Universidad de las Ciencias Informáticas

Facultad 15

Titulo: Implementación del Módulo Índice de Peligrosidad Pre-Delictiva del Proyecto Sistema de Gestión Fiscal.

Trabajo de Diploma para optar por el título de Ingeniero Informático.

Autor: Yunesky del Rio Garcia.

Tutor: Ing. Yelena Hernández Estrada.

Curso 2009-2010

Declaración de auditoría

Declaro que soy el único autor de este trabajo y autorizo al <nombre área> de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

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

_____

Firma del Autor

_____

Firma de la Tutora

Agradecimientos

Agradecerles a mis padres, no solo por traerme a la vida, sino por ayudarme a transitar por ella, por brindarme sus manos cuando el camino era inseguro, por darme la luz cuando la oscuridad era intensa, a ellos antes que nadie por hacerme el ser humano que soy.

A mi familia que siempre han estado al tanto de mis estudios y han puesto todos sus rezos en mi porvenir.

A todos mis amigos y a todos aquellos que de una forma u otra me enseñaron muchas cosas de la vida, de todo corazón; muchas gracias, los quiero y los extrañaré mucho.

A nuestro siempre Comandante Fidel Castro por permitirnos formar parte de este proyecto que ya es historia y una realidad.

A mi tutora quien tuvo mucha calma conmigo e hizo más que lo posible por lograr este resultado que hoy se hace realidad. Gracias por influir positivamente en mi formación!

¡¡¡A todos muchas gracias!!!

Dedicatoria

A mis padres, mis hermanos y mi familia que son el motivo de inspiración en este trabajo. Así como a todas aquellas personas que de un modo u otro han formado y forman parte de mi vida y mi corazón....

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 por la existencia de sistemas informáticos que no explotan las posibilidades que brindan las nuevas tecnologías. Entre estos procesos empresariales se encuentran los Procesos Penales que se realizan en las fiscalías, los que permiten los trámites y acciones a tomar desde que un acusado es detenido en una PNR hasta que este es enjuiciado.

El presente trabajo abarca la implementación del módulo Índice de Peligrosidad Pre-delictiva del subsistema Procesos Penales, lo cual es un paso importante para lograr la eficiencia y la calidad en las acciones que se realizan sobre un individuo al ser acusado. 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 muy 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 15 casos de usos los que se han puesto a prueba por el equipo de calidad de la Facultad 3 mediante el uso de pruebas de Caja Negra.

Índice

Introducción

Desde el comienzo de la humanidad han existido conflictos entre los seres humanos los cuales han generado hechos violentos. Con el transcurso del tiempo fue necesario que se celebrara el Contrato Social, mediante el cual un grupo de individuos le entregaba al Estado sus Derechos Naturales para que éste a su vez se los devolviera convertidos en Derechos Civiles. Así se convierte el Estado en el guardián de la seguridad y el bien común, siendo el encargado de dictar el conjunto de normas para regular y organizar la vida social del hombre que constituyen el Derecho y a su vez, se reserva para sí el poder de administrar justicia, que es la jurisdicción.

Las leyes procesales son las que van a permitir a los individuos de la sociedad conocer cómo van a defender sus derechos cuando los vean violentados, y esto es mediante el proceso penal dentro del cual se enmarca el Índice de Peligrosidad Pre-delictiva, que es un conjunto de reglas que, preservando las garantías procesales, le permiten al juez conocer la verdad de los hechos y aplicar la norma que corresponda según la Ley y el Derecho. Los ciudadanos deben conocer a dónde acudir para resolver sus conflictos, es decir, deben saber a qué Tribunal en específico deben asistir.

El sistema Judicial y Legal de la República de Cuba se suscribe en las tradiciones y características del Derecho Continental Europeo, del que tomó las correspondientes instituciones judiciales aún 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 fue aprobada en el año 1976 en referéndum popular por el 97% de los ciudadanos de la nación. Esta constituye la norma jurídica de mayor nivel y está compuesta a su vez por otras leyes, por lo que en ocasiones suele llamarse “Ley Magna” o “Ley de Leyes”. Su principal función es la de determinar los organismos que tienen la facultad y la capacidad legislativa para regir los principios y fundamentos que están contenidos en estas leyes.

A partir de 1970 se inició un proceso para perfeccionar y modernizar el sistema judicial cubano y este fue dirigido fundamentalmente a todas las ramas del Derecho. De esta forma se asegura que hoy en día se

cuenta con un conjunto de normas bien establecidas y que pueden ser aplicadas en la esfera judicial o más específico; en las fiscalías del país pertenecientes a la Fiscalía General de la República (FGR).

Para el control y la preservación de la legalidad sobre la vigilancia del estricto cumplimiento de la Constitución, las leyes y demás disposiciones legales; fue creada la FGR por parte de los organismos del Estado. La FGR es una organización de prestigio nacional que permite agilizar el proceso de toma de decisiones y la eficiencia en el trabajo de los fiscales logrando una mayor conformidad por parte de los ciudadanos cubanos.

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 los expedientes de los acusados 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 haya una mala decisión sobre una condena o en algunos casos, se tiene que posponer el proceso por desconocimiento de los fiscales sobre la totalidad de los sucesos del caso.

Esto es provocado fundamentalmente por el déficit de personal en las aéreas 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.

Se necesita para el logro de un mejor desempeño de los fiscales que los expedientes de los acusados lleguen a las fiscalías correspondientes en el plazo establecido con el objetivo de que se puedan llenar todas las actuaciones en el tiempo requerido y queden reflejadas todas las causas y los agravantes del por qué el individuo ha sido detenido. De esta forma el fiscal tendrá un mayor conocimiento del caso en

estudio y podrá realizar las averiguaciones pertinentes sin verse agobiado por el tiempo. Como constancia de los movimientos realizados sobre uno o varios expedientes, es necesario realizar facturas donde se especifiquen los lugares de procedencia y destino así como los nombres de los implicados en el proceso, y esto debe quedar reflejado en un historial para que en todo momento se puedan consultar las acciones realizadas en el proceso. Para evitar la sobrecarga de trabajo y siendo evidente el déficit de personal, es importante poder asignarle a un fiscal un caso si este es el más idóneo para asumirlo y además cumple con los requisitos de experiencia y tiempo para hacerlo.

Teniendo en cuenta lo expuesto anteriormente se plantea el siguiente **problema a resolver**: Los medios utilizados actualmente en las fiscalías entorpecen el desarrollo de los procesos que se realizan.

En consecuencia, el **objeto de estudio** es: Desarrollo del módulo Índice de Peligrosidad Pre-delictiva, y específicamente el **campo de acción** se centra en la Implementación del módulo Índice de Peligrosidad Pre-delictiva del Proyecto Sistema de Gestión Fiscal.

Para ello se ha planteado el **objetivo general:** Implementar el módulo Índice de Peligrosidad Pre-delictiva del proyecto Sistema de Gestión Fiscal y se defiende la **idea** de que si se implementa el módulo Índice de Peligrosidad Pre-delictiva del Proyecto Sistema Gestión Fiscal entonces mejorarán los procesos que se llevan a cabo en las fiscalías.

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

1. Elaborar el marco teórico de la investigación.
2. Estudiar de 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 Índice de Peligrosidad Pre-delictiva.
6. Validar los resultados.

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

a) **Analítico-Sintético:** Se han estudiado los mecanismos que se utilizan en las diferentes fiscalías del país y se hará uso de la documentación existente de cada uno de los mecanismos y formas de llevar a cabo un proceso penal.

b) **Hipotético deductivo:** La investigación se rige además por un método hipotético deductivo ya que se ha trazado un problema concreto y a partir de este se han planteado los objetivos y posibles soluciones que en el transcurso de la investigación son resueltos mediante el uso de métodos científicamente bien fundamentados.

c) **Sistémico:** Se tiene en cuenta el problema como un todo, con el propósito de desarrollar un sistema flexible y robusto que cumpla con los requisitos exigidos.

Como **método empírico** utilizado en el cumplimiento de las tareas se empleará:

a) La entrevista: Se hará en orden lógico una serie de entrevistas de forma informal a los principales fiscales u otras personas involucradas en los procesos fiscales de la Fiscalía General de la República las que se tomarán como modelo para la obtención de toda la información necesaria.

CAPÍTULO 1

En este capítulo se realizará un estudio del estado del arte haciendo referencia a las tendencias actuales de las tecnologías en el desarrollo del software. Se hará una breve descripción de las técnicas fundamentales de programación (programación estructurada, programación procedimental, programación modular y programación orientada a objetos), de las principales tecnologías de programación del lado del cliente y del lado del servidor, las herramientas que serán utilizadas para el desarrollo de la aplicación y los patrones de arquitectura existente. Esto nos permitirá tener un mejor dominio de las herramientas y técnicas a utilizar.

CAPÍTULO 2

En este capítulo se realizará la descripción y el análisis de la solución propuesta a través de la valoración del diseño propuesto por el analista, estándares de código a utilizar, modelo de la BD, modelos de implementación, diagramas de clases del diseño y la descripción de las principales funcionalidades a automatizar.

CAPÍTULO 3

En este capítulo se validará la solución propuesta mediante pruebas de software según los test de unidad y se evaluará la solución haciendo un resumen de los principales resultados obtenidos.

Capítulo 1

1.0. Introducción

En la actualidad, se convive con grandes avances tecnológicos; y ante la necesidad de una mayor calidad en el trabajo, se hace evidente la creación de aplicaciones que permitan la construcción de software que resuelvan los problemas que se plantean cada día en la sociedad. 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 las diferentes características que estas presentan, 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, ya que muchas veces las herramientas usadas pueden quedar obsoletas, trayendo consigo una considerable disminución en la calidad de dichas aplicaciones.

1.1. 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. Todos estos sucesos serán descritos en este epígrafe.

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:

- **Compatibilidad Multiplataforma**: una misma versión de la aplicación puede correr sin problemas en múltiples plataformas como Windows, Linux, Mac, etc.
- **Actualización:** las aplicaciones web siempre se mantienen actualizadas y no requieren que el usuario deba descargar actualizaciones y realizar tareas de instalación.
- **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.
- **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.

(10ma)

1.2. Técnicas de programación

1.2.1. Programación estructurada

Se basa en una metodología de desarrollo de programas llamada refinamiento sucesivos. Se plantea una operación como un todo y se divide en segmentos más sencillos o de menor

complejidad. Una vez terminados todos los segmentos del programa, se procede a unificar las aplicaciones realizadas por los programadores. Si se ha utilizado adecuadamente la programación estructurada esta integración debe ser sencilla y no debe presentar problemas, y de presentar alguno, será rápidamente detectable para su corrección. La representación gráfica de la programación estructurada se realiza a través de diagramas de flujo o flow chart, el cual representa el programa con sus entradas, procesos y salidas. (10fe3)

1.2.2. Programación orientada a objetos

La Programación Orientada a Objetos (POO u OOP según siglas en inglés) es un paradigma de programación que define los programas en términos de "clases de objetos", objetos que son entidades que combinan estado (datos), comportamiento (procedimientos o métodos) e identidad (propiedad del objeto que lo diferencia del resto). La programación orientada a objetos expresan un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulo más fáciles de escribir, mantener y reutilizar. De esta forma, un objeto contiene toda la información, (los denominados atributos) que permite definirlo e identificarlo frente a otros objetos. (10fe3)

Se puede afirmar que el tipo de programación más idóneo para la implementación de los módulos en cuestión, es la programación orientada a objetos, pues es la 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.

