UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 3

Implementación del subsistema Penal hasta la sentencia del Proyecto de Informatización de Tribunales

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

Autor: Yasmani Gutiérrez Pérez

Tutor: Ing. Darián González Ochoa

Curso 2011-2012

Declaración de Autoría

Declaro ser autor del presente trabajo de diploma y recomiendo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____	_____
Yasmani Gutiérrez Pérez	Darián González Ochoa
Autor	Tutor

Dedicatoria

A las personas más importantes de mi vida, mis padres Mayda y Ramón.
A mis hermanas Anay y Noelvia.
A mis amigos Rances, Lisbeth, Ivis, Yordi y Ariadna.

Agradecimientos

A mis padres por vivir para mí, por tanto sacrificio para conmigo, por su apoyo, preocupación, por darme confianza y ánimo, por ser como son, por aceptarme y quererme tanto tal y como soy.

A mis hermanas porque sin su colaboración y sus regaños no hubiera sido posible este logro.

A los tutores, en especial a mi cotutor Joan Jon por su preocupación y colaboración con el trabajo, por ayudarme, enseñarme y guiarme en el proyecto.

A la profesora Cristina por sus charlas y consejos.

A todas mis amistades en estos 5 años, a los de mi grupo en primer año, a los del 3505, al piquete del apartamento 9109, que aunque sin éxito, se esforzaron mucho para que no los visitara tan frecuentemente.

A mis amigos Lisandra, Irmel, Yoandra, Maidelin, Reinier y Daina.

A Ivis por confiar en mí, por contagiarme con su alegría y sus locuras.

A Ariadna por las horas de sol que compartimos, por preocuparse por mí y por los flanes.

A mi hermano Rances por ayudarme tanto, muchas veces sin que se lo pidiera, por preocuparse tanto por mí, por darme ánimo.

A Yordi por ser mi mejor amigo, por confiar en mí, por ayudarme en los momentos más difíciles, por darme la posibilidad de contar con él cuando lo necesite.

A Lisbeth por soportar mis conversaciones aburridas, por su comprensión, sus consejos y sus gritos.

Resumen

Con el fin de eliminar una serie de dificultades que hacen en ocasiones ineficientes las labores en los Tribunales Populares Cubanos (TPC) surge la necesidad de crear un sistema que informatice los procesos judiciales que se ejecutan en los TPC. La aplicación que se construye con tales fines está dividida en varios módulos, Penal, Administrativo, Civil, Laboral y Económico donde cada uno responde a las diferentes materias de los TPC, además existen dos módulos Administración y gobierno y Común que están enfocados al sistema.

El presente trabajo tiene como objetivo la especificación de una serie de artefactos que resultan necesarios para una satisfactoria implementación del subsistema Penal hasta la sentencia y su posterior despliegue como parte del Sistema de Informatización de los Tribunales Populares Cubanos en sus diferentes instancias. Para ello se realiza una descripción de las herramientas y otras tecnologías que se utilizan en el desarrollo del subsistema, se describe parte del Modelo de diseño y el de implementación. Se evalúa la calidad de la solución mediante la aplicación de métricas y se ejecutan pruebas de caja blanca utilizando la técnica del camino básico para detectar y corregir errores internos de las funcionalidades.

ÍNDICE

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Ante el indetenible desarrollo de las Tecnologías de la Informática y las Comunicaciones (TIC) y su influencia en la dinámica de los procesos sociales en el mundo actual, donde el tiempo es un recurso cada vez más importante y escaso, se hace imprescindible el uso de las tecnologías para que los servicios a la población sean más ágiles, efectivos y que estén a la altura de una sociedad que está siempre en movimiento. Es por eso que numerosos países han involucrado ampliamente las TIC en las diferentes ramas de la sociedad incluida la jurídica, buscando los beneficios que brinda su utilización.

Cuba no es una excepción, y aquí no se está al margen de las influencias que provocan los avances tecnológicos, es por ello que también se le ha dado la importancia debida al desarrollo de sistemas informáticos que ayuden a humanizar las tareas, ahorrar recursos valiosos, agilizar y elevar la calidad de los servicios que se prestan al pueblo. La Universidad de las Ciencias Informáticas (UCI) cumple un papel protagónico en las tareas de la informatización de la sociedad cubana, y como en cada uno de sus centros de desarrollo; en el Centro de Gobierno Electrónico (CEGEL); se labora hoy en día en conjunto con los Tribunales Populares Cubanos (TPC) en la creación de una aplicación que informatice y supervise los procesos judiciales de acuerdo al conjunto de disposiciones legales que los regulan.

En el Sistema de Informatización de los Tribunales (SIT); el desarrollo está dividido en varios módulos o subsistemas que responden respectivamente a cada una de las diferentes materias en las que se distribuyen los procesos judiciales de los Tribunales, las materias son: Administrativa, Civil, Económica, Laboral y Penal. Debido a la complejidad y extensión de los procesos a informatizar en la materia Penal, se encuentra dividido en dos módulos, Penal hasta la sentencia y Penal después de la sentencia.

Actualmente en el trabajo diario en la sección de lo Penal en los Tribunales, se maneja un gran número de documentos, expedientes y realizan múltiples tramitaciones, todo de forma manual, lo que trae como consecuencia que se produzcan frecuentemente, errores previsibles en la confección de documentos y que las tramitaciones y los procesos se realicen más lentos dado el gran número de archivos que se almacenan en formato físico, resultando tedioso consultar los datos y realizar búsquedas sobre ellos.

Con la implementación paulatina del sistema en los tribunales si bien no se solucionarán todas las dificultades, si se le facilitará el trabajo a los jueces y otros trabajadores. Para hacer realidad esta aplicación se hace necesario, teniendo en cuanta los requisitos definidos en el análisis, la especificación de los artefactos: diagrama de clases, diagrama de componentes y modelo de despliegue para lograr una satisfactoria implementación de la aplicación.

Atendiendo a lo anteriormente planteado se identifica como **problema a resolver**: ¿Cuál es la especificación de los artefactos necesarios para la informatización de los requisitos identificados para el proceso ordinario de la materia penal?

Definiéndose como objeto de estudio **objeto de estudio**: El proceso de desarrollo de software en los sistemas para la gestión judicial y el **objetivo general** de este trabajo se describe como: Realizar la implementación del subsistema Penal hasta la sentencia de modo que se informaticen los requisitos identificados para el proceso ordinario de la materia penal.

Partiendo del objetivo general se define como **campo de acción:** El proceso de desarrollo de software en la gestión de los procesos judiciales para la materia Penal.

Conforme a lo antes descrito se define como **idea a defender**: Con la especificación de los artefactos necesarios se informatizan los requisitos identificados para el proceso ordinario en la materia penal.

Para lograr dar solución al problema planteado y derivándose del objetivo general se trazan los siguientes **objetivos específicos**:

- ✓ Realizar el marco teórico de la investigación para fundamentar las herramientas y tecnologías definidas para el desarrollo del subsistema Penal.
- ✓ Elaborar el modelo de diseño para lograr una especificación técnica del subsistema a implementar a partir de los requisitos identificados.
- ✓ Elaborar el modelo de implementación para proporcionar una visión general de lo implementado en el subsistema Penal.
- ✓ Realizar la implementación del subsistema Penal para lograr un producto funcional.
- ✓ Validar la solución desarrollada para evaluar su calidad.

Con el fin de cumplir el objetivo planteado y darle una solución eficiente al problema se definen las siguientes **tareas a cumplir:**

- ✓ Descripción y caracterización de las tecnologías y herramientas a utilizar en el desarrollo de la solución propuesta.
- ✓ Caracterización del estado del arte sobre buenas prácticas de programación.
- ✓ Creación de diagramas de clases, de secuencia, de componentes y el modelo de despliegue para la solución.
- ✓ Implementación de las clases del módulo Penal hasta la sentencia del proyecto Tribunales Populares Cubanos.
- ✓ Aplicación de casos de pruebas al sistema para validar las funcionalidades.

Los métodos científicos utilizados son:

- ✓ Métodos teóricos

  - Histórico-Lógico: usado en la investigación y análisis de la situación internacional en cuanto al uso de las Tecnologías de la Informática y las Comunicaciones en la rama jurídica.

  - Analítico-Sintético: usado en el estudio del funcionamiento de los procesos Penales actualmente, del trabajo en los Tribunales en general y para comprender y extraer los elementos más importantes que contribuyan al cumplimiento de los objetivos del trabajo.

El trabajo estará estructurado en 3 capítulos:

- ✓ **Capítulo 1:** Se aborda la fundamentación teórica para la realización del sistema e incluye el estado del arte de las técnicas de programación, las herramientas, metodologías y otras tecnologías existentes relacionadas con el desarrollo de la aplicación.
- ✓ **Capítulo 2:** En este capítulo se realiza una descripción de la solución de la investigación. Se aborda la especificación de la arquitectura del sistema, de los patrones de diseño que se utilizan, los estándares de codificación tenidos en cuenta para el proceso de desarrollo así como se describen los diagramas de clases del diseño

y los de secuencia de los casos de uso correspondientes y el modelo de implementación.

- ✓ **Capítulo 3:** Se realiza un análisis con el fin de valorar la calidad y la fiabilidad de la solución obtenida, para ello se aplican algunas métricas utilizadas para la medición de la calidad del diseño y la implementación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

La transformación de los procedimientos penales a una aplicación web lleva implícito una serie de conceptos y conocimientos que resultan de vital importancia dominar, es por ello que en el presente capítulo es necesario la realización de una fundamentación, respecto a las metodologías de desarrollo, herramientas y otras tecnologías que brindan las TIC y que resultan más que útiles, imprescindibles, para el desarrollo de la solución que se propone y de la aplicación como tal.

1.2 Aplicaciones Web

Las aplicaciones web están cada vez más presentes en las redes, han evolucionado desde simples sitios de exposición de información a grandes y complejas aplicaciones, haciéndose cada vez más dinámicas y ajustables a las necesidades de los usuarios.

Son aplicaciones cada vez más parecidas a las de escritorio, están diseñadas para funcionar en línea a través de un navegador web y residir en un servidor web. Esta centralización de la aplicación trae como ventajas que el acceso a ella no se limite a una determinada computadora, sino que pueda ser accedida desde varios puntos, en dependencia de las necesidades de los usuarios, proporcionando movilidad. Son por tanto, independientes del sistema operativo que utilicen los usuarios y salvo en el servidor web; no ocupan espacio ya que no necesitan ser descargadas ni instaladas, no se necesita tener un ordenador con grandes prestaciones para trabajar con estos sistemas, además de que facilita su administración, actualización y mantenimiento, es por esto que en la actualidad se han convertido en sistemas muy usados incluso para proyectos de gran envergadura, además resultan un medio económico y flexible para el acceso a la información y los servicios.

1.3 Paradigma de programación

Un paradigma de programación es una propuesta tecnológica que es adoptada por la comunidad de programadores y que se enfoca en resolver uno o varios problemas bien definidos y delimitados. (Campos, 2011)

En otras palabras no son más que modelos básicos de diseño y desarrollo de programas, formados por un grupo de patrones conceptuales que en conjunto modelan el proceso de diseño y determinan la estructura del programa.

1.3.1 Programación orientada a objetos (POO)

Este paradigma consiste de forma general en definir una serie de objetos e interacciones entre dichos objetos para de esta forma realizar el diseño de aplicaciones y programas a desarrollar.
La programación orientada a objetos es un conjunto completo de conceptos e ideas. Es una manera de pensar en el problema al que va dirigido un programa de ordenador y de afrontarlo de modo más intuitivo e incluso más productivo. (Archer, 2001)
La visión orientada a objetos obliga a reconsiderar ideas acerca de la programación, de lo que significa ponerla en práctica y de cómo debería estructurarse la información.
Este paradigma de programación o técnica como también se le llama, utiliza objetos e interacciones en el diseño de un sistema, está compuesto por una serie de elementos que lo identifican, como son:

- ✓ Clases: son un modelo que se utiliza para crear objetos que comparten un mismo comportamiento. Es una abstracción sobre un conjunto de objetos que comparten una estructura común y un comportamiento común.

- ✓ Métodos: son operaciones que pueden modificar el estado de un objeto o simplemente obtener datos sobre el mismo.

- ✓ Objetos: son entidades provistas de métodos o mensajes a los cuales responde (comportamiento), atributos con valores concretos (estado); y propiedades (identidad).

- ✓ Propiedades y atributos: son variables que contienen datos asociados a un objeto.

La POO posee ciertas características que la identifican de otros patrones de programación entre las más destacadas se encuentran:

- ✓ Abstracción: Define las características esenciales de un objeto.

- ✓ Polimorfismo: Es la capacidad que da a diferentes objetos, la posibilidad de contar con métodos, propiedades y atributos de igual nombre, sin que los de un objeto interfieran con el de otro.

- ✓ Herencia: Es una relación entre clases de manera jerárquica, una clase padre sobre otras clases hijas, donde estas últimas heredan de la clase padre las propiedades y métodos que se hayan especificado como heredables e incorporan las suyas propias.

Este paradigma de POO es el que se emplea en el desarrollo de la aplicación que busca informatizar los procesos de los TPC, debido a que fomenta la reutilización de código, al mismo tiempo que le permite a los programadores agilizar el desarrollo y producir un código de mayor calidad, logrando en la aplicación más estabilidad y que sea más fácil de actualizar y modificar.
El uso de este paradigma de la programación, además ayuda a facilitar la realización de sistemas complejos como es el caso de la aplicación para los TPC, así mismo permite un mejor y más fácil mantenimiento del software, agiliza y facilita el trabajo de equipo entre los desarrolladores.

1.4 Patrones de Diseño

Partiendo de que un patrón es un modelo a seguir para realizar algo y que en ellos está capturada la experiencia existente y probada para promover buenas prácticas. Se puede decir que los patrones de diseño no son más que: Soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan. (García, 2005)

Los patrones de diseños resultan bastante complicados de comprender, pero con ellos, una vez que se entiende su funcionamiento se pueden obtener diseños muchos más flexibles, modulares y reutilizables. Existen algunas denominaciones para clasificar los patrones, están los patrones de diseño GRASP, acrónimo de General Responsibility Assignment Software Patterns que describen los principios fundamentales de la asignación de responsabilidades a objetos y los denominados patrones GoF acrónimo de Gang of Four, estos últimos se clasifican en dependencia del propósito de cada uno de ellos: en creacionales, estructurales y de comportamiento.

Para que una solución sea considerada un patrón debe cumplir características como haber demostrado su efectividad en la solución del problema, debe ser reutilizable y aplicable a otros problemas similares. Cada patrón es adecuado para ser adaptado a un cierto tipo de problema en específico. Si bien el uso de patrones de diseño no es algo obligatorio, si resulta muy aconsejable debido a que evitan la búsqueda de soluciones a problemas ya conocidos y solucionados con anterioridad.

1.4.1 Patrón Inversión de Control

El patrón de diseño Inversión de Control ó IoC (siglas en inglés de Inversion Of Control) es un patrón del catálogo GoF e ideado para permitir un menor acoplamiento entre componentes de

una aplicación y fomentar así la reusabilidad de dichos componentes. Su utilización trae diversos beneficios como que reduce el acoplamiento entre objetos, fomenta el diseño basado en interfaces y permite reconfigurar una aplicación sin modificar el código.

