrado en que se realice una revisión de su cumplimiento.

17. Registrar cantidad de 8.3 aplicados.

El sistema permite registrar la cantidad de medidas administrativas aplicadas por el artículo 8.3 por la PNR en el periodo de tiempo suministrado en que se realice una revisión de su cumplimiento.

18. Mostrar datos de acusado detenido.

El sistema permite mostrar los datos de los acusados detenidos.

19. Mostrar Reportes.

El sistema permite mostrar el resultado de la búsqueda para distintos reportes.

20. Mostrar Indicación para un atestado dado.

El sistema permite mostrar la indicación realizada a un atestado por un Fiscal Superior.

21. Actualizar automáticamente libro de control de denuncias.

El sistema permite actualizar automáticamente en el libro de control de denuncias los datos requeridos de cualquier denuncia sobre la que se realice una salida de la Fiscalía, una entrada a esta posterior a la inicial o cualquier otro tipo de modificación.

22. Adicionar unidad de la PNR.

El sistema permite adicionar una unidad de la PNR.

23. Modificar datos de unidades de la PNR.

El sistema permite dada una unidad de la PNR.

24. Desactivar unidades de la PNR.

El sistema permite poner en desuso una unidad de la PNR.

25. Asignar unidad de la PNR.

El sistema permite asignar una o varias unidades de la PNR a los Fiscales Municipales.

26. Eliminar asignación unidad de la PNR.

El sistema permite eliminar la unidad de la PNR asignada a un Fiscal Municipal.

27. Reasignar Atestado.

El sistema permite que el Fiscal Jefe Municipal reasigne un atestado que estaba trabajando un Fiscal para otro Fiscal.

28. Realizar indicación.

El sistema permite que un Fiscal de una instancia superior pueda realizar cualquier indicación que considere necesaria sobre cualquier atestado por el consultado.

29. Imprimir el Libro de Control.

El sistema permite imprimir el Libro de Control.

30. Imprimir Documentación.

El sistema permite imprimir los documentos:

- M-1, M2, M-3, M-4, M-5, M-6, M-7. M-9, P-3, P-14, P-16, P-18, P-18.1, P-20

31. Crear Factura.

El sistema permite crear una factura.

32. Buscar Factura.

El sistema permite buscar la factura.

33. Buscar Atestado.

El sistema permite buscar el expediente de Sumario.

34. Buscar el Libro de Control.

El sistema permite buscar el libro de control de atestados entregados a la Fiscalía.

35. Buscar el Historial del Atestado.

El sistema permite buscar un expediente de Sumario que exista previamente en el sistema, puede ser uno en particular, o varios que cumplan con los mismos parámetros de búsqueda.

36. Mostrar Factura.

El sistema permite mostrar las facturas listadas en el resultado de la búsqueda.

37. Imprimir la Factura.

El sistema permite imprimir la Factura.

2.2.1.2. Requisitos no Funcionales

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. Restricciones de Diseño

- El sistema se desarrollará en plataforma Linux, Apache y Postgree.
- El lenguaje de programación es PHP.
- El framework de desarrollo es Symfony.
- El generador de reportes que se utilizará en el proyecto es el HTML_to_PDF.
- La herramienta IDE de desarrollo utilizada será Eclipse PDT.
- La herramienta case utilizada es el Visual Paradigm.
- Se utilizara el patrón de arquitectura MVC.
- La herramienta gestor de base de datos es el PostgreSQL.
- Herramienta para la réplica con PostgreSQL es el PgAdminIII.

6. 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.

7. 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.

8. Confiabilidad

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

9. Funcionalidad

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

10. Implantación

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

11. 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**
  - Servidor con funciones integradas: DNS, DHCP, Firewall.
  - Servidor de correo independiente.
  - Servidor independiente LDAP.
  - Servidor Postgrees y FTP.

12. Software

- **Cliente**
  - Debe tener el sistema operativo Debian o Windows.
  - Debe poseer un navegador web.

- **Servidor**
  - Debe poseer servicio de directorio activo.

2.2.1.3. Procesos objeto de automatización

A continuación se describen los procesos del negocio que son objeto de automatización, con el objetivo de brindar posteriormente la solución propuesta.

- **Creación de denuncia:**

La creación de denuncia es uno de los primeros pasos que se realizan en los distintos sectores de las PNR cuando un individuo es detenido. Estas denuncias se crean en formato duro y cuando llegan a la fiscalía, se procede al llenado de cada una de las actuaciones según el tipo de denuncia que se está manejando, quedando claras las causas y los agravantes del por qué el individuo ha sido detenido. Este proceso exige un tiempo considerable a las personas designadas, automatizarlo es prioridad en todo sistema que desee mejorar la eficiencia.

- **Manejo de las facturas de las denuncias:**

Es un proceso donde se hace necesario el control de los movimientos de uno o varios atestados desde una PNR a la Fiscalía, desde la Fiscalía al Tribunal o cualquier combinación entre los anteriores. En dependencia del origen y el destino se le debe otorgar un estado a la tramitación que se lleva a cabo quedando en claro, los nombres de los implicados en el proceso y la fecha del mismo para en un momento posterior se pueda verificar la veracidad y cumplimiento de la acción. Asimismo se deberá tener conocimiento sobre el estado de los atestados implicados de forma tal que se sepa si ya estos han sido tramitados con anterioridad o si son de casos nuevos, por lo que se debe verificar en cada caso, el estado de los mismos. Es algo complicado tener control sobre todo el manejo y la evolución que se le debe realizar a la confección de las facturas de los atestados, por lo que se hace necesario automatizar este proceso que es de vital importancia para el cumplimiento de un buen desempeño de las personas implicadas y evitar la sobre carga de trabajo de las mismas.

2.2.2. Estándares de código

2.2.2.1. Estándares de desarrollo para PHP

- **Estándar 1**: La indexación debe ser a cuatro espacios sin caracteres de tabulación. Esto es debido a que ciertos IDE's de desarrollo introducen caracteres de tabulación cuando indexan un texto automáticamente. Se recomienda el uso de herramientas o editores generales como EMACS u otros.
- **Estándar 2**: 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.
- **Estándar 3**: 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 último paréntesis, el signo de paréntesis cerrado y el signo de punto y coma (**;**).
- **Estándar 4**: 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 (#).
- **Estándar 5**: Cuando se incluya un archivo de dependencia incondicionalmente utilice **require_once** y cuando sea condicionalmente, utilice **include_once**.
- **Estándar 6**: 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.
- **Estándar 7**: 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.
- **Estándar 8**: 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.

2.2.2.2. 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.

2.2.2.3. La Longitud de la Línea máxima

La longitud de la línea designada es de 80 caracteres. Sin embargo, las líneas más largas son aceptables hasta un máximo de 120 caracteres.

2.2.2.4. 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.

2.2.2.5. 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.

2.2.2.6. 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> */

2.2.2.7. 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 Caja{

    var $alto;

    var $ancho;

    var $largo;

    var $contenido;

    var $color;

}

?>
```

2.2.2.8. Las clases objeto disponen de *getters* para los registros de las Columnas:

```
$articulo = new Article();

...
```

```
$titulo = $articulo->getTitle();
```

ArticlePeer y CommentPeer son clases de tipo “peer”; es decir, clases que tienen métodos estáticos para trabajar con las tablas de la base de datos. Proporcionan los medios necesarios para obtener los registros de las tablas.

2.2.2.9. 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  $var=foo($bar, $baz, $guux);  ?>
```

2.2.2.10. Las estructuras de mando

Las estructuras de mando incluyen: **“si, para, mientras, cambie, etc.”**. Este es un ejemplo:

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

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.