1.3. Tecnologías y plataformas de desarrollo

1.3.1. Tecnologías del lado del servidor

1.3.1.1. Lenguaje Java

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. La implementación

original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrolladas por Sun Microsystems en 1995.

Ventajas:

- Un lenguaje multiplataforma potente, se puede descargar de forma gratuita. Dispone de una gran cantidad de paquetes o librerías que añaden una potencia increíble al lenguaje.
- La integración con la red es asombrosa ya que sus posibilidades son muy altas.

Desventajas:

- Su principal desventaja es el compilador pues este dice si un programa al compilarlo tiene errores pero no dice con precisión el lugar donde está el error. La Máquina virtual Java consume muchos recursos, así que debemos tener un ordenador con muy buenas propiedades sino notaremos mucho la recarga de memoria.
  (10fe7)

1.3.1.2. Lenguaje PHP

PHP (Hipertexto Pre-processor) es software libre, puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, soporta la mayoría de los servidores Web de hoy en día y ofrece soporte para unos 20 gestores de bases de datos. Su característica de ser software libre trae como consecuencia que implique menos costes y servidores más baratos. Es además muy rápido, contiene una biblioteca nativa de funciones sumamente amplia e incluida, no requiere definición de tipos de variables ni manejo detallado del bajo nivel, presenta mejoras de rendimiento, es un lenguaje multiplataforma que funciona en todas las plataformas que soporten Apache. No es un lenguaje de marcas y su principal meta es permitir rápidamente a los desarrolladores la generación dinámica de páginas. Soporta el uso de otros servicios que usen protocolos como IMAP, SNMP, NNTP, POP3, HTTP y derivados. También se pueden abrir sockets de red directos (raw sockets) e interactuar con otros protocolos. Al ser un lenguaje libre dispone de una gran cantidad de características que lo convierten en la herramienta ideal para la creación de páginas Web dinámicas: Integración con varias bibliotecas externas, permite generar documentos en PDF (documentos de Acrobat Reader), analizar código XML. (10fe5)

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 Gestión Fiscal 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.

1.3.2. Tecnologías del lado del cliente

1.3.2.1. 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. (Torre)

1.3.2.2. Tecnología AJAX

Acrónimo de **A**synchronous **J**avaScript **A**nd **X**ML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o **RIA** (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa

> aumentar la interactividad, velocidad y usabilidad en las aplicaciones. Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado (scripting lenguaje) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante *XMLHttpRequest*, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML. (Torre)

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 Sistema de Gestión Fiscal 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 **J**avaScript. 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.

1.4. Herramientas

1.4.1. Herramientas CASE

1.4.1.1. 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 requisitos de información, hasta el modelo físico perfeccionado para la característica específica de la base de datos diseñada. 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 para los principales tipos de 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 datos y a su vez, le permite capturar las reglas del 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 otra 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 y foráneas), valores por defecto, restricciones de campos y dominios. Soporta principalmente bases de datos relacionales SQL y bases de datos que incluyen Oracle, Microsoft SQL Server, Sybase. (10fe1)

1.4.1.2. 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. Ayuda a una rápida construcción de aplicaciones de gran calidad, mejoras y a un menor costo de producción. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Forma parte del IDE de Eclipse. Está diseñado para desarrollar software con Programación Orientada a Objetos, permite reducir la duración del ciclo de desarrollo brindando ayuda a los arquitectos, analistas, diseñadores y desarrolladores. Busca también automatizar tareas tediosas que pueden distraer a los desarrolladores. Dentro de sus características fundamentales están:

- **Multiplataforma:** Soportada en plataformas Java para Sistemas Operativos Windows, Linux y Mac OS X.
- **Interoperabilidad:** Intercambia diagramas UML y modelos con otras herramientas. Soporta exportar e importar a XMI19, XML y archivos Excel. Importa archivos de proyectos de Rational Rose. Integración con Microsoft Office Visio.
- **Modelamiento de los requisitos:** Captura de requisitos con sus respectivos diagramas, modelamiento de casos de uso y análisis textual.
- **Colaboración de equipo:** Realiza el modelado en colaboración y simultáneamente con el Visual Paradigm TeamWork Server y Subversion.
- **Generación de documentos:** Comparte y genera los diagramas y diseños en formatos

como PDF20, HTML y Microsoft Word.

- **Editor de detalles de casos de uso:** Entorno que funciona como un 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.
- **Ingeniería de código:** Permite generación de código e ingeniería inversa en lenguajes como Java, C++, CORBA, IDL, PHP, XML Schema, Ada, Python, C#, VB .NET, ODL, Flash ActionScript, Delphi, Perl y Rugby.
- **Modelado de procesos del negocio:** Visualiza, comprende y mejora los procesos del negocio con herramientas muy completas para estos procesos.
- **Integración con entornos de desarrollo:** Apoyo al ciclo de vida completo de desarrollo del software: análisis, diseño e implementación en IDE como Eclipse, Microsoft Visual Studio, NetBeans, Sun ONE, Oracle JDeveloper, JBuilder y otros.
- **Modelamiento de bases de datos:** Generación de bases de datos, conversión de diagramas entidad-relación a tablas de base de datos, mapeos de objetos y relaciones, ingeniería inversa desde gestores de bases de datos.

(10fe2)

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 Sistema de Gestión Fiscal al seleccionar esta herramienta.

1.4.2. Servidor Web

1.4.2.1. Internet Information Server (IIS)

Es una serie de servicios para los ordenadores que funcionan con Windows. Originalmente era parte del Option Pack para Windows NT. Luego fue integrado en otros sistemas operativos de Microsoft destinados a ofrecer servicios, como Windows 2000 o Windows Server 2003. Windows XP Profesional incluye una versión limitada de IIS. Los servicios que ofrece son: FTP, SMTP,

NNTP y HTTP/HTTPS. Este servicio convierte a un ordenador en un servidor de Internet o Intranet por lo que en las computadoras que tienen este servicio instalado, se pueden publicar páginas web tanto local como remotamente. Los Servicios de Internet Information Services (IIS) proporcionan las herramientas y funciones necesarias para administrar de forma sencilla un servidor Web seguro. Se basa en varios módulos que le dan capacidad para procesar distintos tipos de páginas. Pueden ser incluidos los de otros fabricantes, como PHP o Perl. (Postigo, 2008)

1.4.2.2. Apache

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

Características del Apache:

- Es una tecnología gratuita y de código abierto, lo que proporciona transparencia en todo el proceso de instalación.
- Es prácticamente universal por su disponibilidad en multitud de sistemas operativos.
- Posee una alta configurabilidad en la creación y gestión de logs, de este modo es posible tener un mayor control sobre lo que sucede en el servidor.
- Fácil integración con varios lenguajes de programación como: Java, Perl y especialmente PHP. Dicha relación ha dado lugar al desarrollo de aplicaciones como el APPSERV y XAMPP, los cuales instalan el Apache y el PHP configurados para su uso.

(10fe9)

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 Sistema de Gestión Fiscal 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.

1.4.3. Sistema Gestor de Base de Datos

1.4.3.1. MySQL

Es hoy en día uno de los más importantes en cuanto al diseño y la programación de bases de datos de tipo relacional. Cuenta con millones de aplicaciones y aparece en el mundo informático como uno de los más utilizados por usuarios del medio. Se usa como servidor a través del cual pueden conectarse múltiples usuarios y utilizarlo al mismo tiempo. Una de las características más interesantes de MySQL es que permite recurrir a bases de datos multiusuario a través de la web y en diferentes lenguajes de programación que se adaptan a las necesidades existentes. Por otro lado, es conocido por desarrollar alta velocidad en la búsqueda de datos e información, a diferencia de otros sistemas. Las plataformas que utiliza son de diversos tipos y entre ellas se pueden mencionar; LAMP, MAMP, SAMP, BAMP y WAMP (aplicables a Mac, Windows, Linux, BSD, Open Solaris, Perl y Phyton entre otras). El código fuente se puede descargar y está accesible a cualquiera. Usa la licencia GPL para aplicaciones no comerciales. Es una base de datos muy rápida, segura y fácil de usar. Gracias a la colaboración de muchos usuarios, la base de datos se ha ido mejorando optimizándose en velocidad. Por eso es una de las bases de datos más usadas en Internet. (Victoria, 2009)

1.4.3.2. PostgreSQL

Es un gestor de bases de datos relacional orientada a objetos, libre y gratuito. Está liberado bajo la licencia BSD, lo que significa que se puede disponer de su código fuente, modificarlo a voluntad y redistribuirlo libremente. Contiene bloques de código que se ejecutan en el servidor y pueden ser escritos en varios lenguajes, con la potencia que cada uno de ellos brinda; desde las operaciones básicas de programación, tales como bifurcaciones y bucles, hasta las complejidades de la programación orientada a objetos o la programación funcional. Presenta las siguientes propiedades:

- **Atomicidad:** Asegura la realización de una operación, por lo que ante un fallo del sistema esta no queda a medias.
- **Consistencia:** Posibilita la ejecución de aquellas operaciones que no van a romper las reglas y directrices de integridad de la base de datos.

- **Aislamiento:** Mediante un sistema denominado MVCC (Acceso concurrente multiversión) asegura que una operación no pueda afectar a otras, de esta manera dos transacciones sobre la misma información no genera error.

(10fe11)

Aunque son evidentes las ventajas de ambos gestores de bases de datos, la dirección del proyecto Sistema de Gestión fiscal 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.

1.4.4. IDE de desarrollo

1.4.4.1. Zend Studio

Es considerado como uno de los mejores IDE del momento. Se trata de un programa de la casa Zend, uno de los mayores impulsores de PHP y está orientado a desarrollar aplicaciones web. Es un editor de texto para páginas en PHP que proporciona un buen número de ayudas desde la creación y gestión de proyectos hasta la depuración del código. (10fe13)

1.4.4.2. 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 súper completo, una de las particularidades más interesantes para la comunidad es que es de código libre y gratuito. Está programado en Java, por lo que se necesita el entorno de ejecución de Java instalado para que funcione. Existen diversos plugins o añadidos para proveer de nuevas utilidades al programa, enfocadas a diversos usos que los programadores pueden necesitar. Uno de los añadidos de Eclipse más interesante para los desarrolladores de páginas web, es el módulo para programación en PHP, que está formado de varios componentes. Como beneficio de todas las ventajas para

programar en PHP se puede descargar Eclipse, instalarlo y luego instalar dentro de Eclipse los componentes necesarios. (10fe13)

Aprovechando las comodidades que brinda Eclipse PDT, la directiva del proyecto de Sistema de Gestión Fiscal 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.