El uso del patrón IoC es recomiendo en casos como cuando se necesita:

- ✓ Desacoplar las clases de sus dependencias de manera de que las mismas puedan ser reemplazadas o actualizadas con muy pocos o casi ningún cambio en el código fuente de sus clases.
- ✓ Escribir clases que dependan de clases cuyas implementaciones no son conocidas.
- ✓ Testear las clases aisladamente sin sus dependencias.
- ✓ Desea desacoplar sus clases de ser responsables de localizar y gestionar el tiempo de vida de sus dependencias.(Guillerón, 2009)

1.4.2 Patrón Singleton

El patrón de diseño Singleton (instancia única), clasificado como un patrón de tipo creacional es uno de los patrones más conocidos y utilizados siendo también uno de los más sencillos del catálogo del GoF, aunque generalmente un tanto difícil de entender. Este patrón está diseñado para restringir la creación de objetos pertenecientes a una clase, y entre sus propiedades se encuentran:

- ✓ Garantiza que una clase sólo tenga una instancia.
- ✓ Proporciona un punto de acceso global a dicha instancia.
- ✓ Trabaja mediante un método especial que se utiliza para instanciar el objeto.

1.4.3 Patrón de bajo acoplamiento

El acoplamiento es la medida de cuanto una clase está ligada con otras, a las que conoce y utiliza sus funcionalidades, donde una clase con bajo o débil acoplamiento indica que no depende de muchas otras. Mantener un bajo acoplamiento entre clases, más que un simple patrón es un principio que facilita la reutilización, el entendimiento de las clases y permite algo tan importante como que los cambios que se realizan en una clase no afecten a otras. Este patrón pertenece al catálogo de los patrones GRASP.

1.5 Metodologías de desarrollo de software

Para que un software con la envergadura del Sistema de Informatización de Tribunales se obtenga con la calidad requerida, debe llevarse a cabo una metodología que guíe todo el proceso de desarrollo del software.

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software. Es como un libro de recetas de cocina, en el que se van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Además detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla.(Asensio, 2011)

1.5.1 Proceso Unificado de Rational y Lenguaje Unificado de Modelado

El Proceso Unificado de Rational, conocido como RUP, acrónimo en inglés de Rational Unified Process es una metodología de desarrollo surgida gracias a la integración del trabajo de 3 importantes metodologistas, Ivar Jacobson, Grady Booch y James Rumbaugh, se caracteriza por encontrarse dividida en 4 fases de desarrollo las cuales son Inicio, Elaboración, Construcción y Transición. La fase de Inicio tiene como finalidad principal el alcanzar un acuerdo entre todos los interesados respecto a los objetivos del ciclo vital para el proyecto, esta fase es muy significativa fundamentalmente en el desarrollo de nuevas aplicaciones. El principal propósito de la fase de elaboración es el establecimiento de una línea base para la arquitectura del sistema y proporcionar una base para el diseño y la implementación de la fase de construcción. En la tercera fase llamada de construcción se busca completar el desarrollo del sistema y en la cuarta el propósito que se busca es el de garantizar que el software esté disponible para los usuarios.(Corp, IBM, 2006)

Los tres aspectos definitorios de RUP se resumen en tres pequeñas frases:

Dirigido por casos de uso: los casos de uso representan los requisitos funcionales, y todos los casos de uso juntos forman el modelo de casos de uso el cual describe la funcionalidad total del sistema y guían el proceso de desarrollo, ya sea el diseño, la implementación o las pruebas.

Centrado en la Arquitectura: donde dicha arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los

cimientos del sistema. RUP se desarrolla mediante iteraciones, comenzando por los CU relevantes desde el punto de vista de la arquitectura.

Iterativo e Incremental: RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. Cada iteración se realiza de forma planificada.

La metodología RUP tiene como objetivo el desarrollo correcto del software. Es una recopilación de prácticas de ingeniería de software que se están mejorando continuamente de forma regular para reflejar los cambios en las prácticas de la industria. La aplicación de esta metodología provee una serie de ventajas entre las que se encuentran.

- ✓ Mitigación temprana de posibles riesgos altos.
- ✓ Progreso visible en las primeras etapas.
- ✓ Temprana retroalimentación que se ajuste a las necesidades reales.
- ✓ Gestión de la complejidad.
- ✓ Conocimiento adquirido en una iteración puede aplicarse de iteración a iteración.

Debido a la dimensión y complejidad de la aplicación que se quiere desarrollar resulta necesario, mediante modelos, representar gráficamente el diseño de la solución. Los modelos permiten representar como es que se quiere el sistema, permite especificar su estructura y comportamiento. Para ello se usa el Lenguaje Unificado de Modelado ó UML, siglas en inglés de Unified Modeling Language, que no es más que un lenguaje gráfico para representar, visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software, además, facilita la creación de los diferentes diagramas y estandariza la manera en la que se realizan, su significado y la forma en la que se relacionan los mismos.

RUP concibe a UML como lenguaje de modelado, es una metodología con la que el equipo de desarrollo está bastante familiarizado, posee abundante documentación, es robusta y apropiada para proyectos de gran envergadura, genera un gran cúmulo de documentación, muy necesario si se tiene en cuenta que el equipo de trabajo varía y se debe tener documentado el trabajo realizado.

1.6 Lenguajes de programación

Un lenguaje de programación es aquel lenguaje empleado por un programador para dar al ordenador las instrucciones necesarias para la ejecución de un algoritmo determinado, constituyendo un programa fuente.(M.Isabel Gallego Fernández, 2000)

Son variados los lenguajes de programación para Web que nos brindan las nuevas tecnologías. Cada uno de ellos, con sus particularidades, ventajas y desventajas. Los lenguajes de programación web del lado del cliente y del servidor tienen características que los hacen complementarios más que adversarios y lo real es que cada tipo de programación tiene su lugar, y una mezcla de ellas es, frecuentemente, la mejor solución para el desarrollo de las aplicaciones.

1.6.1 PHP 5.2.5

Es el acrónimo de "PHP: Hypertext Preprocessor", es un lenguaje potente, popular, reconocido dentro de la comunidad de programadores y utilizado para la creación de sitios web. Este lenguaje permite aplicar ciertas técnicas de programación orientadas a objetos por lo que facilita la creación de aplicaciones complejas y resulta sumamente utilizado en el mundo entero y en la UCI en especial son numerosos los proyectos que lo usan.

Es un lenguaje interpretado del lado del servidor, especialmente pensado para desarrollos web, no necesita ser compilado para ejecutarse y genera así una página HTML, además puede ser incrustado en páginas HTML y tiene como objetivo fundamental permitir la creación por parte de los programadores de páginas dinámicas de una manera rápida y fácil.(PHP Documentation Group, 2011)

Hoy en día es el motor detrás de millones de aplicaciones web dinámicas. Su amplio conjunto de potentes características lo hacen un lenguaje ideal tanto para el desarrollo web de aplicaciones rápidas y sencillas como para la construcción de sistemas complejos.(David Sklar, Noviembre 2002)

Entre sus características más destacables se encuentran

- ✓ Resulta fácil de aprender.
- ✓ Posee soporte para la conexión a una gran cantidad de servidores de bases de datos, incluidos Postgres, MySQL, Oracle, entre otros.
- ✓ Es multiplataforma.
- ✓ Se caracteriza por ser un lenguaje rápido.

- ✓ Permite aplicar técnicas de programación orientada a objetos.
- ✓ Posee una amplia documentación tanto en el sitio oficial como en una multitud de sitios alternativos en Internet.
- ✓ Es libre, lo que lo convierte en una alternativa económica y de fácil acceso.

1.6.2 Javascript

Fue desarrollado por Netscape y es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con Javascript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. Es una marca registrada de la empresa Sun Microsystems(Pérez, Febrero 2008)

Resulta muy popular y utilizado en el desarrollo en el lado del cliente de las aplicaciones web, debido a que su aplicación facilita y mejora la interacción entre los usuarios y las aplicaciones, al mismo tiempo que resulta relativamente fácil de aprender y utilizar por los desarrolladores.

1.7 Object Relational Mapper(ORM)

Para acceder de forma efectiva a la base de datos desde un contexto orientado a objetos, es necesaria una interfaz que traduzca la lógica de los objetos a la lógica relacional, esta interfaz se llama Mapeo Objeto-Relacional (ORM por las siglas en inglés de Object Relational Mapper).(Fabien Potencier, 2008)

Un ORM permite convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, es decir, las tablas de la base de datos pasan a ser clases y los registros objetos.

La utilización de un ORM trae un conjunto de ventajas. El hecho de contar con un lenguaje de consultas propio, permite separar completamente el sistema del gestor de base de datos que se utilice, lo que permite garantizar que si en un momento determinado es necesario cambiar de gestor de base de datos, este cambio no implicará problemas de incompatibilidad entre el nuevo gestor y la aplicación. En materia de seguridad suelen implementar sistemas para evitar tipos de ataques como pueden ser las inyecciones de código. Como todo, también trae algunos inconvenientes que son necesarios afrontar como es el caso del tiempo extra necesario para aprender a utilizar estas herramientas que suelen ser bastante complejas de utilizar y lleva bastante tiempo aprender a sacar partido de todas las posibilidades que brinda, otro

inconveniente puede ser que las consultas pueden ser un poco más lentas al tener que transformar el lenguaje de las consultas.

1.7.1 Doctrine

Este ORM trabaja con PHP 5.2.3 o superior, se encuentra en la parte superior de una poderosa capa de abstracción de base de datos. La característica más importante es que brinda la posibilidad de escribir consultas de base de datos en un lenguaje propio orientado a objetos denominado DQL. Su uso proporciona a los desarrolladores una poderosa alternativa a SQL.(Sensiolabs, 2010)

Doctrine permite trabajar con un esquema de base de datos como si fuese un conjunto de objetos, y no de tablas y registros, permite generar de manera automática el conjunto de clases necesarias y que representan el modelo de la aplicación. Está dividido en dos capas principales, la DBAL del inglés Database Abstraction Layer y el ORM.

1.8 Marcos de trabajo (Framework)

En general los framework son soluciones completas que contemplan herramientas de apoyo a la construcción (ambiente de trabajo o desarrollo) y motores de ejecución (ambiente de ejecución). (SOA Agenda)

Desde el punto de vista del desarrollo de software, un marco de trabajo o framework es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado. (Alegsa)

El uso de marcos de trabajo permite:

- ✓ Facilitar el desarrollo de software.
- ✓ Evitar los detalles de bajo nivel, permitiendo concentrar más esfuerzo y tiempo en identificar los requerimientos de software.

1.8.1 Ext JS 2.2

Ext JS es una potente biblioteca javascript que permite construir aplicaciones web complejas. Fue originalmente construida como una extensión de la biblioteca YUI (Yahoo User Interface), y sale al mercado el 1 de abril de 2007 con la versión Ext JS 1.0.
Posee tres tipos de licencias, comercial, de código abierto; que implica liberar el proyecto con licencia GNU GPL V3; y licencia revendedor necesaria adquirirla cuando se desea realizar un

Framework o librería basada sobre Ext JS.
Ext JS hace que el desarrollo de las aplicaciones web sea simple debido a que:

- ✓ Proporciona una serie de componentes predefinidos tales como formularios, ventanas, redes, paneles divisibles en secciones, de fácil implementación y compatibles con los principales navegadores que existen en la actualidad.
- ✓ Permite que los usuarios y el navegador interactúen a través de eventos, tales como las pulsaciones de teclado, clics de ratón, y eventos de seguimiento en un navegador como cambiar el tamaño de la ventana, o los cambios tamaño de la fuente.
- ✓ Permite la comunicación con el servidor en segundo plano sin la necesidad de actualizar la página. Esto le permite solicitar o enviar datos desde o hacia su servidor web utilizando Ajax y procesar la información en tiempo real(Shea Frederick, 2008)

1.8.2 Zend Framework

Es un marco de trabajo que si bien algunas empresas como Microsoft y Google han contribuido con componentes o características importantes, es patrocinado principalmente por Zend Technologies.

Se caracteriza por ser un marco de trabajo de código abierto para desarrollar aplicaciones web y servicios web con PHP5. Zend Framework es una implementación que usa código 100% orientado a objetos. La estructura de los componentes de Zend Framework es algo único; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado.(Zend Technologies Inc, 2010)

Este marco de trabajo ofrece un gran rendimiento y una robusta implementación MVC (Modelo Vista Controlador).

1.8.3 Sauxe

Surge producto de la unión de Doctrine, Ext JS y Zend_Ext que a su vez no es más que una extensión de algunos componentes de Zend Framework, desarrollado por el Departamento de Tecnología y la UCID (Unidad de Compatibilización Integración y Desarrollo de Software para la Defensa)

Está desarrollado con PHP versión 5.2.4, en la capa de presentación utiliza HTML y Ext JS por la gran variedad de componentes que se pueden reutilizar, en la capa de negocio utiliza Zend_Ext, y en la capa de acceso a datos el lenguaje de consulta DQL que implementa Doctrine.

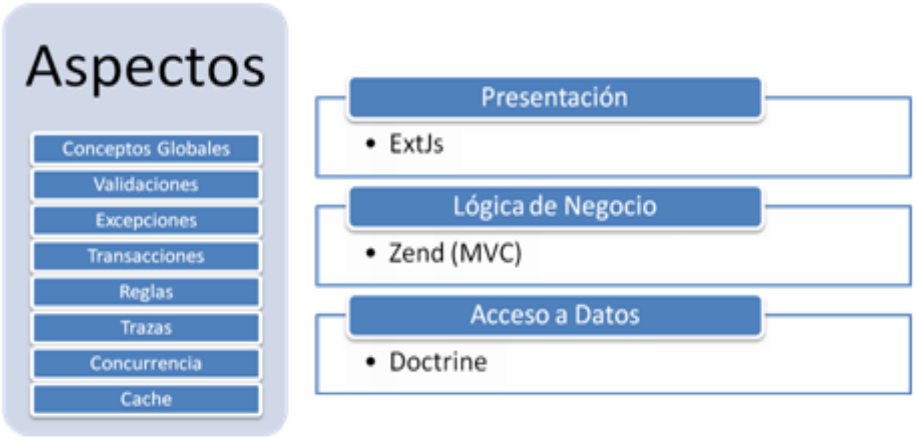

Figura 1: Arquitectura del marco de trabajo Sauxe.

A continuación se muestra una lista de algunas de las especificaciones o características más importantes respecto a los requerimientos funcionales de la arquitectura tecnológica de Sauxe.