2.2.3. Propuesta de solución

2.2.3.1. Modelo de datos

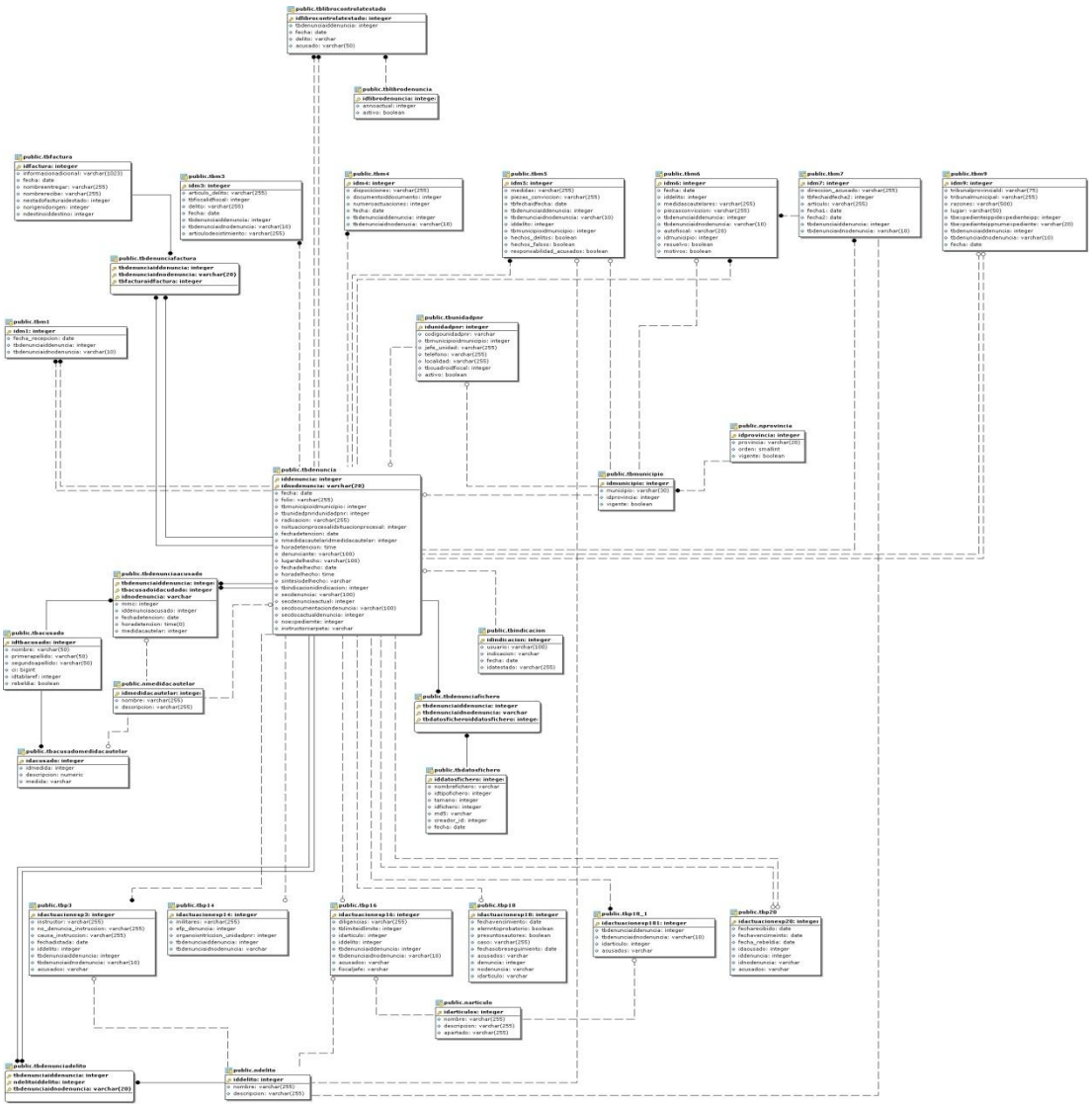

Figura 1: Modelo de datos del sistema

2.2.3.2. Diagramas de Clases de Diseño

Los Diagramas de Clases de Diseño, se confeccionan durante el flujo de trabajo de Análisis y Diseño. Estos muestran la especificación para las clases software de una aplicación. Incluye la siguiente información:

- Clases, asociaciones y atributos.
- Interfaces con sus operaciones y constantes.
- Métodos.
- Navegabilidad.
- Dependencias.

A diferencia del Modelo Conceptual, un diagrama de clases de Diseño muestra definiciones de entidades software más que conceptos del mundo real. (Poz, 2010)

A continuación se muestran como ejemplo los Diagramas de Clases de Diseño de:

- **Crear Denuncia**.

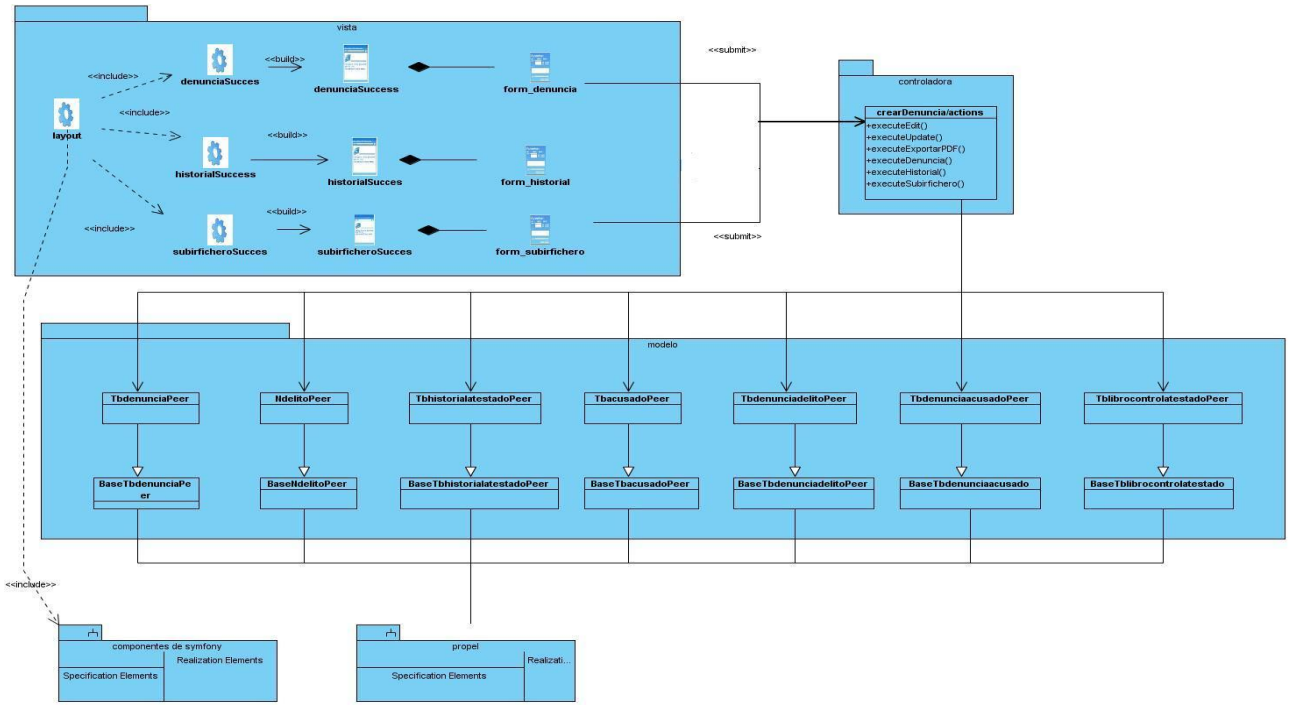

Figura 2: Diagrama de Clases del Diseño del CU: Crear Denuncia.