1.4.5. SVN (SubVersion)

1.4.5.1. 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, 2008)

1.4.5.2. TortoiseSVN

TortoiseSVN es un cliente gratuito de código abierto para el sistema de control de versiones Subversion. Implementado como una extensión al shell de Windows. Es software libre liberado bajo la licencia GNU GPL. Entre sus características fundamentales podemos citar:

- **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.
- **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.
- **Fácil acceso a los comandos de Subversion:** TortoiseSVN añade su propio submenú al explorador.
- **Versionado de carpetas:** Control de la historia de ficheros individuales.
- **Confirmaciones atómicas:** Esto permite a los desarrolladores construir y confirmar cambios como unidades lógicas.
- **Metadatos versionados:** Cada fichero y directorio tiene un conjunto invisible de "propiedades" adjuntas. Puede inventarse y almacenar cualquier par de clave/valor que se desee. Las propiedades se versionan en el tiempo, igual que el contenido de los ficheros.
- **Elección de capas de red:** Tiene una noción abstracta del acceso al repositorio, proporciona varias características importantes gratis: autentificación, autorización, compresión de la transmisión y navegación del repositorio.
- **Manejo de datos consistente:** Expresa las diferencias entre los ficheros almacenados 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 las personas). Ambos tipos de ficheros se almacenan igualmente comprimidos en el repositorio y las diferencias se transmiten en ambas direcciones por la red.
- **Etiquetado y creación de ramas eficiente:** Crea ramas y etiquetas simplemente copiando el proyecto y utiliza un mecanismo similar a los vínculos duros.
- **Extensibilidad:** No tiene lastre histórico, está implementado como una colección de librerías C compartidas con APIS bien definidas. Esto hace que Subversión sea extremadamente mantenible y se pueda utilizar por otras aplicaciones y lenguajes.

(Vitoria-Gasteiz)

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 Sistema de Gestión Fiscal 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.

1.4.6. Framework

1.4.6.1. php.MVC

Implementa el patrón de diseño Modelo-Vista-Controlador (MVC) y alienta el diseño de aplicaciones basadas en el paradigma del modelo 2. Este modelo de diseño permite que la página web u otros contenidos puedan ser en su mayoría, separados del código y de la aplicación interna (Controller / Model), lo que hace que sea más fácil para los diseñadores y programadores centrarse en sus respectivas esferas de competencia. El framework proporciona un único punto de entrada al controlador y este es el responsable de asignar las peticiones HTTP al gestor de acción correspondiente (Modelo) basándose en las asignaciones de configuración. El modelo contiene la lógica del negocio para la aplicación. El controlador envía la solicitud al componente Vista apropiado que suele ejecutarse mediante una combinación de etiquetas HTML con PHP en forma de plantillas. El contenido resultante se devuelve al navegador del cliente o a través de otro protocolo como SMTP. (10fe14)

1.4.6.2. Symfony framework

Symfony es un completo framework diseñado para optimizar gracias a sus características, el desarrollo de las aplicaciones web. Separa la lógica del negocio, la lógica del servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Automatiza las tareas más comunes permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web. Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de

primer nivel. Es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows.

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

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

(Zaninotto, 2007)

Después de haberse realizado un estudio de algunos de los principales framework para PHP, la directiva del proyecto Sistema de Gestión Fiscal 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.

1.5. Patrones de arquitectura

1.5.1. Estilo Basado en Capas

Es un estilo de programación y su objetivo primordial es organizar la estructura lógica de gran

escala de un sistema en capas separadas de responsabilidades distintas y relacionadas que permiten separar la capa de presentación, capa del negocio y la capa de datos, con una separación clara y cohesiva de intereses ya que las capas "más bajas" son servicios generales de bajo nivel, y las capas más altas son más específicas de la aplicación. La colaboración y el acoplamiento se manejan desde las capas más altas hacia las más bajas.

Figura 1: Estilo Basado en Capas.

Ventajas:

- Desarrollos paralelos en cada capa.
- Aplicaciones más robustas debido al encapsulamiento.
- Mantenimiento y soporte más sencillo que posibilita cambiar un componente y codificar una aplicación monolítica.
- Mayor flexibilidad a la hora de incluir nuevos módulos para dotar al sistema de nuevas funcionalidades.
- Existen tecnologías suficientemente desarrolladas y diseñadas para el paradigma de C/S que aseguran la seguridad en las transacciones, la amigabilidad del interfaz, y la facilidad de empleo.
- La principal ventaja de una aplicación distribuida y bien diseñada, es su buen escalado, es decir, que puede manejar muchas peticiones con el mismo rendimiento simplemente añadiendo más hardware. El crecimiento es casi lineal y no es necesario añadir más código para conseguir esta escalabilidad.

(Peláez, 2009)

1.5.2. Patrón Modelo-Vista-Controlador (MVC)

El Modelo-Vista-Controlador se creó para Smalltalk30 a finales de los setenta. A partir de entonces

su uso se ha ido extendiendo cada día más para la construcción de sistemas software con interfaz gráfica. MVC es un patrón de arquitectura utilizado en sistemas Web para separar los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos permitiendo flexibilidad y facilidad a la hora de hacer futuros cambios. La **Vista** es la información presentada al usuario. Una vista puede ser una página Web o una parte de una página. El **Controlador** actúa como intermediario entre el Modelo, la Vista y cualquier otro recurso necesario para generar una página, es el objeto que proporciona significado a las órdenes del usuario actuando sobre los datos representados por el Modelo. El **Modelo** representa las estructuras de datos. Típicamente el modelo de clases contendrá funciones para consultar, insertar y actualizar información de la base de datos. En los años en los que se creó este patrón, los patrones como se entienden hoy en el mundo de la informática no existían pero a lo largo de estos años, y debido a su gran aplicación en el mundo de las aplicaciones web sobre todo, se ha convertido en uno de los patrones más conocidos. MVC es particularmente apropiada para aplicaciones web interactivas, aplicaciones donde un usuario web interacciona con un sitio web. (Lago, 2007)

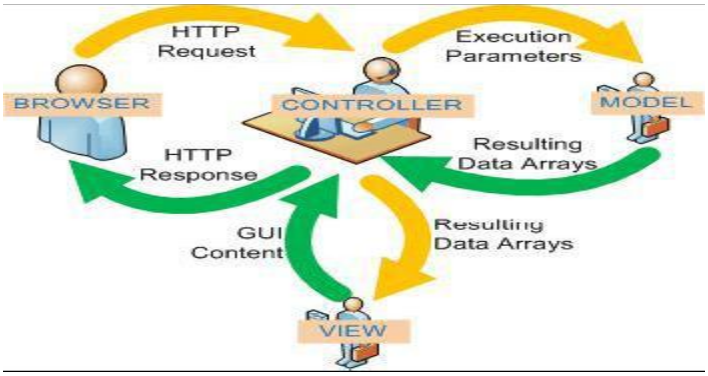

Figura 2: Patrón Modelo-Vista-Controlador.

Siguiendo con la descripción de los leguajes y herramientas escogidos por la dirección del proyecto Sistema de Gestión Fiscal, y como corresponde al lenguaje PHP y al framework Symfony escogidos con anterioridad, se hará uso del patrón modelo-vista-controlador como guía para la arquitectura en la que estará enmarcada la aplicación a desarrollar.

1.6. Pruebas de calidad

1.6.1. Prueba de Caja Blanca

La prueba de la caja blanca del software se encarga de comprobar los caminos lógicos del software proponiendo casos de prueba para que se ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado o mencionado. (D'Onofrio, 2009)

Algunos elementos utilizados alrededor de éste método son los siguientes:

- **Grafo de flujo o grafo del programa:** representa el flujo de control lógico de un programa y se utiliza para trazar más fácilmente los caminos de éste. Cada nodo representa una o más sentencias procedimentales y cada arista representa el flujo de control. (2009)
- **Complejidad ciclomática:** es una métrica de software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto de las pruebas, el cálculo de la complejidad ciclomática representa el número de caminos independientes del conjunto básico de un programa. Esta medida, ofrece al probador de software, un límite superior para el número de pruebas que debe realizar, garantizando así que se ejecutan por lo menos una vez cada sentencia. (2009)
- **Camino independiente:** cualquier camino del programa que introduce, por lo menos, un nuevo conjunto de sentencias de proceso o una nueva condición. (2009)

1.6.2. Prueba de Caja Negra

La prueba de Caja Negra se centra principalmente en los requisitos funcionales del software. Estas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos. La prueba de Caja Negra no es una alternativa a las técnicas de prueba de la Caja Blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la Caja Blanca, como por ejemplo:

- Funciones incorrectas o ausentes.
- Errores de interfaz.

- Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

> Para preparar los casos de pruebas, se necesitan un número de datos que ayuden a la ejecución de estos casos y permitan que el sistema se ejecute en todas sus variantes. Se pueden introducir datos válidos o inválidos para el programa según lo que se desea: hallar un error o probar una funcionalidad. Los datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa corra bien.
> (D'Onofrio, 2009)

Para la validación de los resultados que se obtendrán después de haber desarrollado el software, la directiva del proyecto ha decidido realizar las 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.

1.7. Conclusiones parciales

En este capítulo se ha introducido toda la información necesaria para la comprensión de los pasos seguidos por la directiva del proyecto Sistema de Gestión Fiscal, 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. 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

2.0. Introducción

En este capítulo se delimitan varios puntos importantes en la implementación de los componentes. Se comienza exponiendo los requisitos funcionales detectados 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.

2.1. 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 J. G., 2009)

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 de expedientes de IPP.**

   Se crea el libro de control de expedientes de IPP donde se registran los siguientes datos identificativos del mismo:

   - ✓ No de Expediente de IPP.
   - ✓ Fecha de Radicación.
   - ✓ Tipo de Conducta del pretenso asegurado (Narcomanía, Dipsomanía, Enfermo Mental, Desarrollo Mental Retardado, Embriaguez Habitual, Conducta Antisocial).
   - ✓ Pretenso Asegurado (Nombre del Pretenso Asegurado).
   - ✓ DET(X si la situación procesal es con detenidos).
   - ✓ SDET(X si la situación Procesal es sin detenidos).
   - ✓ Carpeta (Oficial de la PNR que radicó el Expediente de IPP).
   - ✓ Fecha de entrada a la Fiscalía.
   - ✓ Decisión (Salida al Tribunal, Salida a la PNR, Entrada a la Fiscalía de la PNR, Entrada a la Fiscalía del Tribunal).
   - ✓ Fecha de salida de la Fiscalía.

2. **Consultar el libro de control de expedientes de IPP.**

   Una vez creado el libro de control de expedientes de la fiscalía, en cualquier momento este se puede consultar para ver determinada información como la fecha de creación de un expediente etc.

3. **Crear expediente.**

   Se le da de alta en el sistema a un nuevo expediente de IPP que presenta la PNR con los elementos de prueba de cada caso. El mismo tiene como datos identificativos los siguientes:

   - ✓ Denuncia
   - ✓ Datos del pretenso asegurado:
     - Nombre y Apellidos:
     - CI:
     - Hijo de:
     - Vecino de:

   El mismo está compuesto por los siguientes documentos:

   - ✓ Informe conclusivo
   - ✓ Investigación de persona

- ✓ Informe Jefe Sector
- ✓ Informe DTI
- ✓ Charlas profilácticas
- ✓ Dinámica familiar
- ✓ Entrevista
- ✓ Advertencias oficiales
- ✓ Antecedentes policiacos
- ✓ Declaración de testigos
- ✓ Acta de detención
- ✓ Declaración de acusado
- ✓ Comprobación del dicho del pretenso asegurado

Situación Procesal del Pretenso asegurado.

- ✓ Con detenidos.
- ✓ Sin detenidos.

Clasificación del Expediente de IPP.

- ✓ Narcomanía.
- ✓ Dipsomanía.
- ✓ Enfermo Mental.
- ✓ Desarrollo Mental Retardado.
- ✓ Embriaguez Habitual.
- ✓ Conducta Antisocial.