- ✓ Cache: provee capacidad para configurar y gestionar de manera dinámica la cache del marco de trabajo.(Baryol)
- ✓ Integración: provee capacidad para configurar y gestionar de manera dinámica la integración de las aplicaciones o dominios de soluciones que se instancien o construyan con la misma. El soporte de integración permite de manera transparente para el usuario, consumir un servicios gestionado por la plataforma desde 3 alternativas específicas, una por vía de un servicio Web, otra por algún mecanismo de inversión de control que deberá proveer la arquitectura de integración del Marco de trabajo, y la otra desde algún servidor de comunicación o servicio de mensajería.(Baryol)
- ✓ ORM: provee un mecanismo de abstracción de la capa relacional de persistencia. EL ORM Doctrine Framework permite persistencia compuesta, carga perezosa, actualización y modificación sincronizada tanto en el modelo mapeado como en el modelo relacional.(Baryol)

- ✓ Mecanismo de gestión de concurrencia: provee un mecanismo de configuración y gestión dinámica de las características de concurrencia sobre entidades o dominios de la solución lógica que se modela.(Baryol)
- ✓ Mecanismo de administración de transacciones: provee un mecanismo de administración de transacciones. La solución arquitectónica permite configurar el esquema transaccional de un dominio específico de la solución que se construye.(Baryol)
- ✓ Administración del Pool de conexiones: provee un mecanismo o componente de administración y configuración del Pool de conexiones con los gestores más utilizados (PostgresSQL, Oracle, SQLServer, MySQL, etc.).(Baryol)
- ✓ Traza: provee un mecanismo de administración y configuración dinámica de trazas de la solución. La solución permite configurar la arquitectura de traza de un dominio de aplicación específico.(Baryol)
- ✓ Mecanismo de seguridad: provee un mecanismo de autentificación, autorización, administración de perfil y administración de conexiones.
- ✓ Tratamiento de excepciones: provee un mecanismo de gestión, configuración y administración de excepciones de manera dinámica y declarativa. El mismo se integra con el elemento responsable de las trazas.(Baryol)
- ✓ Validación: provee un mecanismo de gestión, configuración y administración de validaciones para sucesos del sistema.(Baryol)

El marco de trabajo Sauxe intenta contribuir con la independencia tecnológica del país permitiendo lograr con él una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo de software, además de que se acopla a las necesidades propias nacionales y resulta una opción novedosa y económica, que integra varias tecnologías libres y resulta la apuesta para el desarrollo de la aplicación.

1.9 Estilos arquitectónicos

Un estilo arquitectural se puede entender como un conjunto de principios que definen a alto nivel un aspecto de la aplicación. Un estilo arquitectural viene definido por un conjunto de componentes, un conjunto de conexiones entre dichos componentes y un conjunto de

restricciones sobre cómo se comunican dos componentes cualesquiera conectados. (Cesar de la Torre LLorente, 2010)

Dichos estilos arquitectónicos constituyen herramientas básicas a la hora de definir la estructura de un sistema software, también tienen una serie de directivas para organizar los componentes del mismo sistema con el objetivo de facilitar la tarea del diseño de tal aplicación. Indican también cómo se va a dividir el sistema en partes y de cómo será la interacción entre estas partes. Generalmente en los sistemas o aplicaciones se suelen utilizar varios estilos arquitecturales para aprovechar las ventajas de cada uno de ellos.

1.9.1 Arquitectura en Capas

El estilo arquitectural en capas se basa en una distribución jerárquica de los roles y las responsabilidades para proporcionar una división efectiva de los problemas a resolver. Los roles indican el tipo y la forma de interacción con otras capas y las responsabilidades la funcionalidad que implementan.(Cesar de la Torre LLorente, 2010)

Esta arquitectura se caracteriza porque al estar dividido el sistema en capas, las interacciones en la gran mayoría de los casos se realizan entre las capas vecinas, las funcionalidades de cada una de las capas van a estar separadas de las de otras de manera clara, donde cada capa solo contendrá funcionalidades relacionales con las tareas que le corresponde proporcionando un bajo acoplamiento. Se caracteriza también porque las diferentes capas no necesariamente deben estar en una misma máquina sino que pueden estar ubicadas en diferentes niveles físicos.

La arquitectura en capas provee de un buen número de ventajas, algunas de ellas se especifican a continuación.

- ✓ Permite mejorar la flexibilidad del sistema.
- ✓ Facilita el soporte y mantenimiento a las aplicaciones.
- ✓ Permite el desarrollo en paralelo.
- ✓ Permite llevar a cabo un desarrollo en varios niveles.
- ✓ Facilita la reutilización de código y la estandarización.
- ✓ Permite crear aplicaciones más robustas debido al encapsulamiento.

- ✓ Contención de cambios de una a pocas capas.

1.9.2 Modelo Vista Controlador

El Modelo Vista Controlador (MVC) es un patrón arquitectónico cuya principal característica es dividir una aplicación en tres capas identificables y con funcionalidades bien definidas:

- ✓ Modelo: representa la lógica del negocio, es el encargado de acceder de forma directa a los datos actuando como "intermediario" entre la base de datos y el controlador.
- ✓ Vista: es la capa encargada de mostrar la información al usuario, es con la que los usuarios interactúan directamente.
- ✓ Controlador: es el intermediario entre la vista y el modelo, es quien controla las interacciones del usuario solicitando los datos al modelo y entregándolos a la vista para que ésta lo muestre al usuario

1.10 Herramientas utilizadas

A continuación se describen las herramientas utilizadas en el desarrollo del subsistema que forma parte del Sistema de Informatización de Tribunales, las cuales se estudiaron para la elaboración de este trabajo de diploma.

1.10.1 Entorno de desarrollo integrado. Netbeans 6.9.

Un entorno de desarrollo integrado ó IDE (siglas en inglés de Integrated Development Environment) no es más que una aplicación que integra varias aplicaciones de desarrollo, aunque entre uno y otros entornos de desarrollo pueden variar algunos de sus componentes, generalmente poseen un compilador, un constructor de interfaces gráficas, un editor de código, entre otras aplicaciones que los diferencian de los editores básicos. Todo esto pues le brinda a los desarrolladores un entorno de desarrollo amigable en el que trabajar, y realizar sus proyectos, existen un buen número de entornos muy profesionales y potentes para el desarrollo de aplicaciones usando el lenguaje de programación PHP, tales como Eclipse, PHP Designer, Zend Studio, NetBeans y muchos otros pero este último en su versión 6.9 fue el que se definió para el desarrollo de la aplicación.

NetBeans IDE es un entorno de desarrollo integrado modular y basado en estándares, escrito con el lenguaje de programación Java.(Oracle Corporation)

Este IDE resulta bastante reconocido entre la comunidad de desarrolladores, y en la UCI resulta sumamente utilizado, está disponible tanto para las plataformas Linux como para Windows y Mac, es un software libre, gratuito, sin restricciones de uso y con una interfaz muy amigable para los desarrolladores que les permite crear con relativa rapidez sus aplicaciones. Es compatible con un gran número de tecnologías incluidas PHP y Subversion, funciona en sistemas que cuenten la máquina virtual Java y requiere para su instalación el Java SE Development Kit (JDK) 6.0 o superior. Para la implementación de la solución se utilizó el NetBeans en su versión 6.9 para Linux.

1.10.2 Mozilla Firefox

Navegador web gratuito creado por Mozilla, es el segundo navegador más utilizado después del Internet Explorer de Microsoft. Posee una serie de características que lo convierten en un navegador moderno, navegación por pestañas que permite tener abiertas varias páginas al mismo tiempo en una sola ventana, posee un gestor de descargas facilitando las tareas de descargas de archivos, una barra de direcciones inteligente y como la característica más interesante la posibilidad de añadirle funcionalidades mediante complementos, entre estos últimos existen algunos que resultan muy útiles y casi imprescindible en el proceso de desarrollo de aplicaciones web como es el caso del Firebug.

1.10.3 Firebug 1.9.1

Firebug es un complemento muy popular en toda la comunidad de desarrolladores web, el mismo se integra al Mozilla Firefox, diseñado especialmente para desarrolladores web. Es un paquete de utilidades con el que se puede analizar, editar, monitorizar y depurar el código fuente, CSS, HTML y Javascript de una página web en tiempo real. A este complemento también resulta posible añadirle una serie de extensiones que le añaden funcionalidades extras que también pueden resultar útiles durante la implementación. El uso de esta herramienta resulta sumamente útil para los desarrolladores y ahorra cantidades de tiempo considerables si atendemos a que con ella podemos realizar análisis más ágiles sobre lo que está ocurriendo durante la ejecución de nuestra aplicación, así mismo nos ayuda a la detección de orígenes de errores que sin el uso del Firebug resultarían mucho más complicados de identificar.

1.10.4 Herramienta para el modelado Visual Paradigm for UML 6.4.

Las herramientas CASE (Ingeniería de Software Asistida por Ordenador) son las aplicaciones que participan en el desarrollo de programas informáticos, facilitando la planificación, el análisis

y diseño, y hasta la generación del código fuente de los programas así como su documentación. El uso de estas herramientas tiene como base la búsqueda de un aumento de la productividad en el desarrollo de software reduciendo el costo de las mismas tanto en términos de tiempo como de dinero.

Visual Paradigm para UML es una herramienta diseñada para Ingenieros de Software, Analistas de Sistemas, Arquitectos de Sistemas y otros usuarios que estén interesados en el diseño de software orientado a objetos. Esta herramienta provee maneras de crear los diferentes diagramas de UML de forma muy intuitiva con solo realizar la operación de arrastrar y soltar.(Visual Paradigm, 2007)

Esta herramienta CASE resulta ser de alta calidad, con buenas prestaciones y muy profesional, también es bastante fácil de usar, soporta el ciclo de vida completo del software, posee soporte para UML 2.1, además de que provee una gran interoperabilidad entre sus modelos, permitiendo la exportación e importación de diagramas entre modelos lo cual ahorra considerables cantidades de tiempo, en fin resulta una alternativa viable para el desarrollo sobre plataformas Linux como lo es Ubuntu, que es uno de los requisitos para el desarrollo del proyecto y por eso es que forma parte del conjunto de herramientas utilizadas.

1.11 Sistema gestor de Base de Datos

Un gestor de base de datos o sistema de gestión de base de datos es un software que permite el almacenamiento, manipulación y consulta de la información de una base de datos. Aunque existen diferentes tipos de gestores de bases de datos, el modelo relacional es el utilizado por casi todos los gestores de bases de datos para computadoras personales.

1.11.1 PostgreSQL 8.4

Es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD (Berkeley Software Distribution) y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales.

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.(2010)

Sus características la hacen una de las bases de datos más potentes, robustas y usadas del mercado. Entre ellas destacan algunas propiedades técnicas y otras generales que lo convierten en un gestor muy popular:

- ✓ Es una base de datos 100% ACID(Atomicidad, Consistencia, Aislamiento, Durabilidad) estas propiedades se describen a continuación:
    - Atomicidad: propiedad que asegura que una operación se efectúe ó no, en caso de un fallo no se puede quedar a medias.
    - Consistencia: propiedad que se refiere a que solo se ejecutan las operaciones que se puedan concluir.
    - Aislamiento: es la propiedad que asegura que una operación no puede afectar a otras.
    - Durabilidad: es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.
- ✓ Funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez.
- ✓ Soporte de todas las características de una base de datos profesional.
- ✓ Posee numerosos tipos de datos y brinda la posibilidad de definir nuevos tipos.
- ✓ Permite acceso encriptado vía SSL.
- ✓ Distribuido bajo licencia BSD: que es licencia de software libre permisiva y que tiene como característica distintiva que permite el uso del código fuente en software no libre.
- ✓ Es multiplataforma, funciona en los principales Sistemas Operativos del mercado.
- ✓ Posee una documentación completa y organizada, pública y libre.
- ✓ Soporte nativo para los lenguajes más populares del medio incluido PHP.

1.12 Control de versiones

El control de versiones involucra procedimientos y herramientas para gestionar los cambios en los elementos creados durante el ciclo de vida del software. (Control de versiones)

Se refiere a la gestión de los cambios que se realizan sobre los elementos de algún producto o sobre la configuración del mismo. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas. El principal objetivo es permitir editar de forma colaborativa y compartir información. Aunque un sistema de control de versiones puede realizarse de forma manual, es muy aconsejable disponer de herramientas que faciliten esta gestión.

Los sistemas de control de versiones son herramientas software que permiten la automatización de tareas comunes sobre archivos (guardar, recuperar, registrar, borrar, identificar, mezclar) manteniendo una correcta gestión sobre las versiones de la información almacenada.(Guía de Subversion)

1.12.1 Subversion

Es un popular sistema de control de versiones centralizado, es libre y empleado en la administración de ficheros y directorios utilizados generalmente en el desarrollo de programas informáticos, permite además la gestión de los cambios sobre los mismos, permite que varios usuarios clientes saquen copias del proyecto al mismo tiempo, posibilita examinar el registro histórico de cambios, deshacer los cambios en un momento dado, comparar diferentes versiones de los archivos, unir cambios realizados por diferentes usuarios en un mismo fichero así como muchas otras características propias de un sistema de control de versiones.

1.12.2 RapidSVN 0.12

El RapidSVN es un cliente gráfico de Subversión, escrito en C++, es una aplicación multiplataforma, posee una interfaz fácil de utilizar, simple para los usuarios principiantes pero bastante flexible para los usuarios más experimentados con el uso de Subversion, se distribuye bajo licencia GNU GPL, resultó la opción elegida por el equipo de arquitectura pues resulta viable su utilización sobre Linux que es donde se desarrolla la aplicación.

1.13 Servidor de Aplicaciones

Software que proporciona aplicaciones a los equipos o dispositivos cliente, por lo general a través de Internet y utilizando el protocolo http y maneja la mayoría de las transacciones relacionadas con la lógica y el acceso a los datos de la aplicación.

1.13.1 Servidor Apache 2.0

Apache es un servidor web de código libre, flexible, rápido, eficiente y continuamente actualizado cuya implementación se realiza de forma colaborativa, con altas prestaciones y funcionalidades equivalentes a las de los servidores comerciales. Permite crear un servidor http en un ordenador de una forma relativamente rápida y sencilla.

Está estructurado en módulos. La configuración de cada módulo se hace mediante la configuración de las directivas que están contenidas dentro del módulo. Los módulos del Apache se pueden clasificar en tres categorías:

- ✓ Módulos Base: Módulo con las funciones básicas del Apache
- ✓ Módulos Multiproceso: son los responsables de la unión con los puertos de la máquina, acepando las peticiones y enviando a los hijos a atender a las peticiones
- ✓ Módulos Adicionales: Cualquier otro módulo que le añada una funcionalidad al servidor. (Desarrolloweb.com)

Entre sus principales características destacan:

- ✓ Multiplataforma
- ✓ Es un servidor de web conforme al protocolo HTTP/1.1
- ✓ Modular: Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con la API de programación de módulos para el desarrollo de módulos específicos.
- ✓ Incentiva la realimentación de los usuarios, obteniendo nuevas ideas, informes de fallos y parches para la solución de los mismos.
- ✓ Se desarrolla de forma abierta
- ✓ Extensible: gracias a ser modular se han desarrollado diversas extensiones entre las que destaca PHP, un lenguaje de programación del lado del servidor.(Félix, 2000)

1.14 Pruebas de software

Las pruebas de software se realizan con el objetivo de determinar el grado de calidad de un programa informático así como el cumplimiento de los requerimientos que debe cumplir. Consisten en la ejecución de un programa con la intención de descubrir un error, donde un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces y una prueba tiene éxito si descubre un error no detectado hasta entonces.

1.14.1 Pruebas de caja blanca

La prueba de la caja blanca no es más que un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivar los casos de prueba e intentan garantizar que se ejecuten todos los caminos independientes al menos en una ocasión, que se utilizan las decisiones en su parte verdadera y su parte falsa, que se ejecutan los ciclos en sus límites y que se utilizan todas las estructuras de datos internas existentes.