➢ **Manejar Facturas**

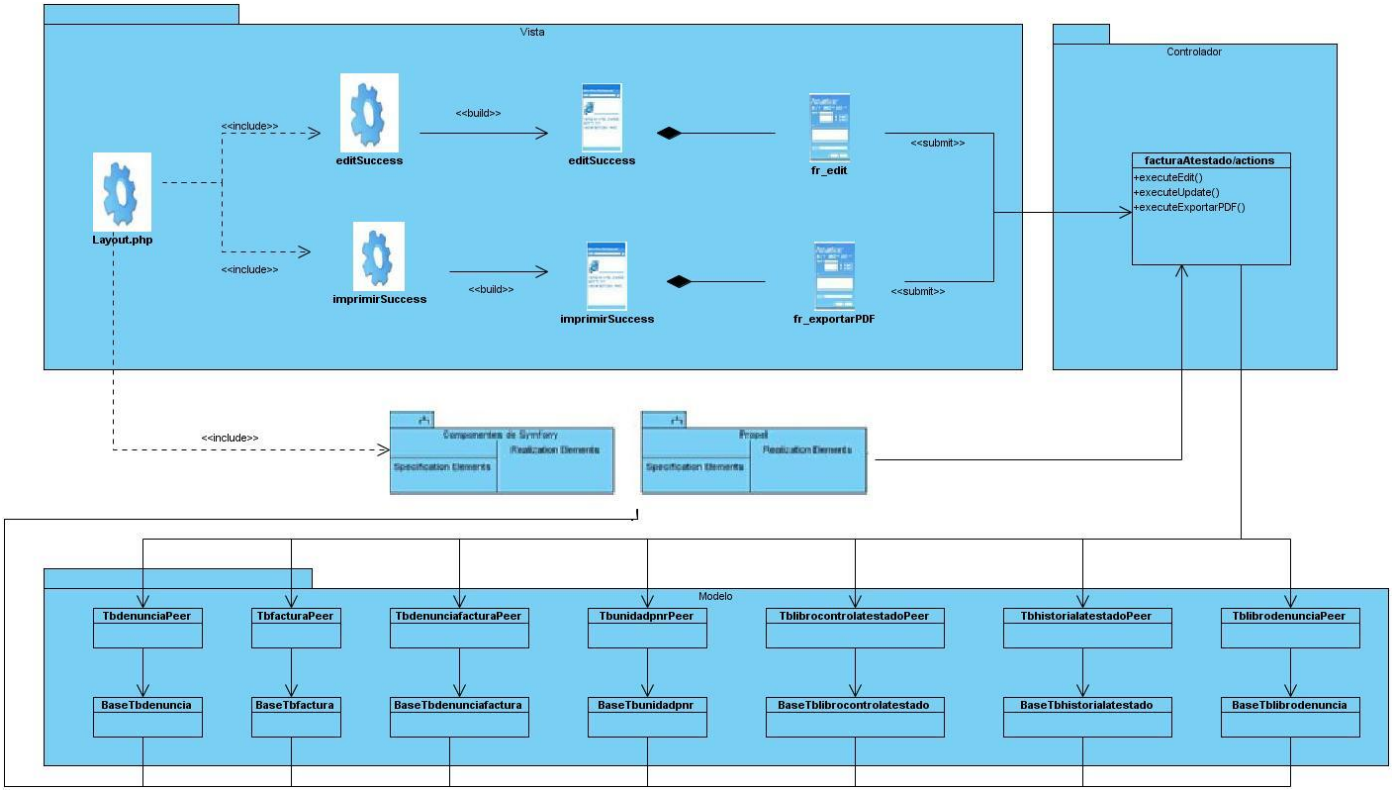

Figura 3: Diagrama de Clases del Diseño del CU: Crear Denuncia.

2.2.3.3. Propuesta del Sistema

Para conectarse a la aplicación se requiere que el usuario se autentique y a partir de ese momento se le mostrarán la página principal de la aplicación con los íconos de cada uno de los módulos al que éste tenga acceso según los niveles accesos otorgados con anterioridad.

Una de las acciones más importantes a realizar, es la creación de la denuncia, donde se registran todos los datos de interés de una denuncia. Este proceso se realiza en sus inicios en la PNR en la cual la denuncia es presentada y posteriormente, de acuerdo con los plazos de tiempos establecidos, esta

denuncia es enviada a la fiscalía en formato y es ahí donde se crea en el sistema, quedando reflejados todos los datos que con anterioridad han sido descritos y el número de la denuncia.

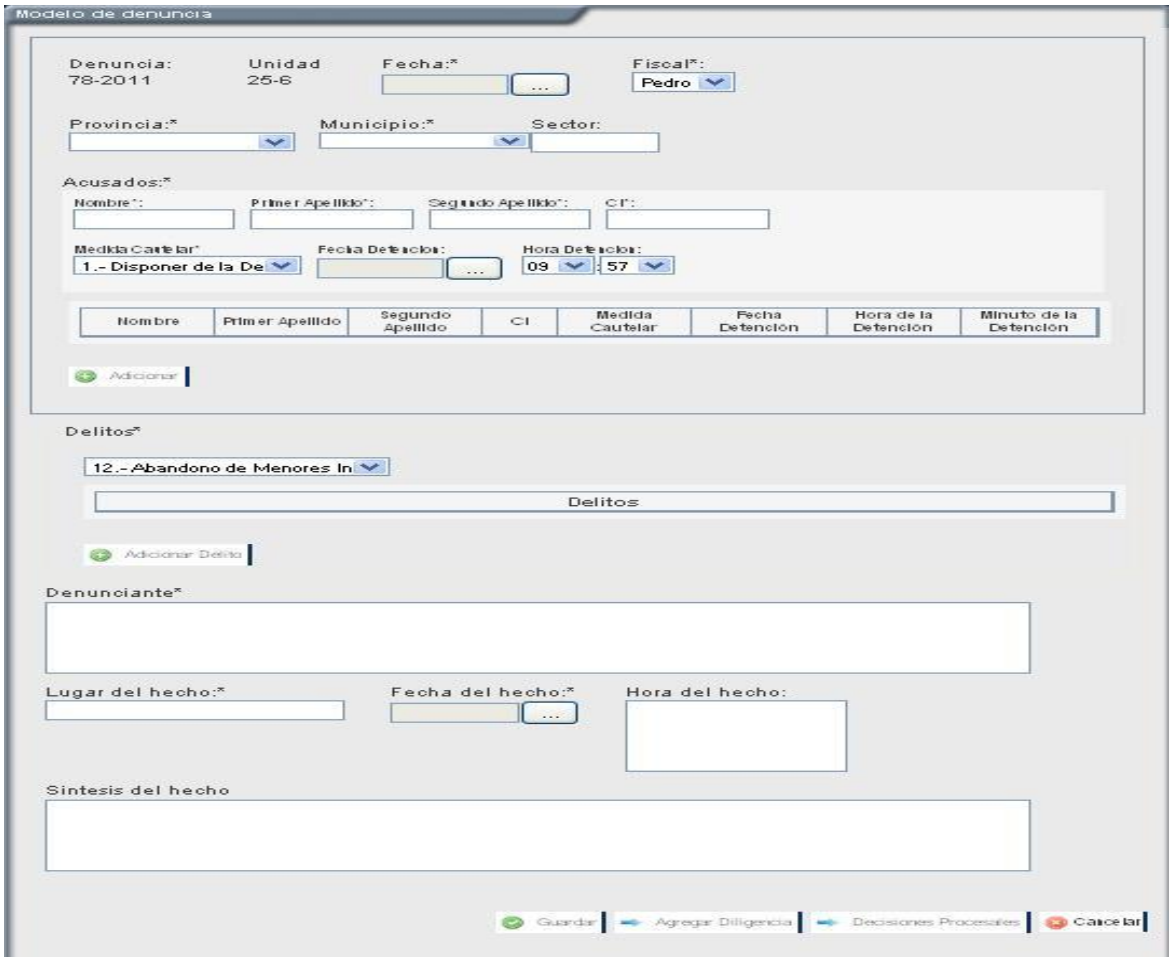

Figura 4: Prototipo de Interfaz de Crear Denuncia.

Para el movimiento de una denuncia de una PNR a la fiscalía, de la fiscalía al tribunal o cualquier combinación entre estos; se deberá crear una factura, la que deberá contener todos los datos que posibiliten identificarla de las demás, así como el estado de tramitación que dependerá del origen y el destino de las denuncias implicadas en el proceso. Además, se debe especificar los nombres de los involucrados y la fecha en que se está realizando la acción.