4. **Abrir expediente.**

   Se abre un expediente de IPP que existe previamente en el sistema.

5. **Actualizar automáticamente en el Libro de Control de Expedientes de IPP los datos de un expediente creado.**

   Se actualizan automáticamente en el libro de control de expedientes de IPP los datos requeridos por el libro de cualquier expediente de IPP que sea creado en el sistema.

6. **Actualizar Expediente**

   Agregar a un expediente que ya se encuentra registrado en el sistema y que fue abierto nuevos elementos de prueba del mismo.

7. Guardar expediente.

Se guarda el expediente de IPP previamente creado, en los directorios correspondientes del sistema.

8. Distribuir automáticamente el expediente de IPP al Fiscal correspondiente.

Se envía automáticamente por parte del sistema el expediente de IPP que se acaba de crear al Fiscal que atiende la unidad de la PNR de la que proviene.

9. Crear historial del expediente.

Se crea automáticamente una lista cronológica de los estados en los que se puede encontrar el expediente así como la situación del mismo en el momento en el que se le da de alta al expediente en el sistema, este historial se adjunta al expediente de IPP.

Estados: estados en los que se puede encontrar un expediente de IPP, por devolución, etc.

Situación: es la situación en la que puede encontrarse un expediente de IPP, por ejemplo, con detenidos, o sin detenidos y siempre dejando claro la situación procesal de los acusados.

10. Abrir historial de un expediente.

El historial de un expediente contiene toda la información de la investigación. Por lo antes expuesto este puede abrirse para consultar.

11. Actualizar automáticamente el historial del expediente de IPP por entrada inicial.

Se actualiza en el historial del expediente de IPP la entrada inicial al sistema de este con sus respectivos datos.

Campos del Historial:

- ✓ Origen (PNR, Tribunal, Fiscalía).
- ✓ Fecha de entrada.
- ✓ Destino (PNR, Tribunal, Fiscalía).
- ✓ Fecha de Salida.
- ✓ Estado (Entrada Inicial, Devolución).
- ✓ Situación Procesal de los Pretensos Asegurados (Con Detenidos, Sin detenidos).

12. Actualizar historial de Expediente Automáticamente.

Una vez que el expediente haya transitado por varios procesos este historial debe ser actualizado con los mismos de forma.

13. Actualizar automáticamente un Expediente recién creado en la pantalla del Fiscal que lo atiende.

Una vez creado el expediente y actualizado el libro de control, este expediente debe salir automáticamente en la PC del Fiscal correspondiente.

14. Crear un documento.

En el proceso se crean diferentes documentos para distintas situaciones. Esta funcionalidad debe estar presente.

- ✓ M-9
- ✓ M-10
- ✓ M-11
- ✓ M-12
- ✓ Factura

15. Abrir un documento.

Se debe poder abrir un documento.

16. Actualizar datos de un documento con los cambios de formato para impresión.

El documento se visualiza para ver su vista previa en caso de necesitarlo este puede sufrir modificaciones de formato y estas modificaciones deben actualizarse en el documento.

17. Guardar un documento.

Una vez que un documento ha sido creado este debe ser guardado.

18. Actualizar automáticamente campos de documento con datos extraídos del expediente.

Cuando se cree un documento este debe actualizarse automáticamente con la mayoría posible de los campos del expediente.

19. Habilitar campos editables en documento.

El sistema debe dar la posibilidad de escribir en cada uno de los campos de los documentos.

20. Verificar completitud de los campos editables de un documento.

Una vez que se haya accedido a un documento este no puede ser guardado con campos en incompletos.

21. Verificar que tengan el formato correcto los datos introducidos en un documento.

Verificar que los datos se correspondan con los datos especificados en el documento.

22. Señalar los campos incompletos en un documento.

Una vez que se haya tratado de guardar un documento el sistema tiene que señalar que campos están vacios para que el usuario los complete.

23. Señalar los campos con datos de formato incorrecto en un documento.

Una vez que se hayan encontrado errores en el formato de los datos introducidos el sistema tiene que señalarle al cliente donde se encuentran esos errores para que sean corregidos.

24. Verificar existencia de parámetro de búsqueda.

En caso de hacer una búsqueda el sistema debe verificar que el criterio de búsqueda está presente.

25. Buscar la cantidad de Expedientes de IPP presentados al Fiscal.

Buscar por fiscal la cantidad de expedientes presentados al mismo.

26. Mostrar la cantidad de Expedientes de IPP presentados al Fiscal.

Se mostrará la cantidad de expedientes de IPP presentados a un fiscal determinado.

27. Buscar la cantidad de Expedientes de IPP promovidos al Tribunal

Buscar la cantidad de expedientes que se hayan promovido al tribunal.

28. Mostrar la cantidad de Expedientes de IPP promovidos al Tribunal.

Se mostrará la cantidad de expedientes de Índices de Peligrosidad promovidos al Tribunal.

29. Buscar la cantidad de Expedientes de IPP devueltos al Fiscal.

Se buscará la cantidad de expedientes que han sido devueltos a un Fiscal determinado.

30. Mostrar la cantidad de Expedientes de IPP devueltos al Fiscal.

Se mostrará la cantidad de expedientes de IPP devueltos a un Fiscal particularmente.

31. Buscar la cantidad de Expedientes de IPP con comparecencias celebradas

Se buscará la cantidad de expedientes que se les haya celebrado comparecencia.

32. Mostrar la cantidad de Expedientes de IPP con comparecencias celebradas.

Se mostrará la cantidad de expedientes de IPP con comparecencias celebradas.

33. Buscar la cantidad de Expedientes de IPP con medidas de detención

Se buscará la cantidad de expedientes que poseen medidas de detención.

34. Mostrar la cantidad de Expedientes de IPP con medidas de detención.

Se mostrará la cantidad de expedientes de IPP con medidas de detención.

35. Buscar la cantidad de Expedientes de IPP con apelaciones celebradas

Se buscará la cantidad de expedientes con apelaciones celebradas.

36. Mostrar la cantidad de Expedientes de IPP con apelaciones celebradas.

Se mostrará la cantidad de expedientes de IPP con apelaciones celebradas.

37. Buscar la cantidad total de Expedientes de IPP devueltos a la PNR

Se buscará la cantidad de expedientes que fueron devueltos a la PNR.

38. Mostrar la cantidad total de Expedientes de IPP devueltos a la PNR.

Se mostrará la cantidad total de expedientes de IPP devueltos a la PNR por la Fiscalía Municipal.

39. Buscar la cantidad total de Expedientes de IPP devueltos a la Fiscalía

Se buscará la cantidad de expedientes que fueron devueltos a la Fiscalía

40. Mostrar la cantidad total de Expedientes de IPP devueltos a la Fiscalía.

Se mostrará la cantidad total de expedientes de IPP devueltos a la Fiscalía Municipal.

41. Buscar expediente.

Se busca un expediente de IPP que exista previamente en el sistema, puede ser uno en particular, o varios que cumplan con los mismos parámetros de búsqueda que son los siguientes:

- ✓ Número.
- ✓ Nombre del pretenso asegurado.
- ✓ Apellidos.
- ✓ Provincia.
- ✓ Municipio.
- ✓ Fiscal.
- ✓ Situación.
- ✓ Estado

- ✓ Causa
- ✓ Por PNR
- ✓ Por oficial de la PNR
- ✓ Por un intervalo de tiempo.

42. Mostrar Expediente

Se muestran los expedientes de interés al Fiscal de acuerdo con su criterio de búsqueda. Pueden ser 0, 1 o varios.

43. Buscar Historiales que controlen Expedientes

Se realiza una búsqueda de Historiales de Expedientes de IPP que le interesen consultar al Fiscal, puede ser uno en particular, o varios que cumplan con los mismos parámetros. Los parámetros para realizar estas búsquedas son los siguientes:

- ✓ Número.
- ✓ Nombre del pretenso asegurado.
- ✓ Apellidos.
- ✓ Provincia.
- ✓ Municipio.
- ✓ Fiscal.
- ✓ Situación.
- ✓ Estado
- ✓ Fecha(creación)
- ✓ Fecha (cerrado o archivado)
- ✓ Causa
- ✓ Por PNR
- ✓ Por oficial de la PNR
- ✓ Por un intervalo de tiempo.

44. Mostrar Historiales

Se muestran los historiales de acuerdo al criterio de búsqueda del Fiscal.

45. Realizar indicaciones sobre un Expediente de IPP consultado.

El sistema debe permitir que un Fiscal de una instancia superior ya sea de la FGR o de la Fiscalía Provincial pueda realizar cualquier indicación que considere necesaria para el Fiscal Municipal que lo atiende sobre cualquier expediente de IPP por él consultado.

46. Verificar existencia de indicación.

Verificar que se hayan realizado indicación en el expediente una vez accedido a este para la realización de la misma.

47. Mostrar datos de los Pretensos Asegurados detenidos.

Se mostrarán los datos personales de los pretensos asegurados detenidos así como la fecha de la detención para controlar el término. Los datos que se muestran son los siguientes:

- ✓ Nombres
- ✓ Apellidos
- ✓ Edad
- ✓ Fecha de la detención

48. Desasignar Unidades de la PNR.

El sistema debe dar la posibilidad que se desasignen las estaciones de la PNR que tienen asignadas los Fiscales Municipales.

49. Reasignar unidad de la PNR.

El sistema debe dar la posibilidad de reasignar unidades de la PNR a los Fiscales Municipales.

50. Crear Factura.

Se crea la factura con los datos requeridos.

Datos:

- ✓ De.(Remitente).
- ✓ A(Destinatario).
- ✓ Números de Denuncia(No. Expedientes).
- ✓ Información Adicional.
- ✓ Fecha
- ✓ Entrega

- ✓ Recibe.

51. Abrir Factura.

Se abre la factura deseada.

52. Guardar Factura.

Se guarda la factura en el sistema.

53. Reasignar caso

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

54. Adjuntar indicación a expediente.

Se adjunta a un expediente una indicación realizada sobre el mismo por fiscal superior. La indicación tiene el siguiente formato:

- Datos del usuario que emitió el comentario.
- Comentario emitido.
- Fecha del comentario

A continuación se proponen los requisitos no funcionales que deberá tener el sistema:

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

12. Software

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

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

2.2. 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 expedientes en la PNR**

La creación de expedientes es uno de los primeros pasos que se realizan en los distintos sectores de las PNR cuando un individuo es detenido. Estos expedientes 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 expediente que se está manejando, quedando claras las causas y los agravantes del por qué el individuo ha sido detenido. Este proceso roba un tiempo considerable a las personas designadas, automatizarlo es prioridad en todo sistema que desee mejorar la eficiencia.

- **Creación de documentos en las fiscalías**

La creación de los documentos anexados a los expedientes de los acusados, es un proceso esencial para la toma de decisiones en un caso analizado, aquí quedan claras las violaciones del acusado en cuanto a lo regido por el código penal y las evaluaciones hechas sobre el caso. Una vez llenados estos, serán evidentes los motivos de la detención y los pasos a realizar por parte de la fiscalía. Este mecanismo requiere de mucho trabajo por parte de los fiscales ya que al aumentar el número de casos tiende a ser engorroso el manejo de estos documentos, por lo que se hace necesaria la automatización del mismo en busca de un proceso más eficiente y transparente.