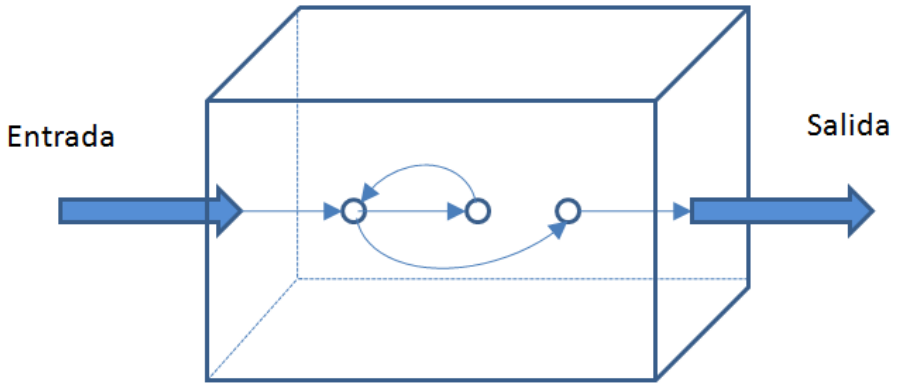

Figura 2: Prueba de caja blanca.

Consiste en la realización de un examen minucioso de los detalles procedimentales, realizando un seguimiento del código fuente, comprobando cada uno de los caminos lógicos del programa, examinando los ciclos, condiciones, y el estado del programa en varios puntos.

En este tipo de pruebas se utilizan diferentes métodos con el fin de analizar el código, como son:

- ✓ Pruebas de camino básico.

- ✓ Pruebas de condición.

- ✓ Pruebas de bucles.

Para la validación de la solución se empleó el método del camino básico que consiste en obtener el nivel de complejidad de un código y utilizar esta medida como guía para definir los caminos básicos y diseños de casos de pruebas que garanticen la ejecución de cada uno al menos una vez.

1.15 Métricas de calidad del software

Las métricas de software son técnicas que nos ayudan a entender tanto el proceso técnico que se utiliza para desarrollar un producto, como el propio producto(Universidad de Guadalajara). Resultan un buen medio para entender, monitorizar, controlar, predecir y probar el desarrollo software.

1.15.1 Tamaño Operacional de las Clases (TOC)

Esta métrica está dada por el número de operaciones asignados a una clase, donde un resultado alto de su aplicación indica que la clase posee bastante responsabilidad, implicando un bajo nivel de reutilización de la clase y complicará la implementación a continuación se especifican y detallan los atributos de calidad que evalúa esta métrica.

- ✓ **Responsabilidad:** Un incremento del TOC está ligado a un aumento de la responsabilidad asignada a una clase.

- ✓ **Complejidad de implementación:** Un aumento del TOC indica un incremento de la complejidad de implementación de una clase.

- ✓ **Reutilización:** Un incremento del TOC implica una disminución del grado de reutilización de una clase.

1.15.2 Relaciones entre Clases (RC)

Esta métrica toma como punto de análisis el número de relaciones de uso de una clase con otras, tiene en cuenta la evaluación de atributos de calidad tales como el acoplamiento, la complejidad de mantenimiento, la reutilización y la cantidad de pruebas, un resultado de la evaluación de la métrica RC indica que alto será también el grado de acoplamiento, la complejidad de mantenimiento y la cantidad de pruebas, mientras que será inversamente

proporcional a la reusabilidad la cual se verá disminuida. Seguidamente se detalla la relación entre los atributos que evalúa y el resultado de dicha métrica.

- ✓ **Complejidad de mantenimiento:** Un aumento del RC está ligado a un aumento de la complejidad del mantenimiento de la clase.
- ✓ **Cantidad de pruebas: Un aumento del RC** implica un aumento de la cantidad de pruebas de unidad necesarias para probar una clase.
- ✓ **Acoplamiento:** Un incremento del RC indica un aumento del acoplamiento de las clases.
- ✓ **Reutilización:** Un aumento del RC implica una disminución en el grado de reutilización de la clase.

1.16 Conclusiones parciales

Con la culminación del capítulo se logró el cumplimiento de los objetivos propuestos. Luego de la investigación se lograron una serie de conocimientos importantes para la continuidad del trabajo, se fundamentó y justificó sobre la metodología de desarrollo, herramientas y otras tecnologías necesarias para el desarrollo de la solución.

2 CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

2.1 Introducción

En el presente capítulo se describe la solución de la investigación. Se aborda la especificación de la arquitectura del sistema, sus ventajas, los patrones de diseño que se aplican, los estándares de codificación necesarios y tenidos en cuenta para el proceso de desarrollo así como, la especificación los artefactos necesarios para la informatización de los requisitos previos, los diagramas de clases del diseño y los de secuencia de los casos de uso correspondientes y el modelo de implementación.

2.2 Arquitectura del Sistema

Teniendo en cuenta las características de lo que se debe implementar, en la aplicación se aplica la arquitectura en capas, una arquitectura que facilita la estandarización, la reutilización de código, permite que los cambios en una de sus capas no sean muy traumáticos en el resto de las capas. Algo que resulta muy importante debido a la dimensión de la aplicación en cuestión, es que permite el desarrollo en paralelo, es decir, que se puede ir trabajando en diferentes capas por separado al mismo tiempo, lo que permite agilizar el desarrollo, también puede llevarse a cabo la implementación en varios niveles, dígase niveles a las formas físicas en las que se distribuyen las capas, en otras palabras se puede trabajar en diferentes máquinas. Además, aprovechando sus beneficios se aplica el patrón arquitectónico modelo vista controlador, muy utilizado en el desarrollo de aplicaciones web.

Como muestra la siguiente figura, la arquitectura en cuestión está formada por la capa de presentación; que es la que los usuarios finales ven; la capa de negocio y la capa de acceso a datos que es la encargada de comunicar la capa de negocio con los orígenes de datos. A continuación se describen las diferentes capas.

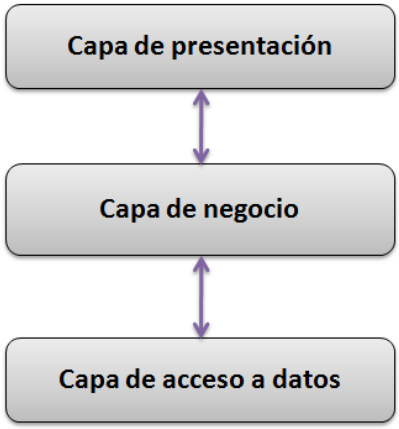

Figura 3: Arquitectura en capas de la aplicación

Capa de Presentación

Es la capa con la que los usuarios finales interactúan directamente, en esta se encuentran las interfaces que muestran la información a los usuarios, los archivos ".phtml" que tienen dependencia de los ficheros Javascript donde se implementa íntegramente el código de las interfaces utilizando la librería Ext JS, aprovechando sus bondades y todo el conjunto de componentes predefinidos que posee, en esta capa también se encuentran las hojas de estilo en cascada o archivos CSS como también se les conoce, así como otros ficheros cuya existencia responde a las necesidades de esta capa.

Capa de Negocio

Se comunica directamente con la capa de presentación y es la encargada de la captura de las solicitudes y peticiones que realiza el usuario desde la vista, así como de enviar los resultados de las operaciones a las interfaces. Resulta el intermediario entre las capas de presentación y de acceso a datos, con esta última interactúa para realizar diferentes operaciones con los datos que se almacenan en el gestor.

Capa de acceso a datos.

La función fundamental de esta capa es la de servir de conector entre la capa de negocio y el gestor de base de datos que en este caso es PostgreSQL. Buscando la optimización, en esta capa está presente el marco de trabajo de persistencia de datos Doctrine el cual permite mediante su lenguaje propio de consultas DQL la independencia de la aplicación respecto al gestor de base de datos.

2.3 Descripción de los casos de uso a implementar

A partir de la especificación de requisitos y el modelo del sistema generados durante la captura de requisitos realizada previamente por el equipo de analistas, resultó que las funcionalidades a implementar sean las siguientes:

N° RF	Caso de Uso	Descripción
RF.06	Registrar testigo.	El sistema permitirá registrar los datos del o los testigos propuestos por el Fiscal.

RF.02	Registrar piezas de convicción.	El sistema permitirá registrar las piezas de convicción en caso de que existan.
RF. 22 **RF. 23** **RF. 24**	Aperturar expediente en fase preparatoria.	Consiste en permitirle al juez ponente gestionar el auto de apertura.
RF.87	Disponer sobre bienes.	El sistema permitirá disponer sobre el destino de los bienes ocupados, muestra la relación de bienes ocupados de una causa con los datos respectivos.
RF.82	Registrar bienes ocupados.	El sistema permitirá registrar los bienes ocupados en caso de que existan.
RF. 41	Crear acta de prestación de fianza.	Consiste en que la aplicación debe permitirle a la secretaria crear el acta de prestación de fianza.
RF. 37 **RF. 38** **RF. 91**	Registrar presentación de acusado.	El sistema debe permitir registrar la presentación del o los acusados que se encuentren citados para asistir al tribunal.
RF. 72	Tramitar orden de arresto.	El sistema debe permitir tramitar la orden de arresto.

2.4 Modelos de diseño

El Modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Además, el modelo de diseño sirve de abstracción de la implementación del

sistema y es, de ese modo, utilizada como una entrada fundamental de las actividades de implementación. (I.Jacobson, 2000)

2.4.1 Diagrama de clases del diseño

El Diagrama de clases se realiza para tener en cuenta los detalles concretos de la implementación del sistema. En él, se describe la estructura de clases del sistema, las funcionalidades, los atributos correspondientes a cada una de las clases, así como las relaciones entre ellas, generalmente de un solo caso de uso.

La siguiente tabla muestra un resumen de la descripción del caso de uso registrar testigo previamente realizado por el equipo de analistas y clasificado como crítico en el desarrollo del módulo Penal, la cual ayuda a la comprensión del diagrama de clases del diseño correspondiente a este caso de uso, la descripción se encuentra descrita íntegramente en conjunto con las descripciones del resto de los casos de uso en el Modelo del Sistema.

Caso de Uso:	Registrar testigo.
Actores:	Fiscal.
Resumen:	El caso de uso se inicia cuando el Fiscal necesita registrar un testigo propuesto por él, en caso de que solicite una prueba testifical. Consiste en que el Fiscal selecciona la opción de registrar testigos y llena los campos requeridos. El caso de uso termina con información del testigo registrado.
Precondiciones:	
Referencias	RF.06.
Prioridad	Crítico.

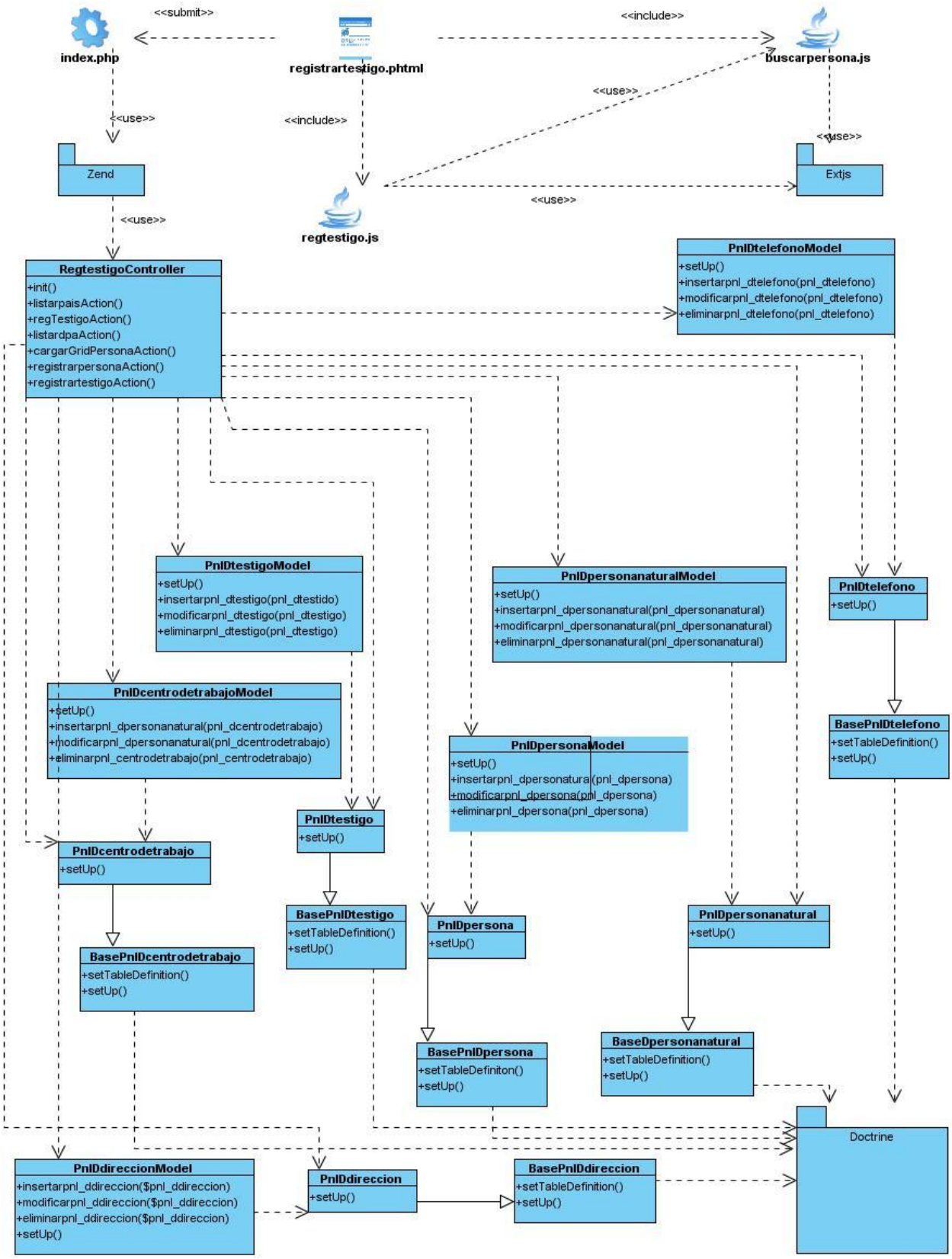

Figura 4: Diagrama de clases del diseño del caso de uso Registrar testigo.

En el diagrama de clases anterior se muestran los archivos correspondientes e involucrados en el caso uso Registrar testigo, entre los principales que se representan se encuentran:

- ✓ El archivo “regtestigo.phtml”, el cual hace referencia a los ficheros Javascript que contienen la implementación de los componentes ofrecidos por la librería Ext JS para la creación de la vista.
- ✓ La clase controladora “RegtestigoController”, encargada de las funcionalidades que debe realizar el caso de uso.
- ✓ Las clases “Base” contienen los atributos de las tablas de la base de datos de las que fueron generadas.
- ✓ Las clases “Model” son las encargadas de insertar, modificar y eliminar etc.