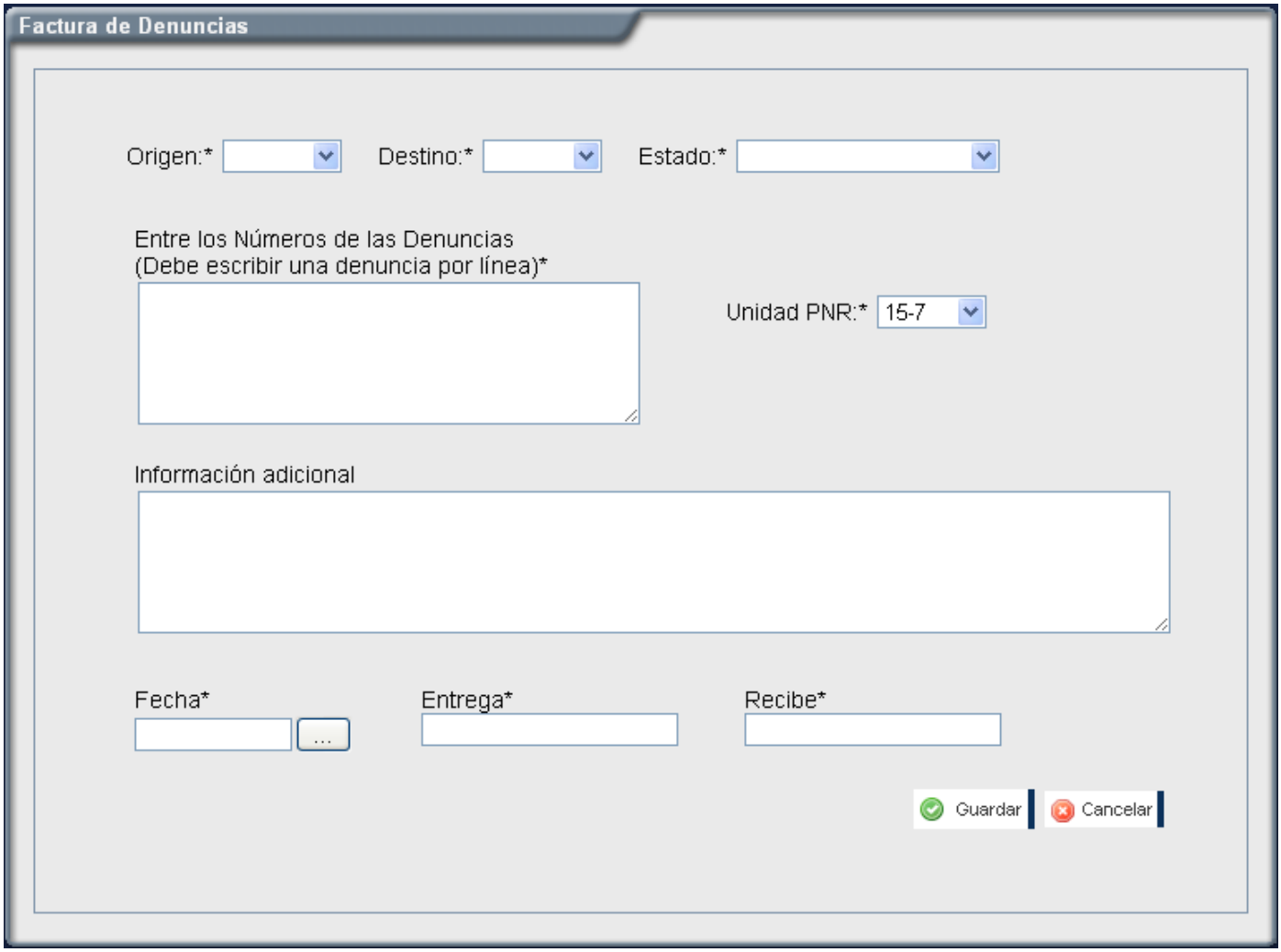

Figura 5: Prototipo de Interfaz de Manejar Factura.

Es de vital importancia el conocimiento del estado en el que se encuentran las denuncias que están involucradas en la factura, por lo que se le informará a la persona que la está tramitando, si alguna de las denuncias es de nuevo ingreso a la fiscalía y debe ser creada en el sistema.

2.2.3.4. Arquitectura

Existen varios lugares donde la arquitectura está presente, principalmente en el desarrollo de aplicaciones informáticas, sin lugar a duda, lo que la convierte en una herramienta de puntería para la organización del trabajo. Partiendo de esto, se puede decir, que hay diferentes caminos por lo cual guiar un trabajo o un proyecto. Para el desarrollo de aplicaciones, hay creados diferentes estilos arquitectónicos que solucionan un problema, en dependencia de las características del mismo, como ejemplo de esto se pueden señalar: el modelo-vista-controlador, arquitectura en capas entre otros. Por tanto, es de vital importancia una selección óptima del mismo.

En la implementación del módulo Sumario, se utilizó el Patrón de Arquitectura Modelo-Vista-Controlador; donde se manifiesta con la utilización del Symfony Framework, la implementación de todo el sistema propuesto a desarrollar. Con esta selección se persigue el objetivo de utilizar la separación de las responsabilidades de cada una de las capas que lo conforman y así lograr facilidades de desarrollo.

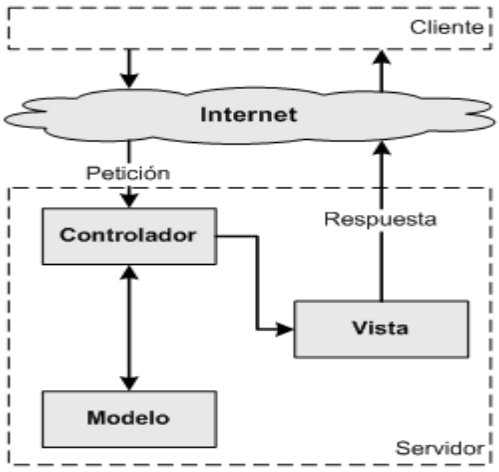

Figura 6: Arquitectura Modelo Vista Controlador.

▪ 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 o en la vista.

(Potencier, 2007)

2.2.3.5. Patrones GRASP implementados

Creador

En las actions se crean los objetos de las clases que representan las entidades, evidenciando de este modo que la clase actions es **creador** de dichas entidades.

Experto

Este es uno de los más utilizados, puesto que Propel es la librería externa que utiliza Symfony para realizar su capa de abstracción en el modelo, encapsula toda la lógica de los datos y son generadas las clases con todas las funcionalidades comunes de las entidades.

Alta Cohesión

Symfony permite asignar responsabilidades con una alta cohesión, por ejemplo la clase actions tiene la responsabilidad de definir las acciones para las plantillas y colabora con otras para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades, es decir, está formada por diferentes funcionalidades que se encuentran estrechamente relacionadas proporcionando que el software sea flexible frente a grandes cambios.

Controlador

Todas las peticiones Web son manejadas por un solo controlador frontal (sfActions), que es el punto de entrada único de toda la aplicación en un entorno determinado. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario.

Bajo Acoplamiento

La clase actions hereda solamente de sfActions para lograr un bajo acoplamiento de clases. (Potencier, 2007)

2.2.3.6. Tipos de patrones GOF que implementa Symfony

En la categoría Creacionales:

Singleton (instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. En el controlador frontal hay una llamada a sfContext::getInstance(). En una acción, el método getContext(), un objeto muy útil que guarda una referencia a todos los objetos del núcleo de Symfony.

Abstract Factory (Fábrica Abstracta): Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando. Cuando el framework necesita por ejemplo crear un nuevo objeto para una petición, busca en la definición de la factoría el nombre de la clase que se debe utilizar para esta tarea.

En la categoría Estructurales:

Decorator (Envoltorio): Añade funcionalidad a una clase, dinámicamente. El archivo layont.php, que también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el layout, o si se mira desde el otro punto de vista, el layout decora la plantilla. Este comportamiento es una implementación del patrón de diseño llamado

Composite (Objeto compuesto): Permite tratar objetos compuestos como si se tratase de un objeto simple. Sirve para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol .Esto simplifica el tratamiento de los objetos creados, ya que al poseer todos ellos una interfaz común, se tratan todos de la misma manera. (Potencier, 2007)

2.2.3.7. Descripción de clases

2.2.3.7.1. Clases Controladoras

La clase controladora es responsable de ejecutar una determinada lógica en función de las acciones que se producen en una aplicación. Se encargan de la captura de eventos, del seteo de entidades y permiten la comunicación con las clases del negocio. (Potencier, 2007)

- Crear Denuncia.

Nombre: actions	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	**Descripción:**
executeDenuncia	Crear un objeto de la tabla Tbdenuncia que responde a los datos que serán introducidos en la vista.
executeUpdateDenuncia	Guarda en la tabla Tbdenuncia los datos introducidos en la vista.

- Manejar Factura

Nombre: actions	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	**Descripción:**
executeCreate	Crea los objetos que serán utilizados en la vista.
executeEdit	Permite crear uno o varios objetos con todos los datos que estos poseen en las tablas de la base de datos.
executeUpdate	Permite guardar los datos recogidos de la vista en las tablas de la base de datos.

2.2.3.7.2. Clases del modelo

Las clases entidad pertenecientes al modelo, manejan la información que poseen una larga vida y que a menudo son conceptos. También se denominan clase dominio, ya que suelen tratar con abstracciones de entidades del mundo real. Se encargan de modelar la información del sistema y el comportamiento asociado a una información. (Potencier, 2007)

- Crear Denuncia.

Tipo de clase: Entidad
Nombre de la clase:
Tbdenuncia
Tbacusado
Tbdenunciaacusado
Ndelito
Tbhistorialdenuncia
Tblibrocontrolatestado
Tblibrodenuncia
Nmedidacautelar
Tbindicacion
Tbfiscalespnr
Tbdenunciadelito

- Manejar Factura

Tipo de clase: Entidad
Nombre de la clase:
Tbdenunciafactura
Tbdenuncia
Tblibrocontrolatestado
Tbfactura

2.2.3.7.3. Clases de la vista

Permiten la interacción del usuario con el sistema mediante un formulario donde se le muestran los datos que deben ser introducidos por el mismo para realizar las diferentes acciones que serán manejadas en el controlador. (Potencier, 2007)

- Crear Denuncia.

Nombre: template	
Para cada responsabilidad:	
Nombre:	**Descripción:**
denunciaSuccess	Muestra un formulario con todos los datos que el usuario debe llenar.
historialSuccess	Muestra un formulario con todos los datos que el usuario desea conocer.