- **Manejo de las facturas de los expedientes de IPP**

Es un proceso donde se hace necesario el control de los movimientos de uno o varios expedientes 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 expedientes 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 expedientes, 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.

2.3. Estándares de código

2.3.1. Estándares de desarrollo para PHP

- **Estándar 1**: La indendación debe ser a cuatro espacios sin caracteres de tabulación. Esto es debido a que ciertos IDE's de desarrollo introducen caracteres de tabulación cuando indendan 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 ultimo 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 palabras, 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.

2.3.2. El sangrado

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

2.3.3. 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.

2.3.4. La terminación de la línea

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

2.3.5. Para mostrar una cadena

Debe estar dentro de comillas dobles o simples (ejemplo: "Hola Mundo", 'Lo que quiero mostrar'). Cabe destacar que si se desea mostrar el símbolo " o ' debe encerrarse en el otro tipo de comillas ("...'...", '..."...')

o usarse un escape (\', \"). Toda línea de instrucción siempre termina en un punto y coma (;), al igual que el lenguaje C.

2.3.6. Insertar un comentario

Para insertar un comentario de una línea debe empezar por // o por #. El resto de la línea es tratado entonces como un comentario.

Para insertar un bloque de comentario, de una o más líneas, se utiliza la combinación /* y */, por ejemplo: /* <COMENTARIOS> */

2.3.7. 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;

}

?>
```

2.3.8. 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.

2.3.9. 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);  ?>
```

2.3.10. Las estructuras de mando

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

```
<?php
 If ((condition1) ||(condition2))
{
    action1;
```

```
 }
else if ((condition3) && (condition4))
{
      action2;
}
else
 {
    defaulaction;
}
?>
```

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.

2.4. Propuesta de solución

2.4.1. Modelo de datos

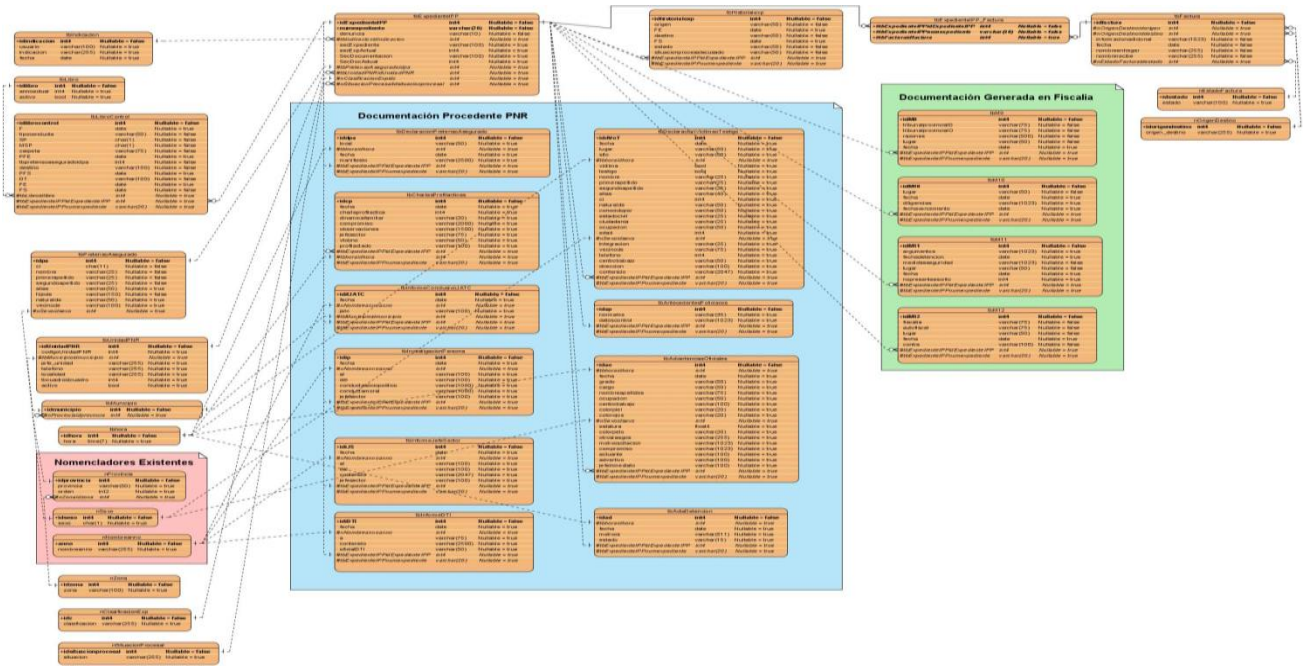

Figura 3: Modelo de datos del sistema

2.4.2. Diagramas de Clases de Diseño

Los Diagramas de Clases de Diseño, se confeccionan durante el flujo de trabajo de Análisis y Diseño de la fase trabajo de Construcción, perteneciente a la metodología RUP. 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 Expedientes de IPP

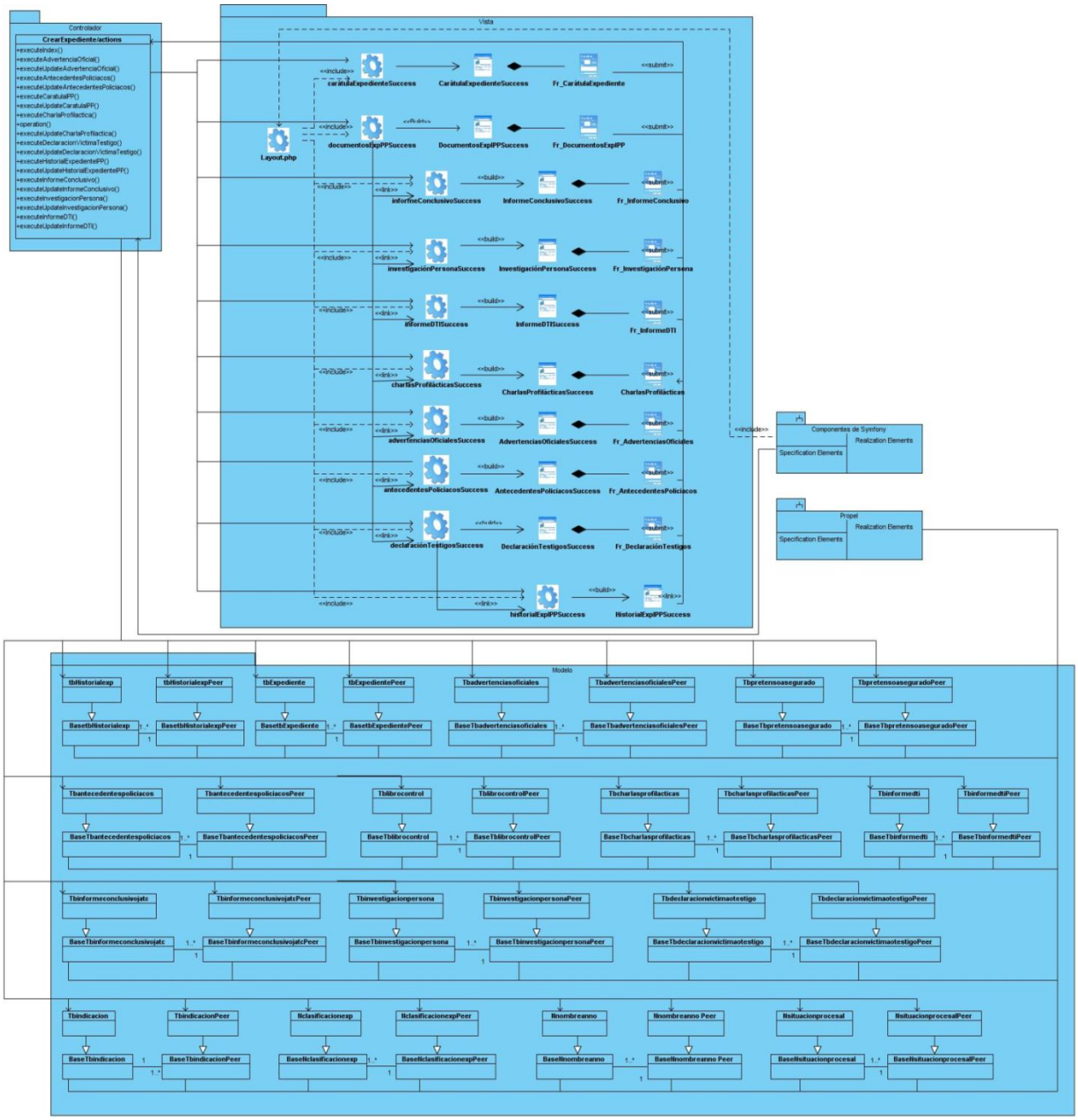

Figura 4: Diagrama de Clases del Diseño del CU: Crear Expediente.

- Crear Documento

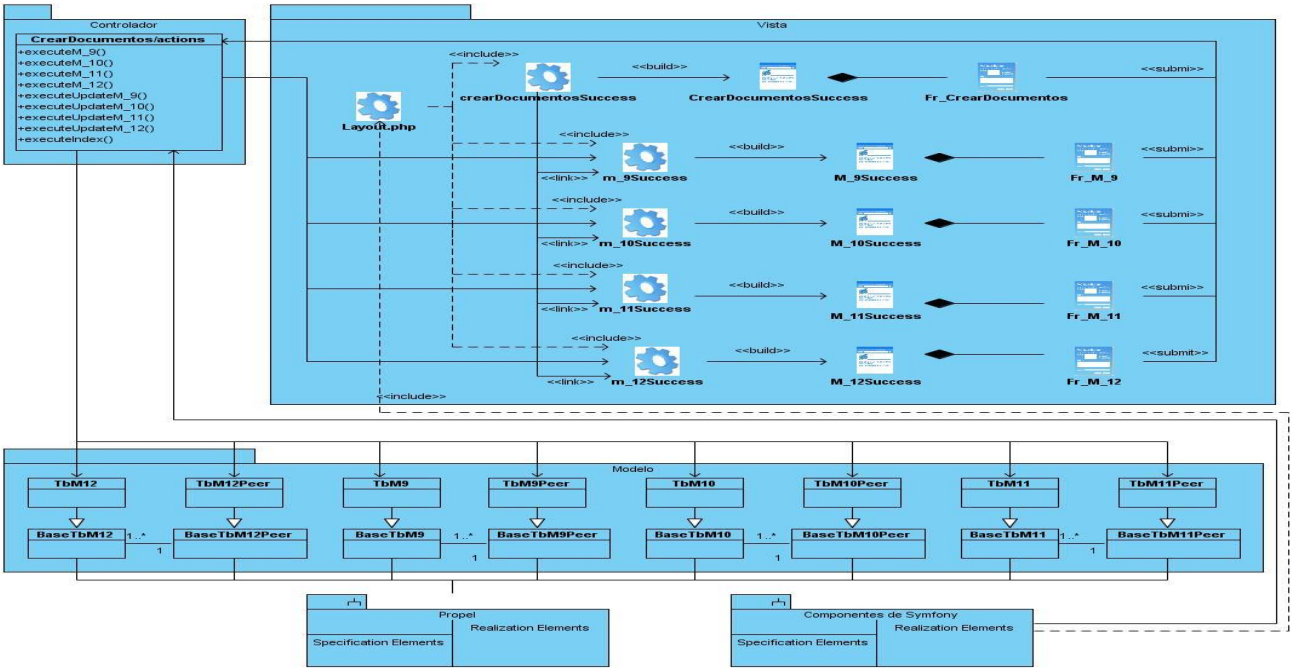

Figura 5: Diagrama de Clases del Diseño del CU: Crear Documentos.