2.4.2 Diagramas de secuencia

Un Diagrama de secuencia es una forma de diagrama de interacción que muestra los objetos como líneas de vida a lo largo de la página y con sus interacciones en el tiempo representadas como mensajes dibujados como flechas desde la línea de vida origen hasta la línea de vida destino. (Sparx Systems Pty Ltd)

Describen la secuencia de acciones en un caso de uso que comienza cuando el actor invoca el caso de uso mediante el envío de algún tipo de mensaje al sistema. Generalmente en un diagrama de secuencia se capta el comportamiento de un solo caso de uso y se realizan con el fin de definir las acciones que se pueden realizar en la aplicación en cuestión y mostrar la forma en la que interactúan los objetos. El diagrama muestra un determinado número de objetos y los mensajes que se pasan entre estos objetos dentro del caso de uso.

El siguiente diagrama corresponde al caso de uso Registrar Testigo, y representa el flujo normal de eventos.

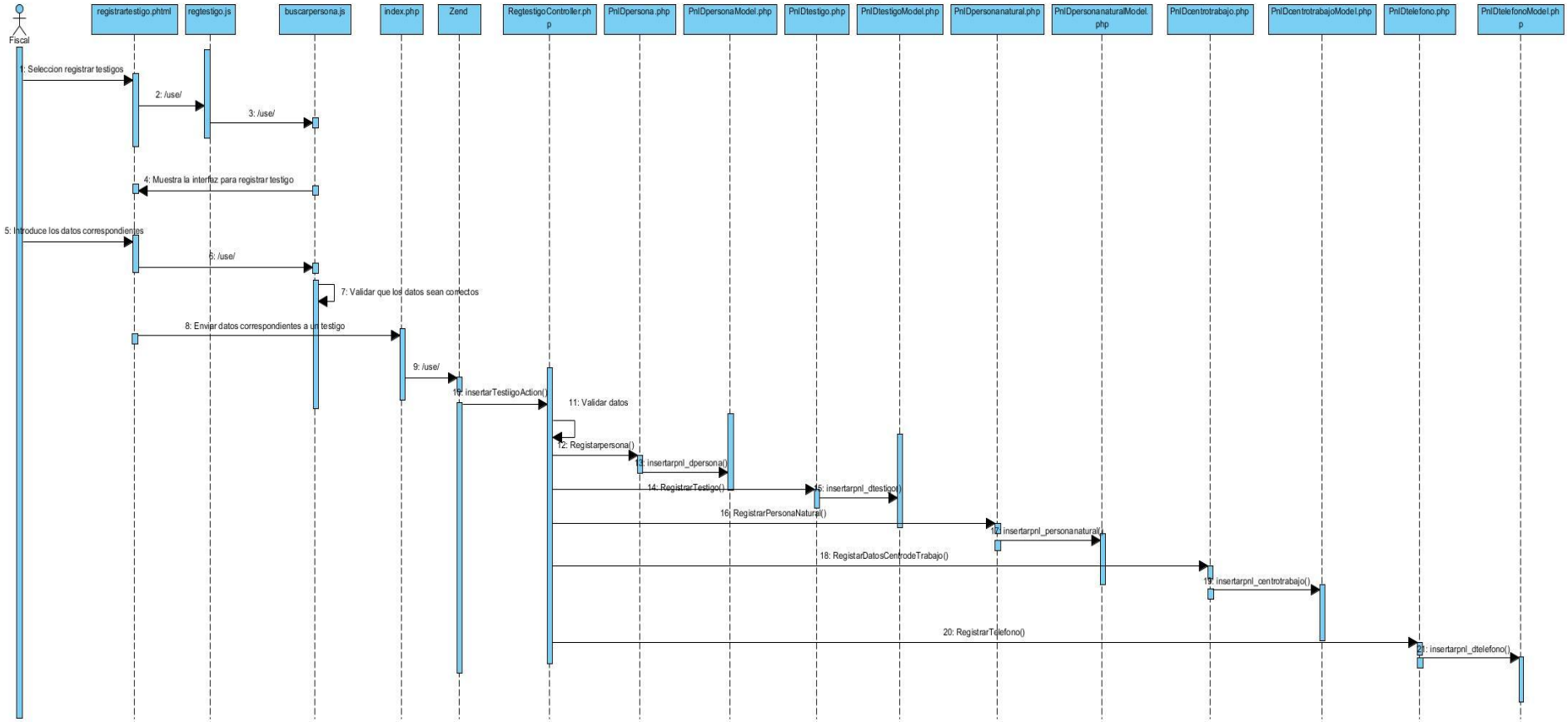

Figura 5: Diagrama de secuencia del flujo normal de eventos del caso de uso Registrar Testigo.

2.5 Modelo de Implementación

El Modelo de implementación es de gran utilidad para la implementación del sistema. Organiza el trabajo para los desarrolladores y describe cómo se implementan en términos de componentes los elementos del modelo de diseño. (I.Jacobson, 2000)

2.5.1 Diagrama de componentes

Los Diagramas de componentes forman parte del modelo de implementación y describen los elementos físicos del sistema y sus relaciones, donde los componentes representan todos los tipos de elementos software que entran en la fabricación de la aplicación y que pueden ser archivos, paquetes, librerías, etc. Las relaciones se utilizan para indicar que un componente utiliza los servicios de otro componente. Con el fin de simplificar los diagramas, los diferentes componentes suelen agruparse en paquetes denominados subsistemas, los cuales a su vez pueden contener otros subsistemas y de manera general organizan la vista de implementación de un sistema.

Estos diagramas se utilizan para mostrar la estructura a alto nivel del modelo de implementación en términos de subsistemas de implementación y las relaciones entre los componentes. (I.Jacobson, 2000)

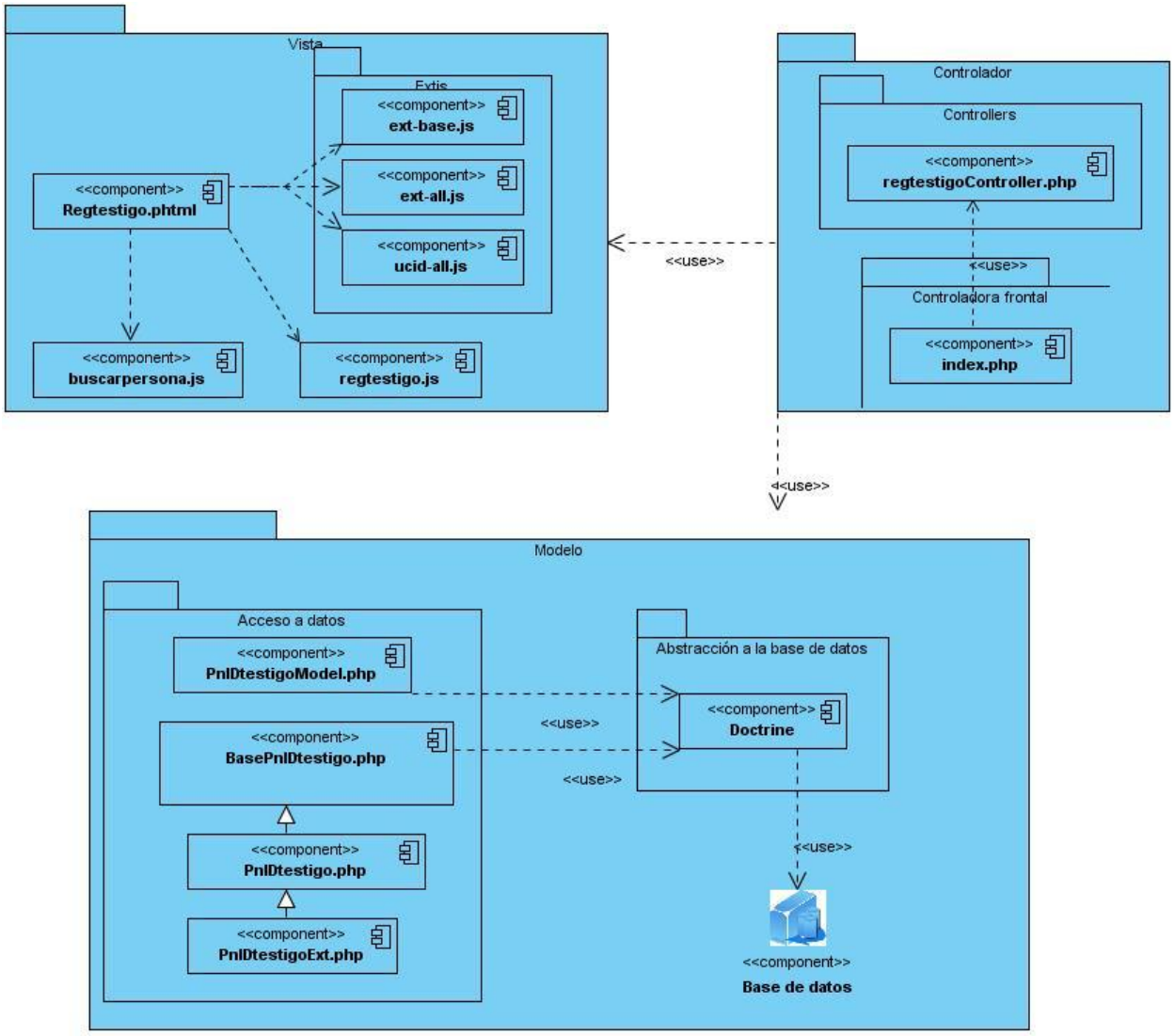

Figura 6: Diagrama de componentes para el caso de uso Registrar Testigo.

2.5.2 Diagrama de despliegue

El Diagrama de despliegue describe como una aplicación se despliega a través de una infraestructura, muestra las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos, puede describir las diferentes configuraciones de red. La funcionalidad de un nodo va a estar dada por los componentes que estarán distribuidos en él. Cada nodo es un elemento físico que representa un recurso computacional y que se utiliza para modelar la topología del hardware sobre el que se ejecuta el sistema y es donde se ejecutan los componentes los cuales no son más que la representación del empaquetamiento físico de los elementos lógicos del sistema. Las

relaciones entre los nodos representan los medios de comunicación entre ellos, tales como TCP/IP, HTTP, USB, etc.

El siguiente diagrama de despliegue tiene la intención de describir cómo va a ser desplegada la aplicación para cada una de las diferentes instancias de los Tribunales Populares Cubanos, los municipales, los provinciales y el tribunal supremo. De acuerdo a los requisitos no funcionales de hardware las estaciones de trabajo de los clientes deben poseer al menos 512 MB de RAM y un procesador de 1.90GHZ y el servidor de base de datos debe disponer de al menos 1 GB de RAM y un procesador de 1.90GHZ.

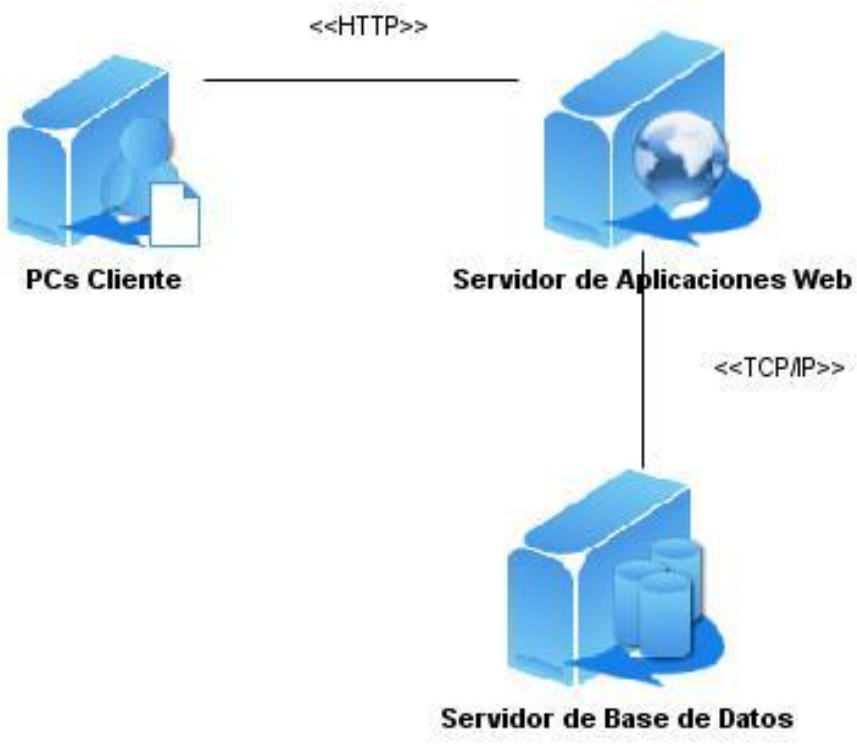

Figura 7: Diagrama de despliegue de la solución.

2.6 Patrones utilizados

2.6.1 Singleton

El marco de trabajo Sauxe define el uso del patrón Singleton que resulta muy útil en el desarrollo de los diferentes subsistemas del proyecto en cuestión. En la imagen siguiente se representa la implementación de este patrón en la clase ZendExt_IoC, donde se puede apreciar dentro de los elementos característicos fundamentales que el constructor se declara privado para no permitir que se creen instancias de él, además esta implementada la función getInstance() mediante la cual se puede obtener la instancia de la clase, ya que esta no puede ser instanciada directamente lo que garantiza que es un singleton. Se utiliza con el fin de que cuando varios clientes distintos precisan referenciar al mismo elemento y se quiere asegurar de que no haya más de una instancia de ese elemento, disminuyendo consigo el consumo de recursos.

```
class ZendExt_IoC implements ZendExt_Aspect_ISinglenton {

        const IOC_SIMULATE_RESULT = false;
        private $module;
        protected $idXMLIoC;
        protected $iocTrace = false;
        protected $iocExceptionTrace = false;

        private function __construct() {
                $this->idXMLIoC = 'ioc';
                $aspectxml = ZendExt_FastResponse::getXML('aspect');
                $this->iocTrace = (string) $aspectxml->beginTraceIoC['active'];
                $this->iocExceptionTrace = (string) $aspectxml->failedTraceIoC['active'];
        }

        static public function getInstance() {
                static $instance;
                if (!isset($instance))
                        $instance = new self();
                return $instance;
        }
```

Figura 8: Implementación del patrón singleton que se encuentra en la clase ZendExt_IoC.

```
$integrator = ZendExt_IoC::getInstance();
$fechahoy = $integrator->moduloComun->ObtenerFechaHoy();
```

Figura 9: Ejemplo de la solicitud de la instanciación de la clase ZendExt_IoC.