- Manejar Factura

Nombre: template	
Para cada responsabilidad:	
Nombre:	**Descripción:**
editSuccess	Muestra un listado de todos los campos de la tabla tbFactura.
imprimirSuccess	Muestra un listado de todos los datos que el usuario desea imprimir.

2.2.4. Diagramas de Componentes

Los Diagramas de Componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación mostrando las relaciones entre sus elementos. A continuación se muestra el diagrama de componentes de los casos de uso Crear Denuncia y Manejar Factura.

- Crear Denuncia.

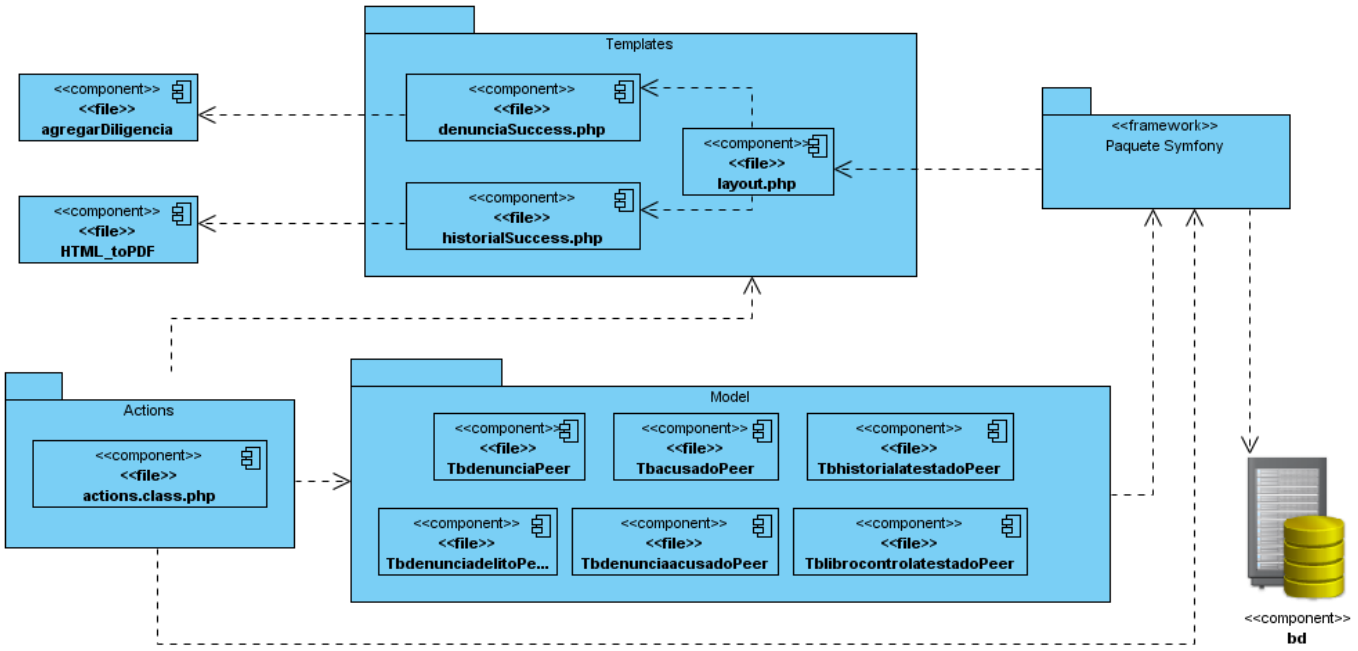

Figura 7: Diagrama de Componentes CU: Crear Denuncia.

- Manejar Factura.

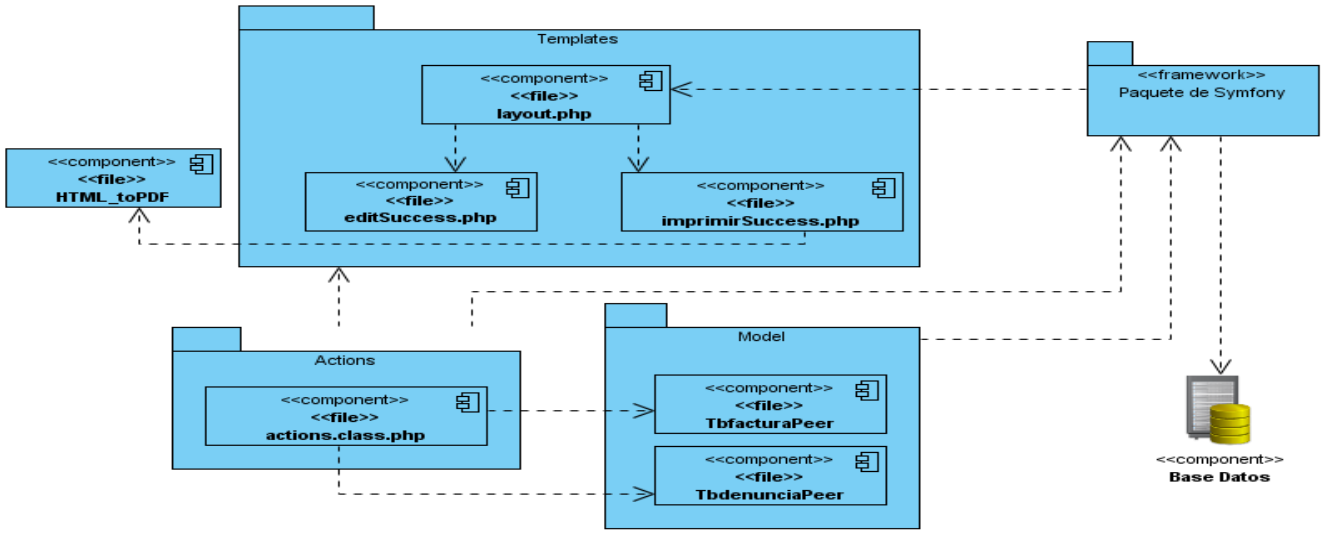

Figura 8: Diagrama de Componentes CU: Manejar Factura.

2.2.5. Diagrama de Despliegue

El Diagrama de Despliegue se utiliza para modelar la arquitectura en tiempo de ejecución de un sistema. Detalla las capacidades de red, las especificaciones del servidor, los requisitos de hardware y otra información relacionada al despliegue del sistema propuesto. Se representa mediante un grafo de nodos unidos por conexiones de comunicación, los nodos son elementos físicos de ejecución que representa un recurso y sirven para modelar la topología del hardware sobre el que se ejecuta el sistema. Un nodo representa, generalmente, un procesador o un dispositivo sobre el que se pueden desplegar los componentes y debe tener un nombre asignado que lo distinga del resto de nodos. (Jacobson, y otros, 2000)

A continuación se muestra el modelo de despliegue del proyecto SIGEF.

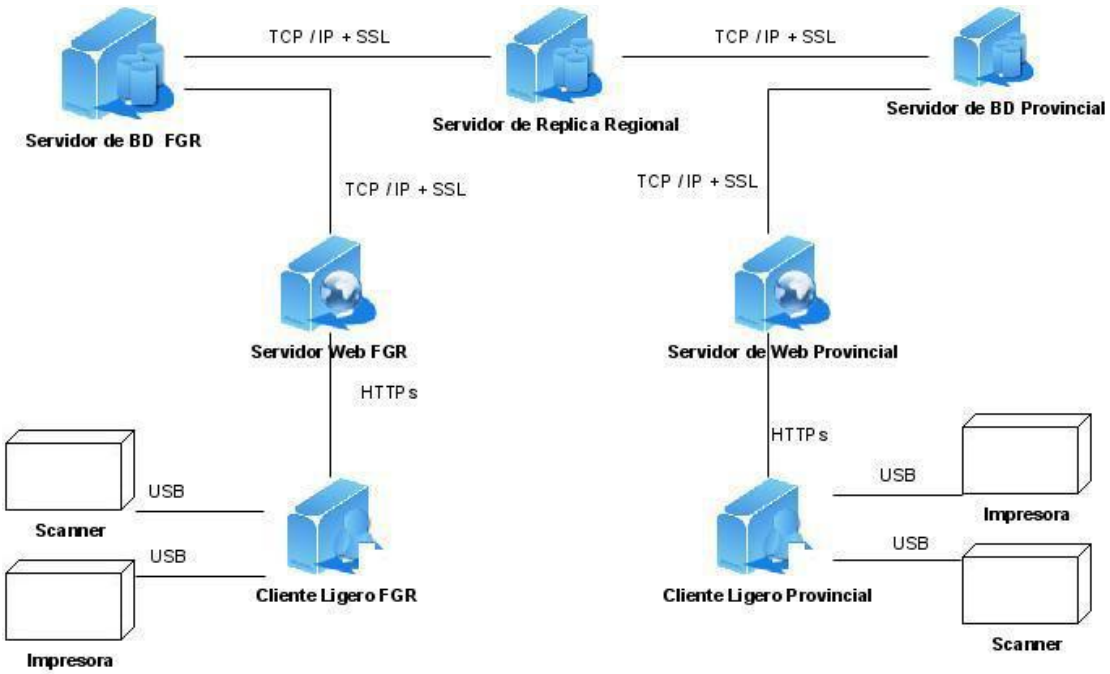

Figura 9: Diagrama de Despliegue del SIGEF.