- Manejar Factura

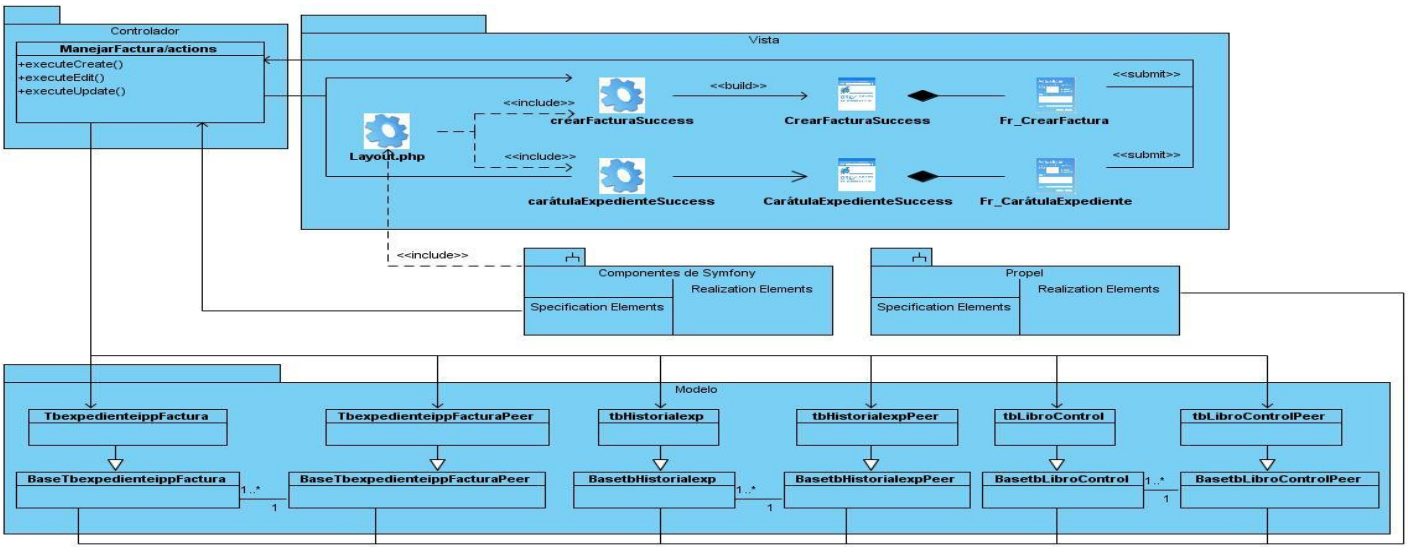

Figura 6: Diagrama de Clases del Diseño del CU: Manejar Factura.

2.4.3. 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 privilegios administrativos otorgados con anterioridad.

Una de las acciones más importantes a realizar, es la creación del Expediente de Índice de Peligrosidad Pre-delictiva, donde se registran todos los datos de interés del individuo que ha sido acusado así como el tipo de conducta que este ha manifestado y se debe especificar si va a ser detenido o no. Este proceso se realiza en sus inicios en la PNR en la cual el pretenso ha sido acusado y posteriormente, de acuerdo a los plazos de tiempos establecidos, este expediente es enviado a la fiscalía en formato duro 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. En esta fase inicial solo se debe llenar la carátula del expediente en cuestión.

Figura 7: Prototipo de Interfaz de la Carátula del Expediente.

Este expediente es asignado a un fiscal que por varios motivos, deberá ser el más idóneo para asumir la tarea y a partir de ese momento, éste procederá a crear los documentos que desee llenar del expediente en cuestión.

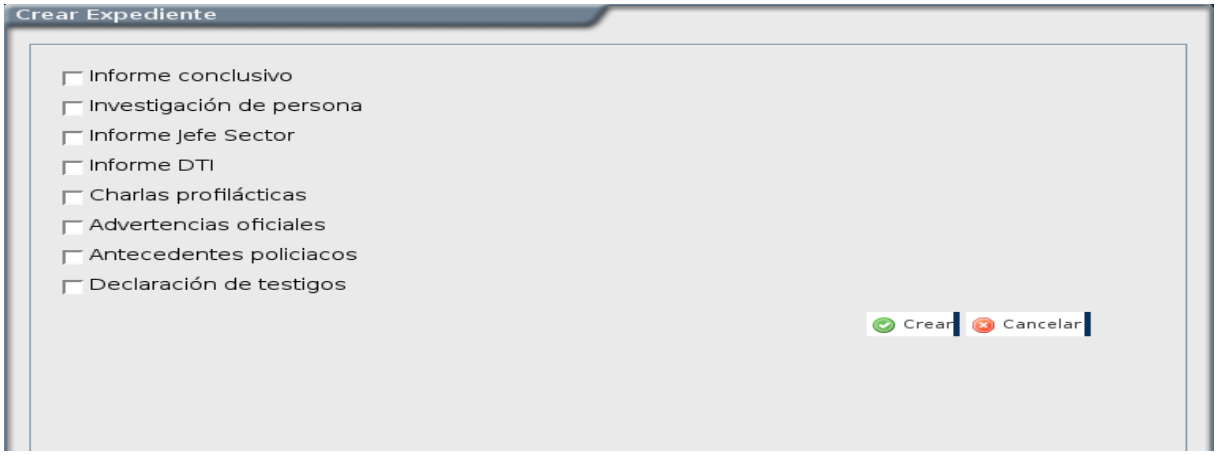

Figura 8: Prototipo de Interfaz de los Documentos del Expediente que deben ser llenados.

Posterior al arribo del expediente del acusado a la fiscalía y ante eventos de apelación u otro tipo, realizados sobre el caso en cuestión, se procede a llenar los documentos correspondientes a estas acciones, donde se reflejan las infracciones del código penal en las que el individuo ha infringido. Para realizar esta acción se le muestran al fiscal los documentos que este puede llenar y este deberá decidir de acuerdo al trámite en proceso cuál o cuáles desea llenar.

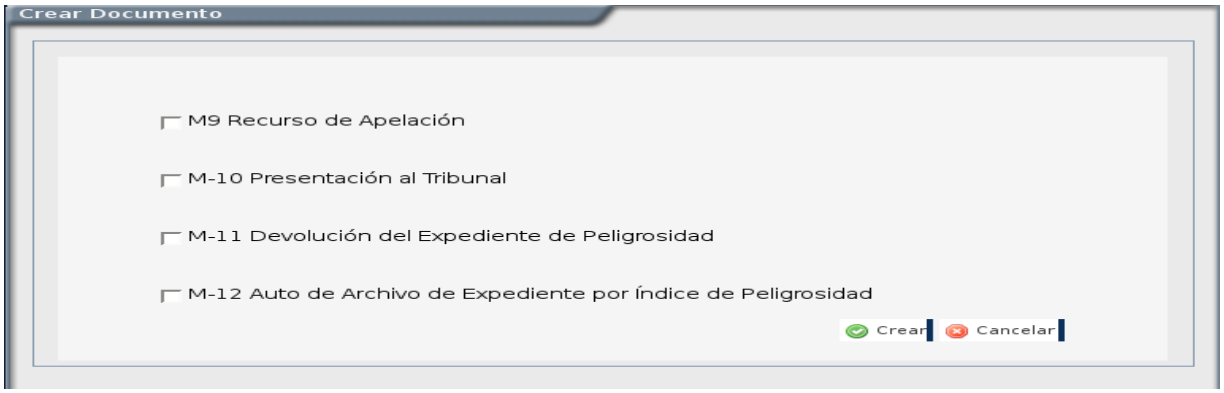

Figura 9: Prototipo de Interfaz del listado de documentos a escoger para su llenado.

Sabiendo que para el movimiento de un expediente 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 los expedientes implicados en el proceso. Además se debe especificar los nombres de los involucrados y la fecha en que se está realizando la acción.

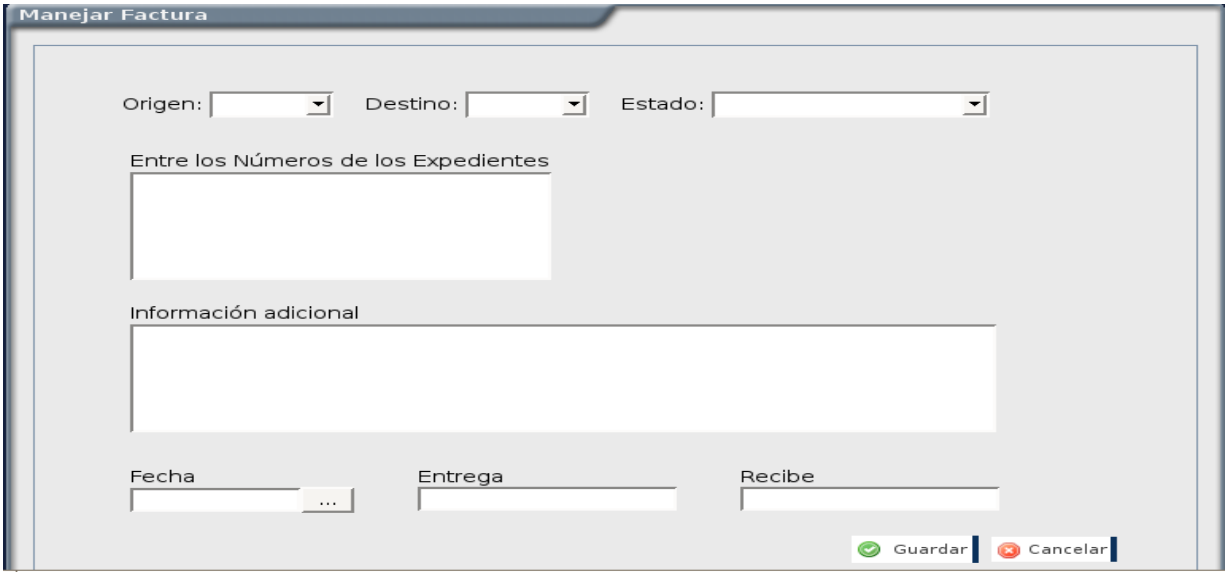

Figura 10: Prototipo de Interfaz de la Factura.

Es de vital importancia el conocimiento del estado en el que se encuentran los expedientes que están involucrados en la factura, por lo que se le informará a la persona que la está tramitando, si alguno de los expedientes es de nuevo ingreso a la fiscalía y debe ser creado en el sistema.

2.5. 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 Índice de Peligrosidad Pre-delictiva, 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.

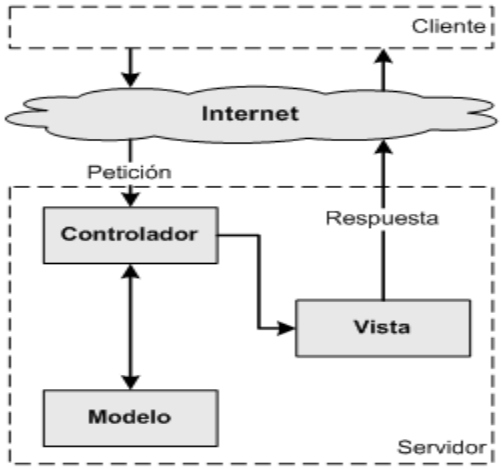

Figura 11: Arquitectura Modelo Vista Controlador.