2.6.2 Inversión de Control

La aplicación del patrón de Inversión de Control en el marco del trabajo Sauxe resulta sumamente útil en el desarrollo del proyecto, debido a que en cada uno de los módulos existen esquemas de base de datos diferentes y en ocasiones es necesario obtener datos de otros módulos, o realizar operaciones comunes para varios módulos. En estos casos es que se utiliza esta técnica para facilitar la interoperabilidad o la comunicación entre los módulos, creando y prestando servicios que son consumidos sin necesidad de conocer el código, simplemente saber su funcionamiento. Dentro de la jerarquía de carpetas del proyecto en "TPC/apps/penal/services" se encuentran el conjunto de clases que implementan los servicios del módulo penal y en "TPC/apps/penal/común/recursos/xml" se encuentran un conjunto de ficheros que entre ellos el "ioc-general.xml" es el encargado de la configuración de servicios para la publicación de los mismos al resto de los módulos, en cada uno de los módulos existe una carpeta similar.

```
class PlantillaService {

        function ListarEscritosResoluciones($tipo)
        {
            if($tipo == 'escrito')
            {
               $temp = NescritoExt::ListarEscritos();
            }
            else
            {
                $temp = NresolucionjudicialExt::ListarResoluciones();
            }
            return $temp;
        }
}
```

Figura 10: Implementación del servicio ListarEscritosResoluciones del módulo penal.

```
<ListarTipo reference="">
   <inyector clase="PlantillaService" metodo="ListarEscritosResoluciones" />
   <prototipo>
       <parametro nombre="tipo" tipo="string" />
       <resultado tipo="array" />
   </prototipo>
</ListarTipo>
```

Figura 11: Configuración del servicio ListarEscritosResoluciones en el archivo ioc-general.xml del módulo penal.

```
$integrator = ZendExt_IoC::getInstance();
$fecha = $integrator->moduloComun->ObtenerFechaHoy();
```

Figura 12: Ejemplo del consumo de un servicio.

2.6.3 Patrón de bajo acoplamiento

Este patrón resulta de gran utilidad en el desarrollo de la solución ya que con el trabajo con el modelo del marco de trabajo Sauxe, existen clases de abstracción de los datos que genera doctrine que solo dependen de las clases “Base”, las cuales son sumamente reutilizables y muy utilizadas en la solución lo que evidencia los beneficios de la presencia de este patrón.

```
<?php
class PnlDbienocupado extends BasePnlDbienocupado
{
   public function setUp()
   {
                parent::setUp();

        $this->hasOne('PnlDexpedientepreparatorio', array('local'
   }

}
?>
```

Figura 13: Ejemplo de clase con bajo acoplamiento.

2.6.4 Modelo Vista Controlador (MVC)

El patrón arquitectónico Modelo Vista Controlador permite el desacople entre la vista y el modelo, de forma que las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos, favoreciendo la reusabilidad, haciendo más sencillo el mantenimiento y que la realización de cambios en una de las capas no implique afectar la otra. Dentro de la

jerarquía de paquetes del marco de trabajo Sauxe existen paquetes bien identificables que delimitan donde quedan las clases del modelo, las de la vista y las controladoras. Las clases que representan la lógica del negocio se localizan en “TPC/apps/penal/models” dentro del paquete “models” se encuentran dos paquetes, uno denominado “bussines” y otro “domain”, el primero está integrado por las clases Model que tienen la responsabilidad de insertar, modificar o eliminar y el segundo por las clases del dominio en las cuales se especifican las consultas a la base de datos, existe allí además un archivo con las clases Base las cuales poseen cada uno de los atributos y relaciones que existen en el modelo de datos del módulo.

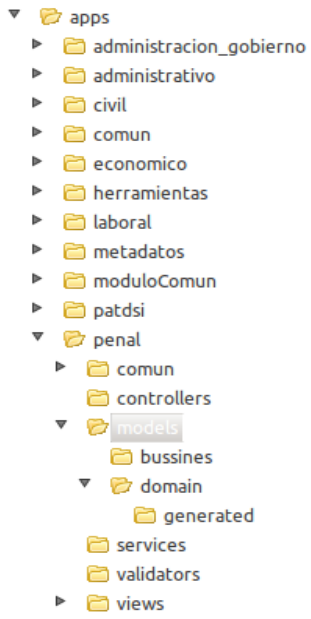

Figura 14: Ubicación de las clases correspondientes al modelo

El conjunto de clases controladoras se encuentran dentro de la jerarquía de paquetes en “TPC/apps/penal/controllers”, este conjunto de clases son las encargadas de recibir las solicitudes o peticiones que se realizan desde la vista, y tienen la capacidad de realizar acciones con el modelo.

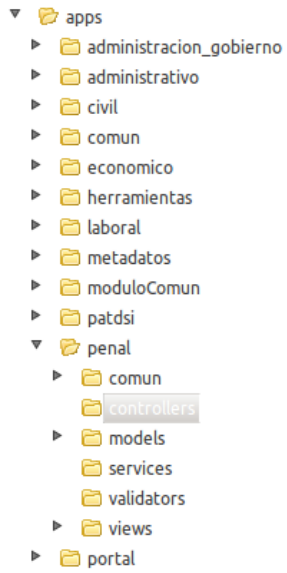

Figura 15: Ubicación de las clases controladoras.

Los archivos correspondientes a la vista están divididos en dos paquetes, el de los ficheros 'phtml', que se encuentra en "TPC/apps/penal/view" y otro en "TPC/web/penal/view" donde se reúnen los ficheros Javascript y otros que responden a la creación de las páginas web y a los que se hace referencia en los 'phtml'.

2.7 Estándares de codificación

Un código fuente legible, fácil de modificar y de darle mantenimiento es el resultado de muchos factores, una faceta del desarrollo de software en la que todos los programadores influyen especialmente es en la técnica de codificación. La mejor vía para asegurarse de que un equipo de programadores realice y mantenga un código limpio, legible y de calidad es establecer un grupo de estándares de codificación los cuales ayuden a que todos los desarrolladores del proyecto trabajen de forma coordinada, logrando con esto que parezca como si un único programador hubiera escrito todo el código de una sola vez.

En el proyecto tribunales debido a la complejidad y la dimensión del proyecto, además del largo período de desarrollo donde las creaciones de varios programadores deben coexistir armoniosamente para el exitoso funcionamiento de los casos de uso que se programan, la aplicación de estándares de codificación resulta muy necesaria y ayuda en gran medida a la colaboración y reutilización de código.

Los estándares de codificación definidos en el proyecto abarcan puntos tales como:

- ✓ Comentarios: las funciones deben estar precedidos por comentarios concisos que le añadan claridad al código y mejoren su comprensión, siempre evitando comentar lo que resulte evidente, deben contar el por qué se hace la operación y no el cómo se hace.

  Ejemplo:

```
//Listar acusados de una causa determinada que no tengan abogado asignado
    public function buscarAcusadosSinAbogados($offset, $limit, $numero) {
        $query = Doctrine_Query :: create();
        $query->SELECT('PN.primernombre,PN.primerapellido,PN.segundoapellido,P
```

Figura 16: Ejemplo del uso de los comentarios.

- ✓ Sentencias compuestas: todas las sentencias del tipo if,for, while, foreach deberán tener llaves aunque sólo contengan una sentencia, de esta forma se evita que ante la introducción posteriormente de nuevas sentencias se produzcan errores accidentales que pueden ser evitados.

  Ejemplo:

```
for ($i = 0; $i < count($ids[0]['PnlDponente']); $i++)
{
    $ponentes[$i] = $ids[0]['PnlDponente'][$i]['idusuario'];
}
```

Figura 17: Ejemplo del uso correcto de las llaves.

- ✓ Caracteres de espacio: se deben usar caracteres de espacio en situaciones como entre un paréntesis cerrado y una llave, después de una coma en la lista de parámetros de un método, entre operadores binarios: +, -, =,* etc.

```
public function buscarAcusadosSinAbogados($offset, $limit, $numero) {
```

Figura 18: Ejemplo del uso de espacios.

- ✓ El nombre de cada una de las clases controladoras estará en correspondencia con el caso de uso correspondiente y seguido de la palabra Controller.

```
class DevolverefpController extends ZendExt_Controller_Secure {
```

Figura 19: Ejemplo del nombre correcto de una clase controladora.

- ✓ Nombres de variables y funciones:

- Evitar dentro de lo posible el uso tanto de nombres largos como de abreviaturas poco conocidas.
- En todo momento se deben utilizar nombres que sean claros, concretos y sin ambigüedades.
- La diferencia entre nombres debe ser bien marcada, evitando nombres que difieran en una letra o en el uso de mayúsculas
- Los nombres de los métodos de las clases controladoras comenzarán con letra minúscula y en caso de estar compuesto por varias palabras la primera comenzara con minúscula pero el resto comenzara con mayúscula y terminarán con la palabra Action.

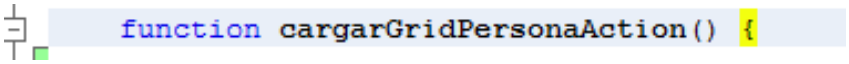

Figura 20: Ejemplo del nombre de las funciones de las clases controladoras.

- En caso de que los nombres de las variables sean compuestos deben estar separadas por un signo de underscore "_" para mejorar su comprensión y lectura.

✓ Cabecera del archivo: En las cabeceras de los archivos ".php" se especifica información referente al autor, la versión, fecha de última modificación tal y como lo muestra la figura.

2.8 Conclusiones

En este capítulo que concluye, se presentaron elementos que forman parte de la solución propuesta como el Modelo de Diseño y el de Implementación, se definió la arquitectura, se especificaron una serie de estándares de codificación y patrones de diseño utilizados en el desarrollo del subsistema, por lo que se puede concluir con que se lograron los objetivos para el capítulo.

.

3 Capítulo 3: Análisis de los Resultados

En este capítulo se realiza un análisis con el fin de valorar la calidad y la fiabilidad de la solución obtenida, para ello se muestran un conjunto de gráficas que son el resultado de la aplicación de algunas métricas utilizadas para la medición de la calidad del diseño y la implementación como el tamaño operacional de las clases, las relaciones entre clases y prueba de caja blanca.

3.1 Evaluación de la calidad utilizando métricas

En este epígrafe se validará la solución mediante el análisis de los resultados que arroja la aplicación de métricas de diseño que permitirán medir de forma cuantitativa ciertos atributos de calidad internos del software que a su vez facilitará expresar una valoración sobre la calidad de la solución. Las métricas que se aplican son las del Tamaño Operacional de las Clases (TOC) y Relaciones entre Clases (RC). Estas métricas que se emplean tienen como objetivos mejorar la comprensión de la calidad de la solución, estimar la efectividad del proceso y mejorar la calidad del trabajo. Están diseñadas para evaluar tanto la complejidad, la funcionalidad como la eficiencia inmersa en el desarrollo de la aplicación y en su conjunto permiten evaluar los siguientes atributos de calidad:

- ✓ **Responsabilidad:** Consiste en la responsabilidad asignada a una clase de un dominio.
- ✓ **Complejidad de implementación:** Se refiere al grado de dificultad que tiene implementado un diseño de clases determinado.
- ✓ **Reutilización:** Consiste en el grado de reutilización presente en una clase dentro de un diseño de software.
- ✓ **Acoplamiento:** Se refiere al grado de dependencia o interconexión de una clase con otras.
- ✓ **Complejidad de mantenimiento:** Se refiere al grado de esfuerzo necesario a realizar para desarrollar una reparación, una mejora o una corrección de algún error de un diseño de software.

- ✓ **Cantidad de pruebas:** Consiste en el grado de esfuerzo para realizar las pruebas de calidad del producto (clase, conjunto de clases, componente, módulo, etc.).

3.1.1 Tamaño Operacional de las Clases

Durante la realización de este trabajo se aplicó la métrica TOC y los resultados se hacen visibles en las gráficas que aparecen a continuación.

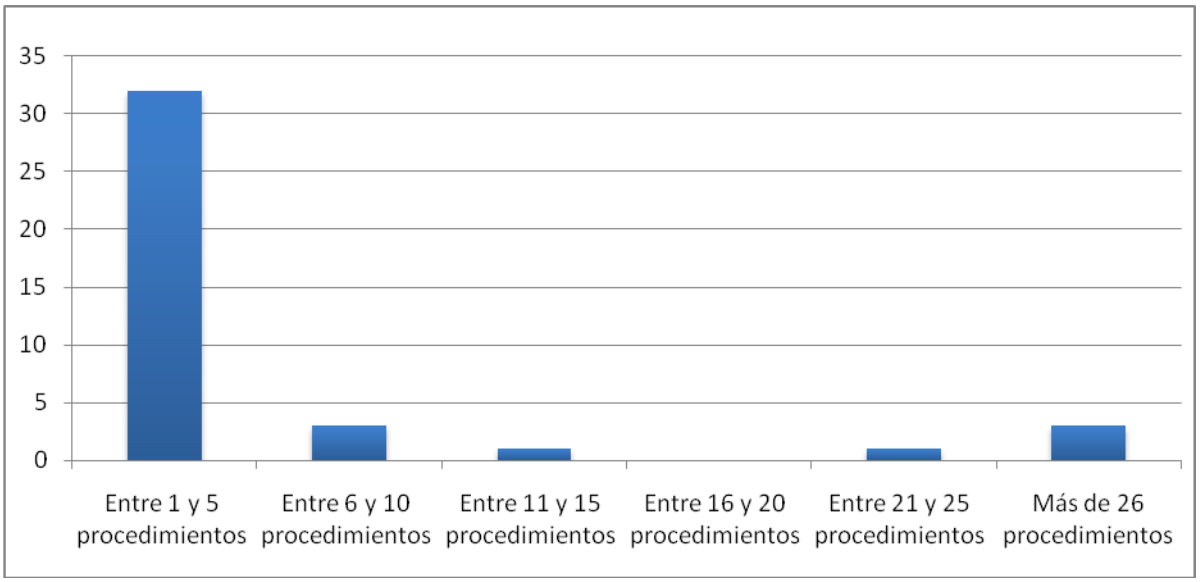

Figura 21: Representación de los resultados obtenidos agrupados en los intervalos definidos.