2.3. Conclusiones parciales

Con el desarrollo de este capítulo se conocieron los procesos objetos de automatización, lo que proporcionó una temprana idea de la complejidad de la solución. Se explicaron los componentes que fueron reutilizados para el desarrollo de la aplicación teniendo en cuenta la arquitectura del sistema. Todo lo anterior posibilitó la implementación de las interfaces de cada uno de los componentes. La descripción de las clases permitió tener una visión para la puesta en práctica de patrones como el Singleton, Experto, Controlador y Decorador. De forma general se logró implementar el módulo Sumario, perteneciente al subsistema Procesos Penales del proyecto SIGEF.

Capítulo 3

3.1. Introducción

En el desarrollo del software, las posibilidades de error son innumerables. Pueden darse por una mala especificación de los requisitos funcionales, uso indebido de las estructuras de datos, errores al enlazar módulos, entre otras. Para resolver este problema, se incluyó a nivel metódico un nuevo proceso en la confección de los sistemas informáticos: el proceso de prueba. Hoy en día, se calcula que la fase o proceso de pruebas, representa más de la mitad del coste de un programa ya que requiere, un tiempo similar al de la programación lo que obviamente acarrea un alto costo económico, cuando estos no involucran vidas humanas, puesto que en este último caso el costo suele superar el 80% siendo esta etapa más cara que el propio desarrollo y diseño de los distintos programas que conforman el sistema. El desarrollo del software ha de ir acompañado de alguna actividad que garantice la calidad; la prueba es un elemento crítico para la garantía de calidad del software. La prueba y validación de los resultados no es un proceso que se realiza una vez desarrollado el software, sino que debe efectuarse en cada una de las etapas de desarrollo. Es fundamental medir la cobertura de las pruebas, es decir, la determinación de cuándo se han realizado las suficientes pruebas. Si se siguen encontrando errores cada vez que se procesa el programa, las pruebas deben continuar. En este capítulo se valida la solución propuesta en el capítulo anterior, de manera que se puede comprobar la eficiencia de las clases u operaciones utilizadas, por lo que se presenta la descripción de clases de prueba que verifican la validez de las funcionalidades del módulo Sumario, así como verificar el rendimiento interno de las mismas, además de evaluar mediante métricas el diseño propuesto por los analistas.

3.2. Desarrollo

3.2.1. Pruebas de software

Las pruebas de software, son el conjunto de técnicas que permiten determinar la calidad de un producto. Se integran a las diferentes fases del ciclo de desarrollo dentro de la ingeniería de software, manifestándose mediante la ejecución de un programa o mediante técnicas experimentales con el

propósito de descubrir errores. Un buen caso de prueba es aquel que tiene alta probabilidad de mostrar un error no descubierto hasta entonces. Una prueba tiene éxito si descubre un error no detectado hasta entonces. La prueba no puede asegurar la ausencia de defectos; sólo puede demostrar que existen defectos en el software. La fase de pruebas es la que añade al producto final el valor para afirmar que ya se ha alcanzado la calidad requerida. Gran porciento de los programas que se desarrollan tienen errores, y es en la fase de pruebas donde se descubren, ese es el valor que añade esta etapa, el objetivo específico de la fase de pruebas es encontrar cuantos más errores mejor. Es por ello que probar es una de las fases más importantes para que un producto salga con la calidad máxima y sin errores. (Collado, 2003)

3.2.1.1. Objetivos

La prueba del software es un elemento crítico para la garantía de la calidad del software. El objetivo de la etapa de pruebas es garantizar la calidad del producto desarrollado. Además, esta etapa implica:

- Verificar la interacción de componentes.
- Verificar la integración adecuada de los componentes.
- Verificar que todos los requisitos se han implementado correctamente.
- Identificar y asegurar que los defectos encontrados se han corregido antes de entregar el software al cliente.
- Diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo.

La prueba no es una actividad sencilla, no es una etapa del proyecto en la cual se asegura la calidad, sino que la prueba debe ocurrir durante todo el ciclo de vida: probar la funcionalidad de los primeros prototipos, probar la estabilidad, cobertura y rendimiento de la arquitectura, probar el producto final,

entre otras. Lo que conduce al principal beneficio de la prueba: proporcionar feedback36 mientras hay todavía tiempo y recursos para hacer algo. (Collado, 2003)

3.2.1.2. Alcance

Se lleva a cabo la prueba y se evalúan los resultados obtenidos frente a los resultados esperados. Si se descubren datos erróneos implica que hay un error y hay que corregirlo y empieza el proceso de depuración de errores. Se basa en las estructuras de control del diseño procedimental para generar los casos de prueba que:

- Garanticen que se recorren por lo menos una vez todos los caminos independientes de cada módulo.
- Se ejecutan todas las decisiones lógicas en su parte verdadera y en su parte falsa.
- Se recorren todos los bucles.
- Se utilizan las estructuras de datos internas para garantizar su validez.(Collado, 2003)

3.2.2. Descripción de los test de unidad

Las pruebas de unidad son la manera de comprobar el funcionamiento correcto de determinado módulo de código y ayuda a independizar el mismo, significa esto que se pueden probar los módulos, independientemente uno de otros. Antes de iniciar cualquier otra prueba es preciso probar el flujo de datos de la interfaz del módulo. Si los datos no fluyen correctamente, todas las demás pruebas no tienen sentido. Los “test de unidades” son orientados casi siempre a las pruebas de “caja blanca” aunque para realizar uno de estos test es necesario probar el flujo de datos desde la interfaz del componente. Si los datos no se comportan correctamente al ser introducidos en la aplicación, todas las demás pruebas no cumplen función hacerlas. (Fernando, 2006)

3.2.3. Pruebas de Caja Blanca o Funcionales

También conocidas como Pruebas de Comportamiento o Pruebas Inducidas por los Datos. Estas pruebas se basan en la especificación del programa o componente a ser probado para elaborar los casos de prueba. El componente se ve como una “Caja Negra” cuyo comportamiento sólo puede ser determinado estudiando sus entradas y las salidas obtenidas a partir de ellas. No obstante, como el estudio de todas las posibles entradas y salidas de un programa sería impracticable, se selecciona un conjunto de ellas sobre las que se realizan las pruebas. Para seleccionar el conjunto de entradas y salidas sobre las que trabajar, hay que tener en cuenta que en todo programa existe un conjunto de entradas que causan un comportamiento erróneo en el sistema, y como consecuencia producen una serie de salidas que revelan la presencia de defectos. Entonces, dado que la prueba exhaustiva es imposible, el objetivo final es pues, encontrar una serie de datos de entrada cuya probabilidad de pertenecer al conjunto de entradas que causan dicho comportamiento erróneo sea lo más alto posible.

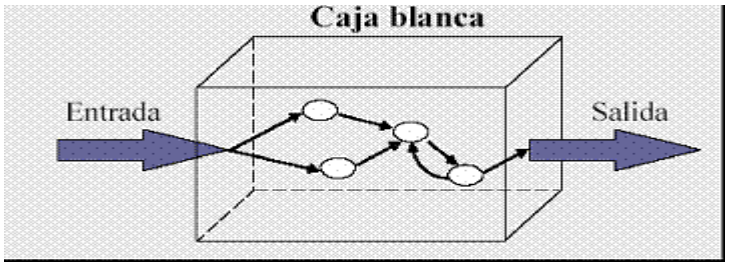

Figura 10: Prueba de Caja Blanca

Algunas técnicas utilizadas en la prueba de caja negra son:

- **Partición de Equivalencia:** Técnica que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a una definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

  Ejemplo: Si una condición de entrada especifica un rango de valores, o sea, si un contador puede ir de 1 a 999, la clase válida sería “1 <= contador <= 999. Mientras que las clases no válidas serían “contador < 1” y “contador > 999”.