▪ 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)

2.6. Descripción de clases

2.6.1. 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 Expediente

Nombre: actions	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	**Descripción:**
executeAdvertenciaOficial	Crear un objeto de la tabla tbadvertenciaoficial que responde a los datos que serán introducidos en la vista.
executeUpdateAdvertenciaOficial	Guarda en la tabla tbadvertenciaoficial los datos introducidos en la vista.
executeAntecedentesPoliciacos	Crear un objeto de la tabla tbantecedentespoliciacos que responde a los datos que serán introducidos en la vista.
executeUpdateAntecedentesPoliciacos	Guarda en la tabla tbantecedentespoliciacos los datos introducidos en la vista.
executeCaratulaIPP	Crear un objeto de la tabla tbexpedienteipp que responde a los datos que serán introducidos en la vista.
executeUpdateCaratulaIPP	Guarda en la tabla tbexpedienteipp los datos introducidos en la vista.
executeCharlaProfilactica	Crear un objeto de la tabla tbcharlaprofilactica que responde a los datos que serán introducidos en la vista.
executeUpdateCharlaProfilactica	Guarda en la tabla tbcharlaprofilactica los datos introducidos en la vista.
executeDeclaracionVictimaTestigo	Crear un objeto de la tabla tbdeclaracionvictimatestigo que responde

	a los datos que serán introducidos en la vista.
executeUpdateDeclaracionVictimaTestigo	Guarda en la tabla tbdeclaracionvictimatestigo los datos introducidos en la vista.
executeHistorialExpedienteIPP	Crear un objeto de la tabla tbhistorialexpedienteipp que responde a los datos que serán introducidos en la vista.
executeUpdateHistorialExpedienteIPP	Guarda en la tabla tbhistorialexpedienteipp los datos introducidos en la vista.
executeInformeConclusivo	Crear un objeto de la tabla tbinformeconclusivo que responde a los datos que serán introducidos en la vista.
executeUpdateInformeConclusivo	Guarda en la tabla tbinformeconclusivo los datos introducidos en la vista.
executeInvestigacionPersona	Crear un objeto de la tabla tbinvestigacionpersona que responde a los datos que serán introducidos en la vista.
executeUpdateInvestigacionPersona	Guarda en la tabla tbinvestigacionpersona los datos introducidos en la vista.
executeInformeDTI	Crear un objeto de la tabla tbinformedti que responde a los datos que serán introducidos en la vista.
executeUpdateInformeDTI	Guarda en la tabla tbinformedti los datos introducidos en la vista.

- Crear Documento

Nombre: actions	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	**Descripción:**
executeDocumentoM9	Crea un objeto de la tabla tbm9 que corresponde a los datos introducidos en la vista.
executeUpdateDocumentoM9	Guarda en la tabla tbm9 los datos introducidos en la vista.
executeDocumentoM10	Crea un objeto de la tabla tbm10 que corresponde a los datos introducidos en la vista.
executeUpdateDocumentoM10	Guarda en la tabla tbm10 los datos introducidos en la vista.
executeDocumentoM11	Crea un objeto de la tabla tbm11 que corresponde a los datos introducidos en la vista.
executeUpdateDocumentoM11	Guarda en la tabla tbm11 los datos introducidos en la vista.
executeDocumentoM12	Crea un objeto de la tabla tbm12 que corresponde a los datos introducidos en la vista.
executeUpdateDocumentoM12	Guarda en la tabla tbm12 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.

2.6.2. 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 Expediente

Tipo de clase: Entidad
Nombre de la clase:
Tbadvertenciasoficiales
Tbexpedienteipp
Tbpretensoasegurado

Tbantecedentespoliciacos
TbhistorialExp
Tblibrocontrol
Tblibro
Tbcharlasprofilacticas
Tbinformeconclusivojatc
Tbinformedti
Tbinvestigacionpersona
Tbdeclaracionvictimaotestigo
Tbindicacion
Nclasificacionexp
Nnombreanno
Nsituacionprocesal

- Crear Documento

Tipo de clase: Entidad
Nombre de la clase:
Tbexpedienteipp
Tbm9

Tbm10
Tbm11
Tbm12

- Manejar Factura

Tipo de clase: Entidad
Nombre de la clase:
TbexpedienteippFactura
TbhistorialExp
Tblibrocontrol
Tbfactura

2.6.3. 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 Expediente

Nombre: template	
Para cada responsabilidad:	
Nombre:	**Descripción:**

advertenciaOficialSuccess	Muestra un formulario con todos los datos que el usuario debe llenar.
antecedentesPoliciacosSuccess	Muestra un formulario con todos los datos que el usuario debe llenar.
caratulaIPPSuccess	Muestra un formulario con todos los datos que el usuario debe llenar.
charlaProfilacticaSuccess	Muestra un formulario con todos los datos que el usuario debe llenar.
declaracionVictimaTestigoSuccess	Muestra un formulario con todos los datos que el usuario debe llenar.
historialExpedienteIPPSuccess	Muestra un formulario con todos los datos que el usuario debe llenar.
indexSuccess	Muestra un formulario con todos los datos que el usuario debe llenar.
informeConclusivoSuccess	Muestra un formulario con todos los datos que el usuario debe llenar.
investigacionPersonaSuccess	Muestra un formulario con todos los datos que el usuario debe llenar.
informeDTISuccess	Muestra un formulario con todos los datos que el usuario debe llenar.
indexSuccess	Muestra un listado de todos los documentos donde el usuario puede escoger cuál desea llenar.

- Crear Documento

Nombre: template	
Para cada responsabilidad:	
Nombre:	**Descripción:**
documentoM9Success	Muestra un formulario con todos los datos que el usuario debe llenar.
documentoM10Success	Muestra un formulario con todos los datos que el usuario debe llenar.
documentoM11Success	Muestra un formulario con todos los datos que el usuario debe llenar.
documentoM12Success	Muestra un formulario con todos los datos que el usuario debe llenar.
indexSuccess	Muestra un listado de todos los documentos donde el usuario puede escoger cuál desea llenar.

- Manejar Factura

Nombre: template	
Para cada responsabilidad:	
Nombre:	**Descripción:**
editSuccess	Muestra un listado de todos los campos de la tabla tbFactura.

2.7. 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 Decorator. De forma general se logró implementar el módulo Índice de Peligrosidad Pre-delictiva, perteneciente al subsistema Procesos Penales del proyecto Sistema de Gestión Fiscal.

Capítulo 3

3.0. 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 Índice de Peligrosidad Pre-delictiva, así como verificar el rendimiento interno de las mismas, además de evaluar mediante métricas el diseño propuesto por los analistas.

3.1. 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)

3.1.1. 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)

3.1.2. 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.

Se invierte tiempo y esfuerzo en los detalles de control debido a que:

- Los errores suelen estar en situaciones fuera de las normales.
- A menudo caminos que se piensa que tienen pocas posibilidades de recorrerse, son recorridos regularmente.
- Los errores tipográficos son aleatorios. Puede que no sean detectados por los procesadores de la sintaxis del lenguaje particular y estar presentes en cualquier camino lógico.

(Collado, 2003)

3.2. 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)

3.2.1. 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. (Barrios, 2007)

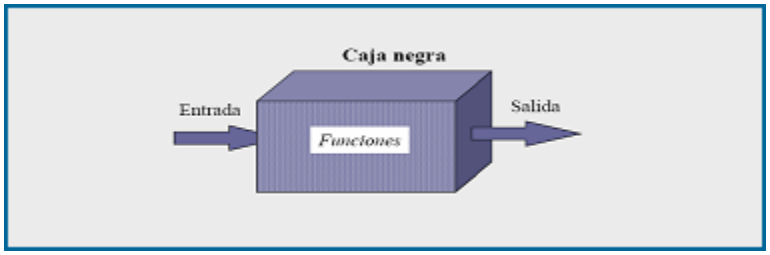

Figura 13: Prueba de Caja Blanca

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.

  Las pautas para desarrollar casos de prueba con esta técnica son:

  - ✓ Si una condición de entrada especifica un rango de valores, se diseñarán casos de prueba para los dos límites del rango, y otros dos casos para situaciones justo por debajo y por encima de los extremos.
  - ✓ Si una condición de entrada especifica un número de valores, se diseñan dos casos de prueba para los valores mínimo y máximo, además de otros dos casos de prueba para valores justo por encima del máximo y justo por debajo del mínimo.
  - ✓ Aplicar las reglas anteriores a los datos de salida.
  - ✓ Si la entrada o salida de un programa es un conjunto ordenado, habrá que prestar atención a los elementos primero y último del conjunto.

3.3. Aplicación de Pruebas de Caja Negra

Para la aplicación de este tipo de prueba se usará el requisito Crear Expediente. El objetivo de este requisito es la creación de un expediente 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 el expediente en formato duro que está digitalizando. Se realizarán una serie de acciones automáticas como son el llenado y actualización del Historial del Expediente y del Libro de Control.

3.3.1. 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 crear Factura.	1.1. El sistema muestra la interfaz “Manejar factura” con los siguientes datos a llenar: • Origen • Destino • Estado • Expediente • 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 los expedientes 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 expedientes.
	EC 1.3: El Gestor común marca el botón Cancelar.	1.3. El sistema cierra la interfaz y muestra la Página Principal.

3.3.2. Descripción de las variables

No	Nombre del campo	Clasificación	Puede ser nulo	Descripción
1	Origen	object_select_tag	NO	Se debe seleccionar como tipo el origen de los expedientes.
2	Destino	object_select_tag	NO	Se debe seleccionar como tipo el destino de los expedientes.
3	Estado	object_select_tag	NO	Se debe seleccionar como tipo el estado de los expedientes que dependerá del origen y el destino

				anteriormente seleccionados.
4	Listado de expedientes	textarea_tag	NO	Se deben introducir el número de los expedientes a los que se les desea realizar la factura.
5	Información adicional	object_textarea_tag	SI	Se incluirá una información adicional a la factura en caso de que así lo requiera.
6	Fecha de la factura	object_input_date_tag	NO	Se debe seleccionar una fecha para la factura que se está procesando.
7	Nombre de la persona que entrega el o los expedientes	object_input_tag	NO	Se debe introducir el nombre de la persona que hace la entrega de los expedientes.
8	Nombre de la persona que recibe el o los expedientes	object_input_tag	NO	Se debe introducir el nombre de la persona que recibe los expedientes.

3.4. 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. Se evaluó el diseño propuesto por los analistas mediante el uso de métricas donde se pudo apreciar como cada uno de los indicadores pertenecientes a

las métricas evaluadas, se comportaron de manera satisfactoria, beneficiando de esta manera que el diseño realizado 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 Índice de Peligrosidad Pre-delictiva, 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 los expedientes y documentos que son necesarios en el en las acusaciones pre-delictivas, 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 Índices de Peligrosidad Pre-delictiva 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 Índice de Peligrosidad Pre-delictiva del proyecto Sistema de Gestión Fiscal, cumpliendo con todos los requisitos establecidos.
- Se realizaron las pruebas de Caja Negra con el objetivo de comprobar la viabilidad de la solución.