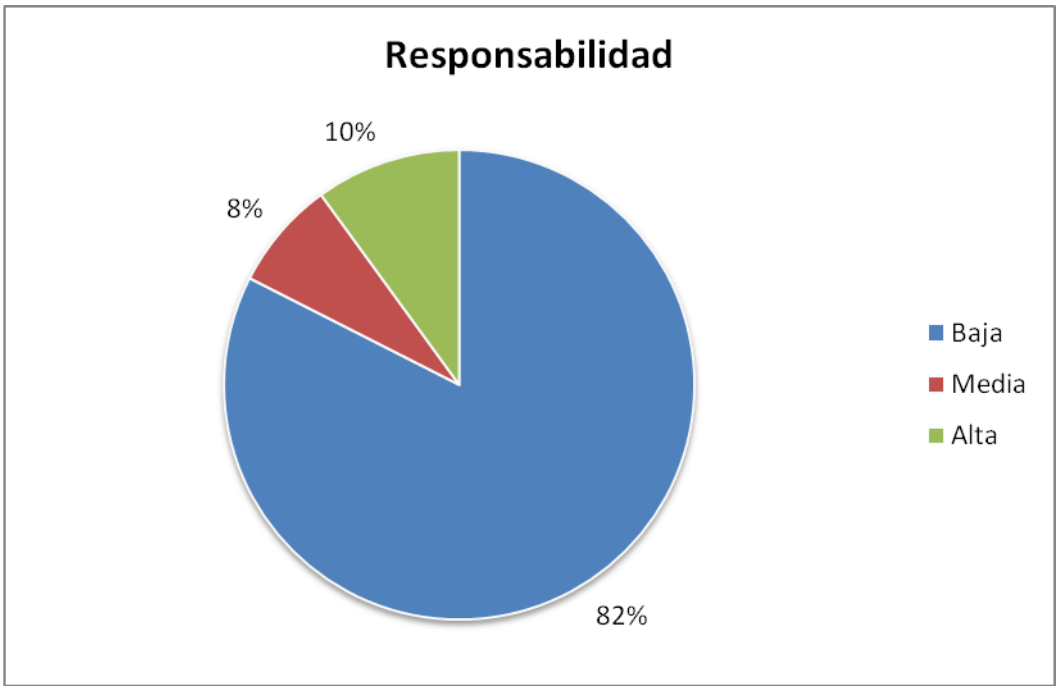

Figura 22: Representación de la incidencia del resultado de la evaluación de la métrica TOC para el atributo Responsabilidad

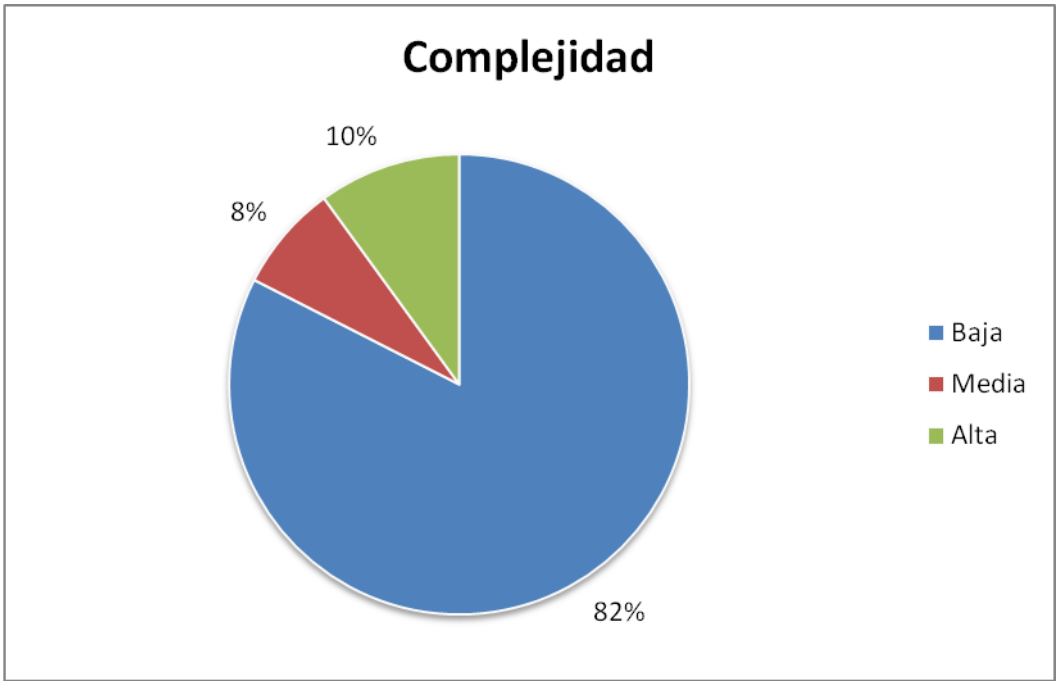

Figura 23: Representación de la incidencia del resultado de la evaluación de la métrica TOC para el atributo Complejidad de implementación.

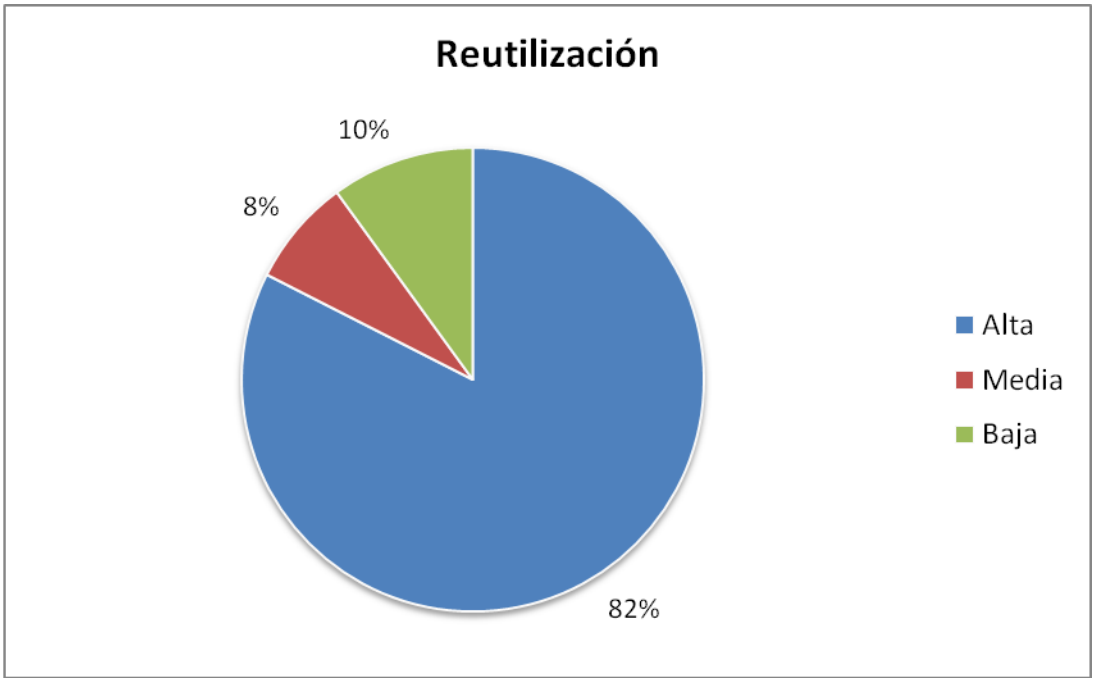

Figura 24: Representación de la incidencia del resultado de la evaluación de la métrica TOC para el atributo Reutilización

Partiendo de que se obtuvieron resultados bajos de la métrica TOC y basándose en las gráficas resultantes que representan los diferentes atributos de calidad que se evalúan, se puede afirmar que la implementación de los casos de uso cuenta con buenos niveles de calidad y eficiencia, dado que se obtuvieron resultados que indican un predominio de altos niveles de reutilización de las clases, bajos niveles de complejidad en la implementación así como una tendencia a bajos niveles de responsabilidad en las clases involucradas en la solución.

3.1.2 Relaciones entre Clases

Durante la realización de este trabajo se aplicó la métrica RC donde los resultados obtenidos se muestran en las gráficas siguientes.

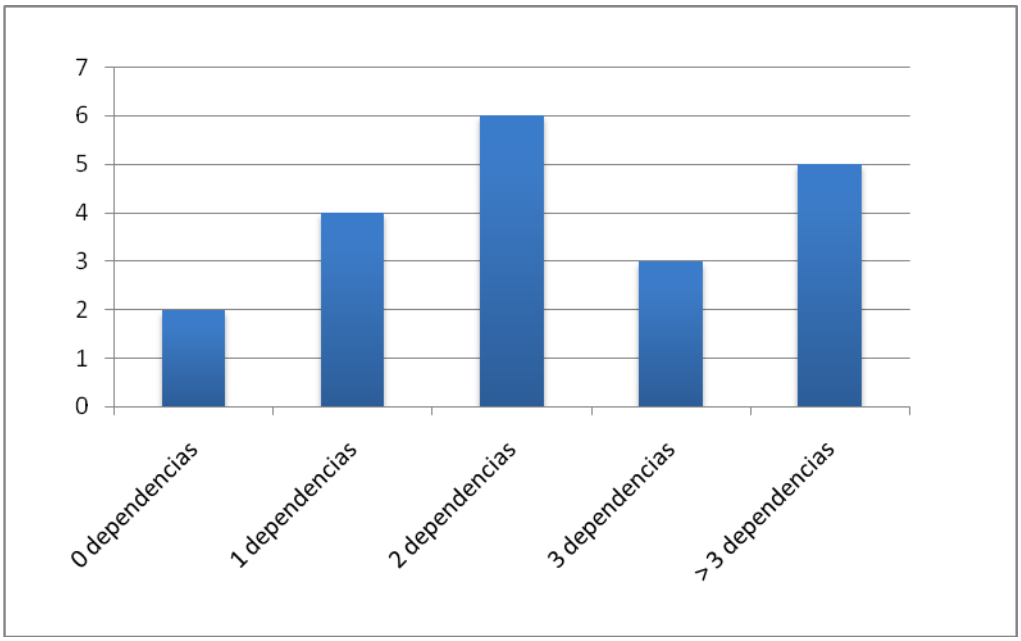

Figura 25: Representación de los resultados obtenidos al analizar las dependencias entre clases.

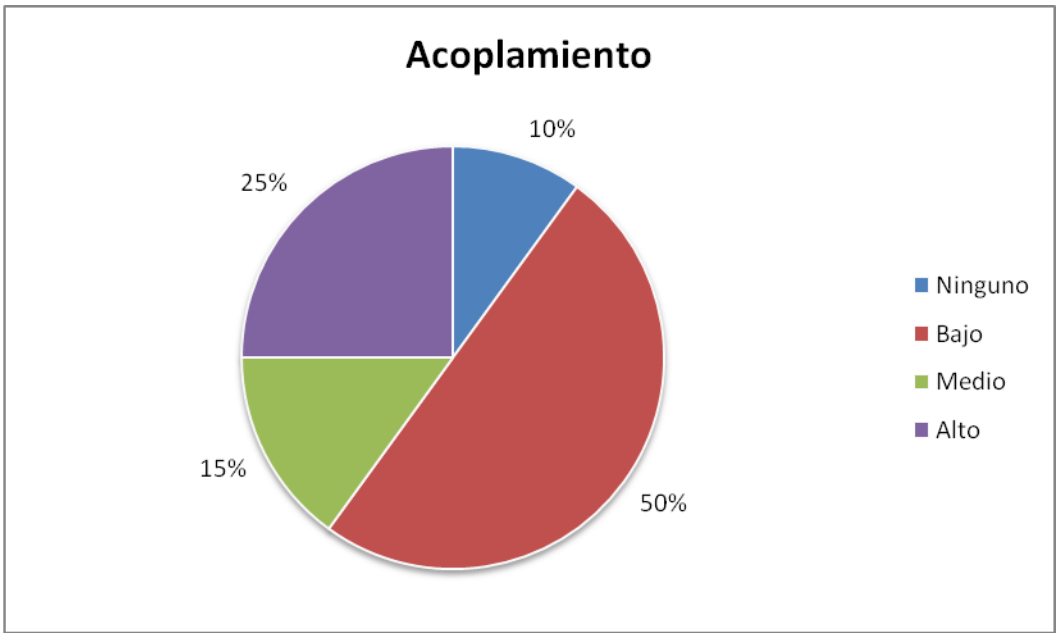

Figura 26: Representación de la incidencia del resultado de la evaluación de la métrica RC en el atributo Acoplamiento.

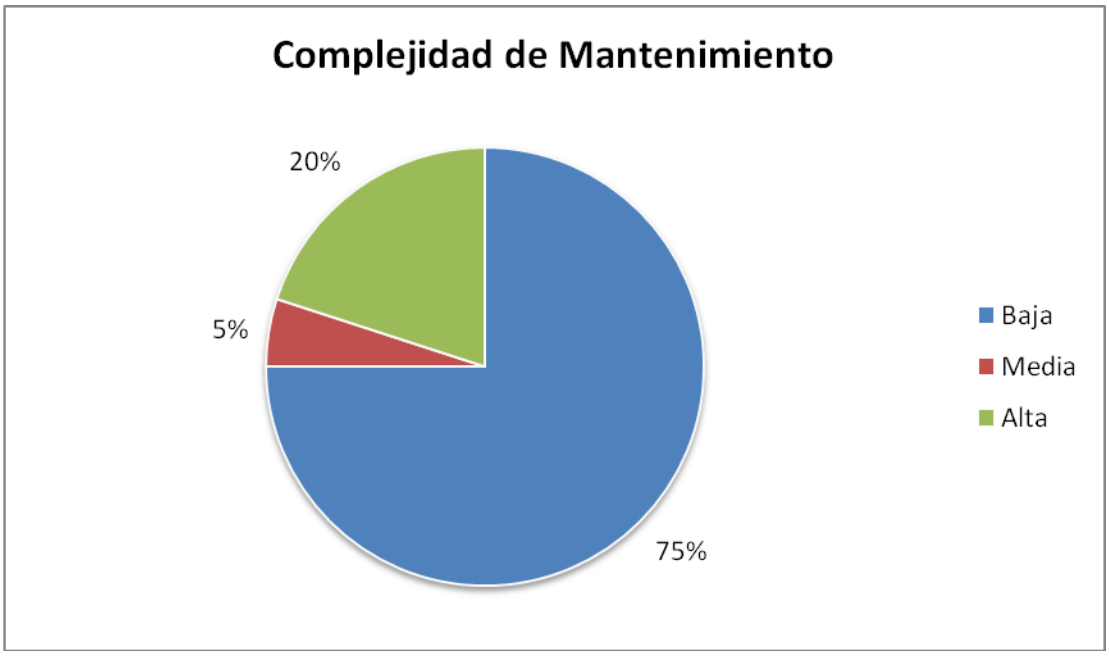

Figura 27: Representación de la incidencia del resultado de la evaluación de la métrica RC para el atributo de calidad Complejidad de Mantenimiento.

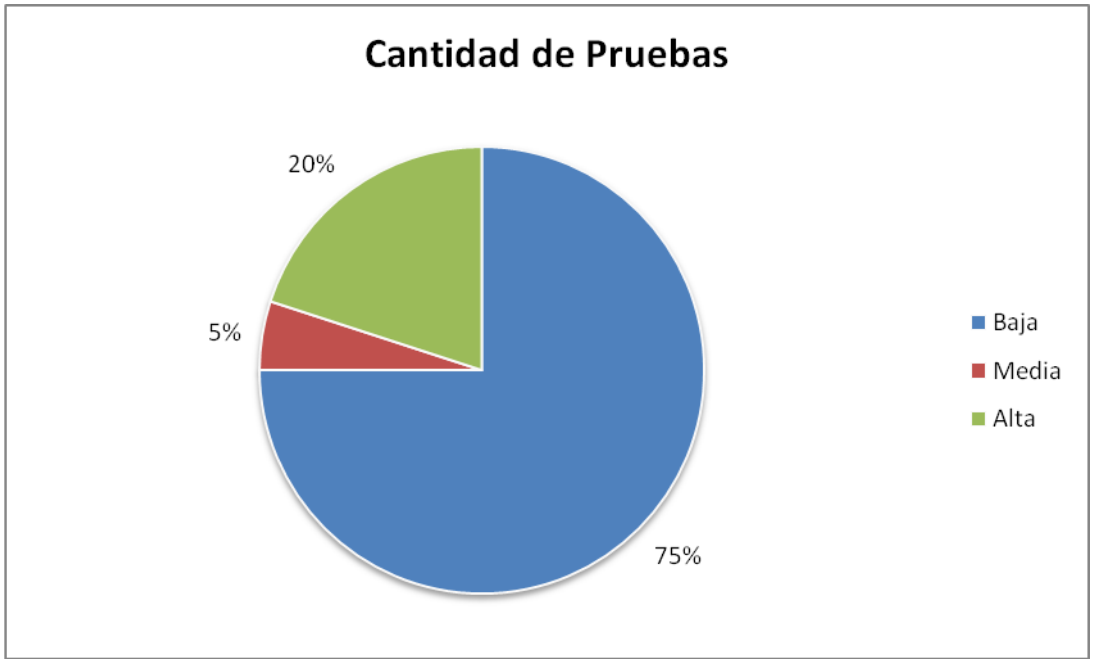

Figura 28: Representación de la incidencia del resultado de la evaluación de la métrica RC para el atributo de calidad Cantidad de Pruebas.