- **Análisis de Valores Límite:** La experiencia muestra que los casos de prueba que exploran las condiciones límite producen mejor resultado que aquellos que no lo hacen. Las condiciones límite son aquellas que se hallan en los márgenes de la clase de equivalencia, tanto de entrada como de salida. Por ello, se ha desarrollado el análisis de valores límite como técnica de prueba. Esta técnica nos lleva a elegir los casos de prueba que ejerciten los valores límite. (Barrios, 2007)

3.2.4. Aplicación de Pruebas de Caja Negra

Para la aplicación de este tipo de prueba se usará el requisito Manejar Factura. El objetivo de este requisito es la creación de una denuncia por parte de un Gestor Común que en este caso puede ser el Fiscal Municipal o la Secretaria, pudiendo adicionar todas las actuaciones que desee en correspondencia con la factura en formato duro que se está digitalizando. Se realizarán una serie de acciones automáticas como son el llenado y actualización del Historial del Denuncia y del Libro de Control.

3.2.4.1. Aplicando la prueba de Caja Negra al CU: Manejar Factura

Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**
<SC 1 Manejar Factura >	EC 1.1: El Gestor común selecciona la opción de Manejar Factura.	1.1. El sistema muestra la interfaz “Manejar factura” con los siguientes datos a llenar: • Origen • Destino • Estado • Denuncia • Información adicional

		• Fecha • Entrega • Recibe
	EC 1.2: El Gestor común llena los campos obligatorios y presiona el botón Guardar.	1.2. El sistema verifica que todos los datos tengan el formato correcto y todos los campos obligatorios estén llenos. 1.2.1. El sistema guarda la factura en la base de datos. 1.2.2. El sistema verifica si las denuncias de la factura creada están ya en el sistema o deben ser creados en el mismo, mostrando un mensaje. 1.2.3. El sistema dependiendo de la verificación del paso anterior muestra un reporte con los expedientes no creados en el sistema para su

		creación. 1.2.4. Se actualizan el historial y el libro de control de los atestados.
	1. **EC 1.3: El Gestor común marca el botón Cancelar.**	***1.3.*** **El sistema cierra la interfaz y muestra la Página Principal.**

3.2.4.2. Descripción de las variables.

No.	Nombre del campo	Clasificación	Puede ser nulo	Descripción
1	Origen	Lista desplegable	No	La introducción de este dato no admite error ninguno.
	Destino	Lista desplegable	No	La introducción de este dato no admite error ninguno.
	Estado	Lista desplegable	No	La introducción de este dato no admite error ninguno.
	Número de las Denuncias	2. ***Campo de texto***	No	No permite la entrada de letras y de caracteres extraños. Solo admite números.

	Unidad de la PNR	Lista desplegable	No	La introducción de este dato no admite error ninguno.
	Fecha	Lista desplegable	No	La introducción de este dato no admite error ninguno.
	Entrega	3. ***Campo de texto***	No	Debe estar completo y los datos introducidos el formato correcto. Solo admite más de dos letras.
	Recibe	4. ***Campo de texto***	No	Debe estar completo y los datos introducidos el formato correcto. Solo admite más de dos letras.
	Información adicional	5. ***Campo de texto***	Si	Admite números, letras y caracteres extraños como puntos, punto y coma y coma.

3.2.5. Métricas de Diseño

Se conoce que las medidas y las métricas son componentes clave de cualquier disciplina de la ingeniería; la ingeniería de software orientada a objetos no es una excepción. Lamentablemente, la utilización de métricas para sistemas orientados a objetos ha progresado con mucha más lentitud que la utilización de los demás métodos orientados a objetos. Sin embargo, a medida que los sistemas orientados a objetos van siendo más habituales, resulta fundamental que los ingenieros del software

dispongan de mecanismos cuantitativos para estimar la calidad de los diseños y la efectividad de los programas. (Doria, 2001)

Atributos de calidad que se abarcan:

1. **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.
2. **Complejidad del diseño**. Consiste en la complejidad que posee una estructura de diseño de clases.
3. **Complejidad de implementación.** Consiste en el grado de dificultad que tiene que implementar un diseño de clases determinado.
4. **Reutilización.** Consiste en el grado de reutilización de presente en una clase o estructura de clase, dentro de un diseño de software.
5. **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.
6. **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.
7. **Cantidad de pruebas.** Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (Componente, módulo, clase, conjunto de clases, etc.) diseñado.

Las métricas concebidas como instrumento para evaluar la calidad de los Módulos de Sumario y su relación con los atributos de calidad definidos en este trabajo son las siguientes:

De la serie Chidamber y Kemner:

- Respuesta para una Clase (RC). Ver instrumentos y tabla de resultados en Anexo 1

Se utilizó de la serie definida por Lorenz y Kidd:

- Tamaño Operacional de Clases (TOC). Ver instrumentos y tabla de resultados en Anexo 2

(Alarcos, Grupos)

Relaciones entre clases (RC): Está dado por el número de relaciones de uso de una clase con otras.

Atributo que afecta	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del Acoplamiento de la clase.
Complejidad del mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Figura 11: Relaciones entre clases (RC).

Tamaño operacional de clase (TOC): Está dado por el número de métodos asignados una clase.

Atributo que afecta	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución en el grado de reutilización de la clase.

Figura 12: Tamaño operacional de clase (TOC).

3.2.5.1. Resultados del instrumento de evaluación de la métrica Tamaño operacional de clase **(TOC).**

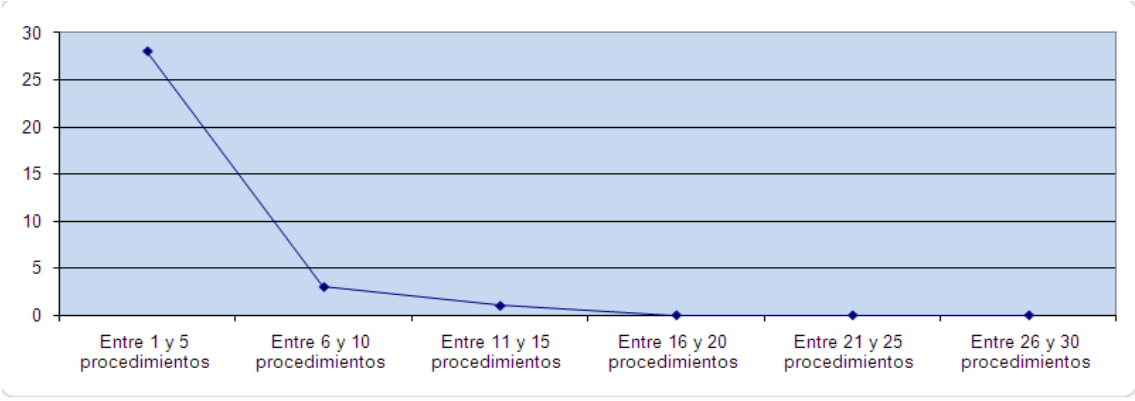

Figura 13: Representación de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

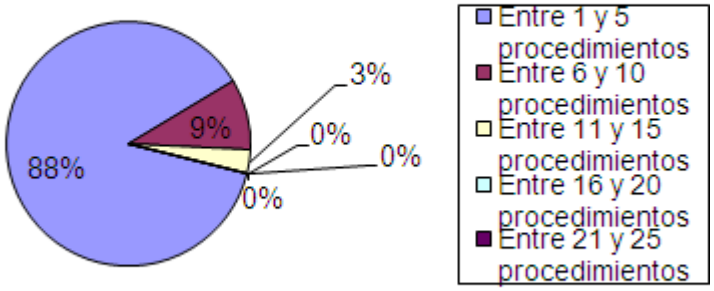

Figura 14: Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

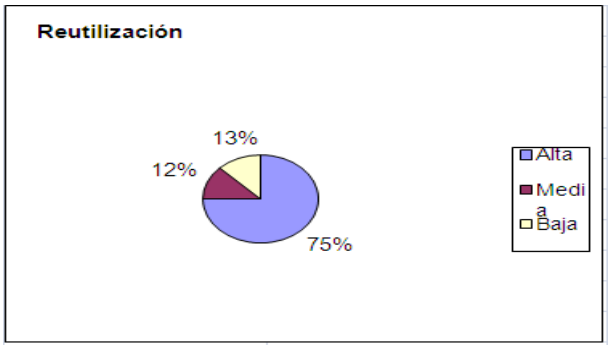

Figura 15: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en los atributos Responsabilidad, Complejidad y Cohesión.

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 módulo Sumario tiene una buena calidad teniendo en cuenta que el 88 % de las clases incluidas poseen menos cantidad de operaciones que el promedio de procedimientos para una clase, lo cual influye positivamente en el hecho de que predomine una responsabilidad baja de las clases en un 75%. Además, el 75% de las clases poseen evaluaciones positivas en los atributos de calidad complejidad de implementación y reutilización lo cual garantiza una solidez en el diseño del sistema puesto que habrá gran reutilización.