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. Migrar la aplicación a versiones superiores del framework en busca de un mayor rendimiento y funcionalidad.
2. Realizar pruebas a los módulos para comprobar el rendimiento del sistema con un mayor volumen de datos en la BD.
3. Realizar un análisis más profundo que permita incorporar nuevas funcionalidades al software.

Referencias bibliográficas

(s.f.). Recuperado el 10 de febrero de 2010, de http://www.monografias.com/trabajos24/herramientas-case/herramientas-case.shtml#ejempl

(s.f.). Recuperado el 10 de febrero de 2010, de http://www.monografias.com/trabajos24/herramientas-case/herramientas-case.shtml#ejempl

(s.f.). Recuperado el 10 de febrero de 2010, de http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/

(s.f.). Recuperado el 10 de febrero de 2010, de http://www.lenguajes-de-programacion.com/programacion-estructurada.shtml

(s.f.). Recuperado el 10 de febrero de 2010, de http://teleformacion.edu.aytolacoruna.es/PASCAL/document/modular.htm

(s.f.). Recuperado el 10 de febrero de 2010, de http://www.desarrolloweb.com/articulos/303.php

(s.f.). Recuperado el 10 de febrero de 2010, de http://es.csharp-online.net

(s.f.). Recuperado el 10 de febrero de 2010, de http://www.webtaller.com/manual-java

(s.f.). Recuperado el 10 de febrero de 2010, de http://www.alegsa.com.ar/Dic/apache.php

(s.f.). Recuperado el 10 de febrero de 2010, de http://www.xolucionesinformaticas.com/index.php?option=com_content&view=article&id=77&Itemid=85

(s.f.). Recuperado el 10 de febrero de 2010, de http://www.netpecos.org/docs/mysql_postgres/x15.html

(s.f.). Recuperado el 10 de febrero de 2010, de http://www.arpug.com.ar/trac/wiki/tutorial.html

(s.f.). Recuperado el 10 de febrero de 2010, de http://www.guia-ubuntu.org/index.php?title=PgAdmin_III

(s.f.). Recuperado el 10 de febrero de 2010, de http://www.tufuncion.com/ide-php

(s.f.). Recuperado el 10 de febrero de 2010, de http://www.phpmvc.net/

(s.f.). Recuperado el 10 de febrero de 2010, de http://kumbia.sourceforge.net/

(s.f.). Recuperado el 10 de febrero de 2010, de http://www.phpontrax.com/

(s.f.). Recuperado el 10 de febrero de 2010, de http://www.proactiva-calidad.com/java/patrones/mvc.html

(s.f.). Recuperado el 10 de febrero de 2010, de http://www.csae.map.es/csi/silice/Global71.html

(s.f.). Recuperado el 8 de marzo de 2010, de http://www.logindesarrollos.com/es/Servicios/Desarrollo-de-aplicaciones-online

(19 de septiembre de 2008). Recuperado el 10 de febrero de 2010, de http://techtastico.com/post/7-sistemas-control-versiones/

(mayo de 2009). Recuperado el 28 de marzo de 2010, de http://www.lcc.uma.es/~av/Libro/CAP1.pdf

(02 de marzo de 2009). Recuperado el 10 de febrero de 2010, de http://www.todeto.net/soft-plugins/669-zend-studio-6-01-ide-php-descarga.html

Álvarez, I. F. (s.f.). Recuperado el 10 de febrero de 2010, de http://www.google.com.cu/#hl=es&q=Programaci%C3%B3n+procedimental&meta=lr%3Dlang_es&aq=f&oq=Programaci%C3%B3n+procedimental&fp=4bc4de680f947bd3

Álvarez, J. G. (2009). Recuperado el 8 de marzo de 2010, de http://www.di.uniovi.es/~claudio/isoft/recursos/Requisitos.pdf

Barrios, J. R. (2007). *Grupo ARQUISOFT*. Recuperado el 26 de abril de 2010, de http://www.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node28.html

Collado, M. (marzo de 2003). Recuperado el 26 de abril de 2010, de http://www.lml.ls.fi.upm.es/ftp/ed2/0203/Apuntes/pruebas.ppt

Criado, A. B. (04 de febrero de 2008). Recuperado el 10 de febrero de 2010, de http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=xampp

D'Onofrio, D. L. (marzo de 2009). Recuperado el 10 de febrero de 2010, de http://www.elguille.info/Clipper/probando.htm

Fernández, J. A. (1999). Recuperado el 26 de abril de 2010, de www.rogeliodavila.com/tcs/.../Parte_16_TestWhite.ppt

Fernando. (1 de diciembre de 2006). *lacoctelera*. Recuperado el 26 de abril de 2010, de http://www.inwebwetrust.net/post/2006/12/01/testeando-varios-valores-un-atributo-un-test-unidad

Lago, R. (Abril de 2007). Recuperado el 10 de febrero de 2010, de http://www.proactiva-calidad.com/java/patrones/index.html#algunos_patrones

Leopoldo, C. (19 de septiembre de 2008). Recuperado el 10 de febrero de 2010, de http://techtastico.com/post/7-sistemas-control-versiones/

Peláez, J. (29 de mayo de 2009). Recuperado el 10 de febrero de 2010, de http://geeks.ms/blogs/jkpelaez/archive/2009/05/29/arquitectura-basada-en-capas.aspx

Postigo, J. V. (11 de febrero de 2008). Recuperado el 10 de febrero de 2010, de http://javiervidal.net/los-servidores-web-mas-utilizados/

Potencier, F. (21 de octubre de 2007). *Guía definitiva de Symfony.* Recuperado el 8 de marzo de 2010, de www.librosweb.es

Poz, P. (2010). *Clickear*. (P. Pedro, Editor, & P. Pedro, Productor) Recuperado el 08 de marzo de 2010, de Clickear: http://www.clikear.com/manuales/uml/faseconstruccionbajonivel.aspx

sd. (sd). sd. En sd (Ed.), *ds. ds*, pág. sd. sdsds: ds.

Torre, A. d. (s.f.). Recuperado el 10 de febrero de 2010, de http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html

Victoria. (25 de febrero de 2009). Recuperado el 10 de febrero de 2010, de http://www.definicionabc.com/tecnologia/mysql.php

Vitoria-Gasteiz. (s.f.). Recuperado el 10 de febrero de 2010, de http://www.google.com.cu/url?sa=t&source=web&ct=res&cd=10&ved=0CB0QFjAJ&url=http%3A%2F%2Fww.ejie.net%2Fdocumentos%2FHerramientas%2FTortoiseSVN.%2520Manual%2520de%2520usuario.doc&rct=j&q=TortoiseSVN+%2B+concepto&ei=9pFzS4PjN6imtgf6gsWDCg&usg=AFQjCNH1uMJCj

Zaninotto, F. P. (21 de octubre de 2007). Recuperado el 10 de febrero de 2010, de http://www.symfony-project.org/

Glosario de términos

- ✓ Un **framework** simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, un framework proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, un framework facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas.
- ✓ **Lenguajes:** Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una computadora. Consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Un lenguaje de programación permite a uno o más programadores especificar de manera precisa: sobre qué datos una computadora debe operar, cómo deben ser estos almacenados, transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural.
- ✓ **Lenguajes del lado del servidor:** Se les clasifica así a los lenguajes de programación en la tecnología cliente servidor que se ejecutan del lado del servidor y de los que los cuales los usuarios solo obtienen el beneficio del procesamiento de la información.
- ✓ **Lenguajes del lado del cliente:** Un lenguaje del lado cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio. Pero la página no se verá bien si el ordenador cliente no tiene instalados los plug-in adecuados. El código, tanto del hipertexto como de los scripts, es accesible a cualquiera y ello puede afectar a la seguridad.
- ✓ **Herramientas CASE:** “CASE es una sigla, que corresponde a las iniciales de: **C**omputer **A**ided **S**oftware **E**ngineering; y en su traducción al Español significa Ingeniería de Software Asistida por Computación. Las herramientas CASE representan una forma que permite Modelar los Procesos del negocios. Un elemento importante conveniente de destacar, es que las herramientas CASE, son eso: "HERRAMIENTAS", y que como tales permiten aumentar la productividad en el desarrollo de un proyecto y como herramientas que son, deben ser aplicadas a una metodología determinada.”
- ✓ **SVN (SubVersion):** SVN (SubVersion) es un sistema de control de versiones, que mantiene los registros de todos los cambios que se han realizado a los archivos de un software, lo que permite

el trabajo de distintos desarrolladores en un mismo proyecto, está herramienta es muy usada por los programadores de software libre. Existen dos razones fundamentales para el uso de esta herramienta:

- Gestiona las modificaciones durante el desarrollo.
- Permite que varias personas trabajen sobre los mismos ficheros.

- ✓ **Arquitectura:** "Una arquitectura orientada a objetos bien estructurada está llena de patrones. La calidad de un sistema orientado a objetos se mide por la atención que los diseñadores han prestado a las colaboraciones entre sus objetos. Los patrones conducen a arquitecturas más pequeñas, más simples y más comprensibles." Uno de los elementos bases del proceso de desarrollo de software es diseñar la Arquitectura de Software. Esencialmente sobre ella se sustentan todos los mecanismos de diseño y representaciones de la estructura general de la aplicación a desarrollar. De la cohesión, utilidad y flexibilidad de los componentes de la arquitectura dependerán la calidad final y la utilidad del software. La correcta definición del estilo arquitectónico a utilizar, patrones y mecanismos de diseño es la raíz de lo anteriormente descrito.
- ✓ **Tecnología:** es el conjunto de conocimientos, ordenados científicamente, que permiten construir objetos y máquinas para adaptar el medio y satisfacer nuestras necesidades. Es una palabra de origen griego, τεχνολογία, formada por téchnē (τέχνη, "arte, técnica u oficio") y logía (λογία), el estudio de algo. Aunque hay muchas tecnologías muy diferentes entre sí, es frecuente usar el término en singular para referirse a una de ellas o al conjunto de todas. Cuando se lo escribe con mayúscula, tecnología puede referirse tanto a la disciplina teórica que estudia los saberes comunes a todas las tecnologías, como a educación tecnológica, la disciplina escolar abocada a la familiarización con las tecnologías más importantes. La actividad tecnológica influye en el progreso social y económico, pero también ha producido el deterioro de nuestro entorno (biosfera). Las tecnologías pueden ser usadas para proteger el medio ambiente y para evitar que las crecientes necesidades provoquen un agotamiento o degradación de los recursos materiales y energéticos de nuestro planeta. Evitar estos males es tarea no sólo de los gobiernos, sino de todos. Se requiere para ello una buena enseñanza-aprendizaje de la tecnología en los estudios de enseñanza media o secundaria y buena difusión de los problemas, diagnósticos y propuestas de solución en los medios de comunicación social.
- ✓ **El aseguramiento de calidad del software** es el conjunto de actividades planificadas y

sistemáticas necesarias para aportar la confianza en que el producto (software) satisfará los requisitos dados de calidad. Se diseña para cada aplicación antes de comenzar a desarrollarla y no después.

- ✓ **Las pruebas** es una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requisitos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente.
- ✓ **La indendación** significa mover un bloque de texto hacia la derecha insertando espacios o tabuladores para separarlo del texto adyacente, lo que en el ámbito de la imprenta se ha denominado siempre como sangrado o sangría. Se utiliza para mejorar la legibilidad del código fuente por parte de los programadores y su uso es determinado por los diferentes lenguajes de implementación que lo usan.