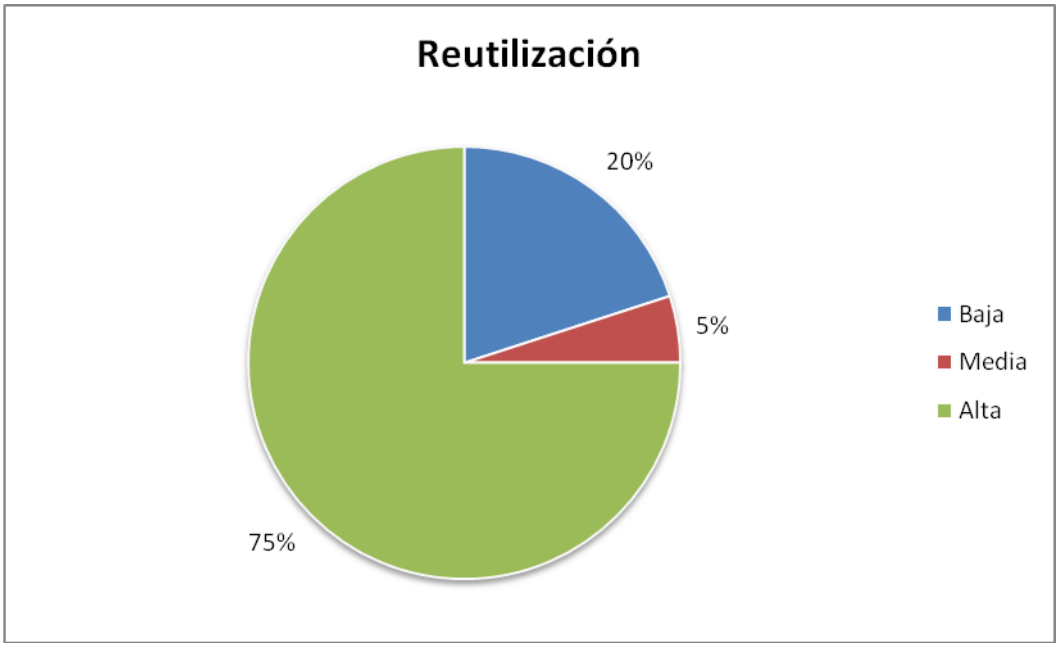

Figura 29: Representación de la incidencia del resultado de la evaluación de la métrica RC para el atributo de calidad Reutilización.

Los valores obtenidos mediante la métrica RC fueron relativamente bajos, lo que indica tal y como muestran las gráficas un predominio del bajo acoplamiento dado por un bajo nivel de dependencias en las clases, los bajos valores representan además resultados que indican un predominio de la no complejidad en el mantenimiento y la cantidad de pruebas necesarias, mientras que los niveles de reutilización son generalmente altos lo que apunta a buenos niveles de eficiencia en la solución.

3.2 Pruebas de caja blanca

Durante la validación de la solución del módulo, se realizó la prueba de caja blanca a un grupo de funcionalidades importantes y relativamente complejas, para ello se aplicó el método del camino básico la cual permite obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como una guía que ayude a la definición de un grupo básico de caminos de ejecución. En la imagen que se muestra a continuación se representa la función cargarAcusadosenSessionAction() de la clase DevolverefpController y se muestra como ejemplo de la aplicación del método de camino básico para la prueba de caja blanca. Se analiza el código y se enumeran cada una de las instrucciones como se puede apreciar en la imagen.

```
function cargarAcusadosenSessionAction()
{
 $efppp =$this->_request->getPost('ExpFP');//1
 $efp = substr($efppp,1,strlen($efppp)-2); //1
 $listado_acusados = DacusadoExt::listarAcusadosSituacion($efp);//1
 $datos =  array(); //1
for($i = 0; $i < count($listado_acusados); $i++)//2
  {
     $datos[$i]['situacionlegal']=
     $listado_acusados[$i]['PnlDacusado'][0]['PnlDacusadosituacionlegal'][0]['PnlNsituacionlegal']['s
     $datos[$i]['idacusado']=$listado_acusados[$i]['PnlDacusado'][0]['idacusado'];
     $datos[$i]['estado']=''; //3
    if($listado_acusados[$i]['segundonombre']!=null||$listado_acusados[$i]['segundonombre']!=' ')//4
    {   $datos[$i]['nombreacusado']=$listado_acusados[$i]['primernombre'].' '.
        $listado_acusados[$i]['segundonombre'].' '.
        $listado_acusados[$i]['primerapellido'].' '.
        $listado_acusados[$i]['segundoapellido']; //5
    }
    else
    {   $datos[$i]['nombreacusado']=$listado_acusados[$i]['primernombre'].' '.
        $listado_acusados[$i]['primerapellido'].' '.
        $listado_acusados[$i]['segundoapellido']; //6
    }
}
  unset($_SESSION['Aperturar']); //7
 $_SESSION['Aperturar']['medidacautelar']['datos']=$datos;//7
 $_SESSION['Aperturar']['medidacautelar']['cantidad_filas']=count($listado_acusados);//7
 echo 'true';//7
 return ;//7
}
```

Figura 30: Función de ejemplo a la que se le aplica el método camino básico.

El primer paso para aplicar el método camino básico es obtener el grafo de flujo, a partir del código de la función.

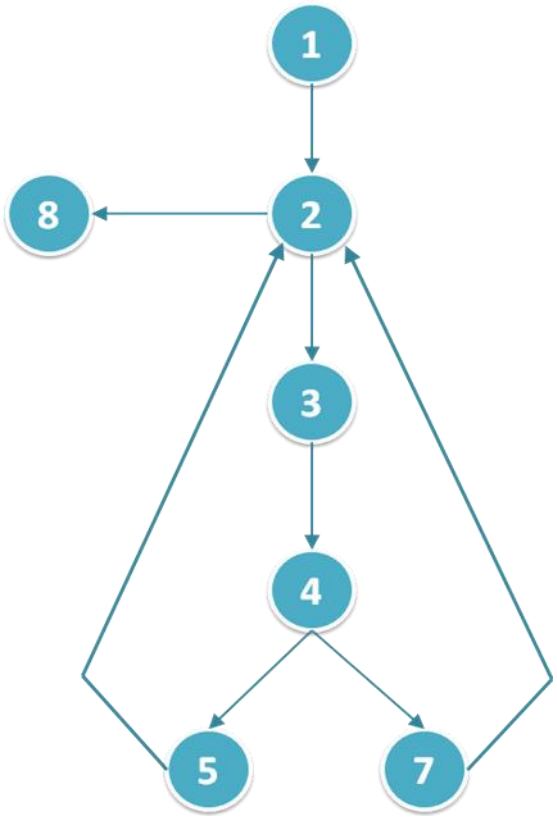

Figura 31: Grafo de flujo correspondiente a la función cargarAcusadoenSessionAction

El segundo paso para llevar a cabo la prueba de caja blanca es obtener la complejidad ciclomática del grafo de flujo, la cual proporciona una idea de la complejidad lógica del programa que se prueba, se puede obtener por tres vías diferentes.

- ✓ La complejidad ciclomática V(G) de un grafo de flujo G se define como:

  V(G) = cantidad de aristas(A) – cantidad de nodos(N) +2

  V(G)=8-7+2=3

- ✓ También se puede definir la complejidad ciclomática V(G) de un grafo como la cantidad de predicados +1, en el grafo anterior los nodos predicados son el 2 y el 4, entonces:

  V(G) = cantidad de nodos predicados + 1

  V(G)= 2+1=3

- ✓ La complejidad coincide con el número de regiones del grafo de flujo.

  V(G)=3

El segundo paso dio como resultado que la complejidad ciclomática del grafo es 3, lo que indica que el conjunto básico de caminos independientes también es 3. El próximo paso es determinar los caminos básicos, cuya especificación es la siguiente:

- ✓ Camino #1: 1 - 2 7
- ✓ Camino #2: 1 - 2 - 3 - 4 - 5 – 7
- ✓ Camino #3: 1 - 2 - 3 - 4 - 6 – 7

Luego se procede a determinar los casos de prueba que permitan la ejecución de cada uno de los caminos, para hacer dichos casos es necesario tener en cuenta 4 aspectos, la descripción del caso de prueba, la condición de ejecución, la entrada y los resultados esperados.

Descripción: La función intenta cargar en sesión datos de los acusados que están relacionados con un expediente determinado por el id y que debe recibir como parámetro.

Caso de prueba para el camino básico #1

Condición de ejecución: Longitud del arreglo $listado_acusados sea cero 0**;**

Entrada: El arreglo $listado_acusados está vacío;

Resultado esperado: Se carga en la sesión un arreglo que no contiene acusados.

El resultado fue el esperado.

Caso de prueba para el camino básico #2

Condición de ejecución: Los acusados no tienen nombres compuestos.

Entrada: El arreglo $listado_acusados contiene datos de acusados pero todos tienes nombres simples.

Resultado esperado: Se carga en la sesión un arreglo con los datos de los acusados.

El resultado fue el esperado.

Caso de prueba para el camino básico #3

Condición de ejecución: Existe al menos un acusado con el nombre compuesto en el arreglo $listado_acusados.

Entrada: El arreglo $listado_acusados contiene datos de acusados y al menos uno tiene un nombre compuesto.

Resultado esperado: Se concatenan los nombres y se carga un arreglo con los datos de los acusados en la sesión.

El resultado fue el esperado.

3.3 Conclusiones

Con la realización del capítulo se determinó que el diseño de la solución propuesta presenta buenos niveles de calidad y fiabilidad, donde las métricas aplicadas indicaron como principales resultados un predominio de la reutilización, el bajo acoplamiento y de la baja complejidad en la implementación y el mantenimiento. Para mejorar la eficiencia y confiabilidad de la solución se aplicaron pruebas de caja blanca para la detección de los errores internos de la aplicación, para su corrección

4 Conclusiones Generales

Las conclusiones a las que se llega con la realización del presente trabajo de diploma son las siguientes:

- ✓ Con la investigación se obtuvo una serie de conocimientos que permitió la fundamentación de las herramientas, técnicas y otras tecnologías necesarias para el desarrollo de la solución.
- ✓ La realización de los Modelos de Diseño e Implementación fue decisiva y de gran ayuda para lograr el desarrollo de la solución.
- ✓ Partiendo de los Modelos de Diseño y de Implementación realizados se logró el efectivo desarrollo de los casos de uso correspondientes.
- ✓ La aplicación de métricas y pruebas de software permitió evaluar en buena medida el nivel de calidad del diseño y la implementación de la solución.

5 Recomendaciones

- ✓ Comenzar el diseño de los Modelo de diseño e implementación del subsistema Penal después de la sentencia.
- ✓ Comenzar el desarrollo de los casos de uso correspondientes al subsistema Penal después de la sentencia.

REFERENCIAS BIBLIOGRÁFICAS

Sparx Systems Pty Ltd. Sparx Systems. [En línea] http://www.sparxsystems.com.ar/resources/tutorial/uml2_sequencediagram.html.

Alegsa. Alegsa - Definicion de Framework - ¿qué es Framework? [En línea] http://www.alegsa.com.ar/Dic/framework.php.

Archer, Tom. 2001. *A Fondo C#.* Madrid : McGraw-HillAnteramerican, 2001. 0-7356-1288-9.

Asensio, Rafael Menéndez Barzanallana. 2011. Universidad de Murcia. [En línea] 13 de Octubre de 2011. http://www.um.es/docencia/barzana/IAGP/IAGP2-Metodologias-de-desarrollo.html.

Baryol, Oiner Gómez. *ARQUITECTURA TECNOLÓGICA PARA EL DESARROLLO DE SOFTWARE.* La Habana : s.n.

Campos, Oscar. 2011. Genveta:dev. [En línea] 25 de septiembre de 2011. http://www.genbetadev.com/paradigmas-de-programacion/diferencias-entre-paradigmas-de-programacion.

Cesar de la Torre LLorente, MIguel angel González Barroso, Unai Zorrilla Castro. 2010. *Guía de Arquitectura N-Capas orientada al dominio con .Net.* s.l. : Krasis Consulting, 2010. 978-84-936696-3-8.

Control de versiones. **Oviedo, Juan.**

Corp, IBM. 2006. *Ayuda RUP.* 2006.

David Sklar, Adam Trachtenberg. Noviembre 2002. *PHP Cookbook.* s.l. : O'Reilly, Noviembre 2002.

Desarrolloweb.com. Desarrollo web, Arquitectura del servidor Apache. [En línea] http://www.desarrolloweb.com/manuales/41/.

Fabien Potencier, François Zaninotto. 2008. *La guía definitiva de symfony.* 2008.

Félix, Alvaro del Castillo San. 2000. *El servidor de web Apache: Introducción práctica.* 2000.

García, Joaquín. 2005. Ingeniero Software. [En línea] 27 de Mayo de 2005. http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

Guía de Subversion.

Guillerón, Gastón. 2009. Gln Blog Ingeniería & Consultoría. [En línea] Julio de 2009. http://glnconsultora.com/blog/?p=3.

I.Jacobson, G.Booch, J.Rumbauggh. 2000. El proceso unificado de desarrollo de software. s.l. : Pearson Addison Wesley, 2000.

León Welicki. El Patrón Singleton. *MSDN.Microsoft.com.* [En línea] http://msdn.microsoft.com/es-es/library/bb972272.aspx.

M.Isabel Gallego Fernández, Manuel Medina Llináz. 2000. *Algorítmica y programación para ingenieros.* Barcelona : Edicions UPC, 2000.

Oracle Corporation. NetBeans. [En línea] http://netbeans.org/community/releases/61/index_es.html.

Pérez, Javier Eguíluz. Febrero 2008. *Introducción a Javascript.* Febrero 2008.

PHP Documentation Group. 2011. *Manual de PHP.* 2011.

2010 . PostgreSQL - es. [En línea] 2010 . http://www.postgresql.org.es/sobre_postgresql.

Pressman, Roger S. 2005. Ingeniería de software. Un enfoque práctico. 2005.

Sensiolabs. 2010. *Doctrine ORM for PHP.* 2010.

Shea Frederick, Colin Ramsay, Steve Blades. 2008. *Learning Ext JS.* s.l. : Packt Publishing, 2008.

SOA Agenda. SOA agenda. [En línea] http://www.soaagenda.com/journal/articulos/que-son-los-frameworks/.

Universidad de Guadalajara. *Métricas, Estimación y Planificación en Proyectos.* Guadalajara : s.n.

Visual Paradigm. 2007. *VP-UML 6.0 User's Guide.* 2007.

Zend Technologies Inc. 2010. Zend Framework des.com. [En línea] 2010. http://manual.zfdes.com/es/introduction.overview.html.