3.3. Conclusiones parciales

En el desarrollo de este capítulo se comienza con el análisis de diferentes aspectos como el significado de las pruebas de software, sus objetivos y alcance. Se realiza una descripción de los test de unidad, dentro de los cuales se analizaron los tipos de prueba: Caja Blanca y Caja Negra, la ejecución de las pruebas de Caja Negra y los resultados obtenidos en las mismas, la aplicación de métricas donde se evalúan cada uno de los indicadores correspondientes, donde los resultados se comportaron de manera satisfactoria, beneficiando de esta manera que la implementación realizada se valorara de forma satisfactoria.

Conclusiones Generales

Con la finalización del trabajo realizado, se ha confirmado que es verdaderamente importante y necesaria la implementación del módulo Sumario, pues este permitió la informatización de todos los procesos en los que se ve envuelto. Se resolvió de forma eficiente uno de los problemas que más afecta a la Fiscalía General de la República en la actualidad: el llenado de las denuncias y documentos que son necesarios en el en los procesos Sumarios, así como las facturas que se le realizan a los mismos. De manera general se concluye con lo siguiente:

- Se estudiaron de forma satisfactoria, sistemas informáticos vinculados con los procesos Sumarios identificando ventajas y deficiencias de los mismos, permitiendo erradicar los problemas de los sistemas existentes y fusionar las mejores prácticas y funcionalidades.
- Se utilizaron para el correcto funcionamiento de este módulo las metodologías, herramientas y lenguajes acordes a las necesidades del país de migrar al software libre.
- Se realizó un estudio de los estándares de codificación elegidos por la línea de Arquitectura, encargada de seleccionar todo lo relacionado con la arquitectura del proyecto, lo cual permitió entender el por qué de su selección, siendo de gran utilidad en la implementación; ya que sirvió para aprender nuevos métodos y formas de tener organizado el código, para en futuros mantenimientos o iteraciones, mejorar la aplicación.
- Se cumplió el objetivo de la investigación, pues se realizó la implementación del módulo Sumario del proyecto Sistema de Informatización de la Gestión de la Fiscalía, cumpliendo con todos los requisitos establecidos.
- Se realizaron las pruebas de Caja Negra con el objetivo de comprobar la viabilidad de la solución.
- Se realizaron Métricas de diseño con el objetivo de medir atributos importantes en el desarrollo de software.

El sistema resultante logró cubrir todas las funcionalidades previstas, facilitando así el trabajo y la toma de decisiones de los fiscales de la Fiscalía General de la República y demás sedes fiscales que harán uso de la aplicación.

Recomendaciones

Con la culminación de este trabajo se recomienda:

1. Realizar pruebas a los módulos para comprobar el rendimiento del sistema con un mayor volumen de datos en la BD.
2. Realizar un análisis más profundo que permita incorporar nuevas funcionalidades al software.

Bibliografía

Alarcos, Grupos. Grupos Alarcos. *Grupos Alarcos.* [Online] [Cited: Mayo 26, 2011.] http://alarcos.inf-cr.uclm.es/doc/Calidad/capitulo09.ppt.

Álvarez, José García Fanjul y Claudio de la Riva. 2009. [Online] 2009. [Cited: marzo 8, 2010.] http://www.di.uniovi.es/~claudio/isoft/recursos/Requisitos.pdf.

Alvarez, Miguel Angel. 2001. desarrolloweb.com. *desarrolloweb.com.* [Online] Julio 18, 2001. [Cited: Enero 26, 2011.] http://www.desarrolloweb.com/articulos/497.php.

Apache. Apache.org. *Apache.org.* [Online] [Cited: Mayo 29, 2011.] http://httpd.apache.org/docs/2.0/new_features_2_0.html.

Arevalo, Maria Eugenia. 2010. Maria Eugenia Arevalo's Blog. *Maria Eugenia Arevalo's Blog.* [Online] diciembre 2, 2010. [Cited: Marzo 16, 2011.] http://arevalomaria.wordpress.com/2010/12/02/introduccion-al-patron-de-arquitectura-por-capas/.

ArPUG. Grupo de usuarios PostgreSQL de Argentina. *Grupo de usuarios PostgreSQL de Argentina.* [Online] ArPUG. [Cited: febrero 02, 2011.] http://www.arpug.com.ar/trac/wiki/tutorial.html.

Barrios, Johanna Rojas y Emilio. 2007. Grupo ARQUISOFT. [Online] 2007. [Cited: abril 26, 2010.] http://www.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node28.html.

Collado, Manuel. 2003. [Online] marzo 2003. [Cited: abril 26, 2010.] http://www.lml.ls.fi.upm.es/ftp/ed2/0203/Apuntes/pruebas.ppt.

Definicion ABC. 2009. Definicion ABC.com. *Definicion ABC.com.* [Online] Febrero 25, 2009. [Cited: Marzo 5, 2011.] http://www.definicionabc.com/tecnologia/mysql.php.

Doria, Heidi Gonzalez. 2001. UDLAP. *UDLAP.* [Online] Mayo 7, 2001. [Cited: Mayo 29, 2011.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/gonzalez_d_h/capitulo6.pdf.

Duque, Raúl González. Mundo Geek. *Mundo Geek.* [Online] [Cited: Marzo 18, 2011.] http://mundogeek.net/archivos/2010/10/30/frameworks-php/.

Eclipse. Eclipse.org. *Eclipse.org.* [Online] [Cited: Febrero 10, 2011.] http://www.eclipse.org/org/#about.

Fernando. 2006. lacoctelera. [Online] diciembre 1, 2006. [Cited: abril 26, 2010.] http://www.inwebwetrust.net/post/2006/12/01/testeando-varios-valores-un-atributo-un-test-unidad.

Gill, Manuel Torres. Indalog.ual.es. *Indalog.ual.es.* [Online] [Cited: Marzo 16, 2011.] http://indalog.ual.es/mtorres/LP/Prueba.pdf.

Huanca, Daniel. Herrorsoft. *Herrorsoft.* [Online] [Cited: Marzo 16, 2011.] http://herrorsoft.zxq.net/pruebacajablanca.html.

IIS. IIS. *IIS.* [Online] IIS. [Cited: Marzo 5, 2011.] http://www.iis.net/.

Jacobson, Ivar y Booch, Grady y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000.

Leopoldo, Carlos. 19. techtastico. *techtastico.* [Online] septiembre 2008, 19. [Cited: Febrero 24, 2011.] http://techtastico.com/post/7-sistemas-control-versiones/.

Marley, Jimi. Programacion en castellano. *Programacion en castellano.* [Online] [Cited: Enero 29, 2011.] http://www.programacion.com/articulo/por_que_elegir_php_143.

McGraw-Hill. McGraw-Hill. *McGraw-Hill.* [Online] Interamericana de España. [Cited: Enero 10, 2011.] http://www.mcgraw-hill.es/bcv/guide/capitulo/8448146433.pdf.

Najera, Fernando P. TortoiseSVN. *TortoiseSVN.* [Online] [Cited: Marzo 10, 2011.] http://tortoisesvn.net/docs/release/TortoiseSVN_es/tsvn-intro-features.html.

Potencier, Fabien. 2007. Guía definitiva de Symfony. [Online] octubre 21, 2007. [Cited: marzo 8, 2010.] www.librosweb.es.

Poz, Pedro. 2010. Clickear. *Clickear.* [Online] 2010. [Cited: marzo 08, 2010.] http://www.clikear.com/manuales/uml/faseconstruccionbajonivel.aspx.

Sub-Jefatura de Informática. 1999. Scribd. *Scribd.* [Online] Noviembre 1999. [Cited: Febrero 26, 2011.] http://www.scribd.com/doc/43573942/Herramientas-Case.

taller.web. El taller web. *El taller web.* [Online] [Cited: Marzo 4, 2011.] http://zend.netmx.mx/es/productos/info/zend-studio/inicio.

TECOnSITE. TECOnSITE. *TECOnSITE.* [Online] TECOnSITE Soluciones Integrales en Intenet. [Cited: Marzo 2, 2011.] http://www.teconsite.com/programacion-web/ajax.

Valdés, Damián Pérez. 2007. maestros del web. *maestros del web.* [Online] Noviembre 7, 2007. [Cited: Enero 26, 2011.] http://www.maestrosdelweb.com/principiantes/los-diferentes-lenguajes-de-programacion-para-la-web/.

Visual Paradigm. Visual Paradigm. *Visual Paradigm.* [Online] Visual Paradigm. [Cited: Febrero 19, 2011.] http://www.visual-paradigm.com/product/vpuml/.