Universidad de las Ciencias Informáticas

Facultad 3

DESARROLLO DE LOS PROCESOS DISCIPLINA Y DERECHO DE LA MATERIA LABORAL EN LOS TRIBUNALES POPULARES CUBANOS.

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

Autores: Mónica Aragón Mujica

Rances Miguel Hernández Peraza

Tutor: Ing. Elsydania López Guerra

La Habana, Cuba

Curso 2011-2012

“

...Revolución es unidad, es independencia, es luchar por nuestros sueños de justicia para Cuba y para el mundo, que es la base de nuestro patriotismo, nuestro socialismo y nuestro internacionalismo.

”

Fidel Castro

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Autores:

_____ _____

Mónica Aragón Mujica Rances Miguel Hernández Peraza

Tutor:

_____

Ing. Elsydania López Guerra

A mi mamá, por estar siempre junto a mí en todos los momentos difíciles de mi vida, por apoyarme, por darme todo su amor y su cariño, por escucharme, por enseñarme que todo se logra con esfuerzo y dedicación.

A mi papá, por cuidarme, por darme su fuerza y su valor, por hacerme sonreír cuando más lo necesité, por inspirarme a intentar ser cada día mejor, por ser mi más leal apoyo durante toda mi vida de estudiante.

A toda mi gran familia, principalmente a mi abuelito que siempre lo tengo presente donde quiera que esté, mi abuelita que la quiero con la vida, a todos mis primos y tíos que formamos una hermosa familia que se apoya en todos los momentos.

A Yesenia, mi amiga de siempre, que aunque pasen los años sé que puedo contar con ella para lo bueno y lo malo.

A mis compañeros del proyecto TPC, especialmente a Yosviel, por ayudarme tanto en mi tesis y explicarme con detalle cada duda, al profe Maikel por ser un gran apoyo, al profe Joan Jon por estar siempre dispuesto a enseñarme, a Dairon, a Martin, a Darián y a mi tutora Elsydania.

A los amigos que hice en la UCI, a Carluchu, a Alayna, a Félix, a Dailen, a Yaidel, a Yoslenys, que siempre han estado ahí para brindarme su apoyo.

A todos los que de alguna manera hemos compartido estos cinco años de alegrías y preocupaciones.

A mis amigos de la Lenin que aunque estemos lejos, los recuerdo siempre.

Mónica Aragón Mujica

A mi mamá, por ser la persona más importante de mi vida, quien me ha brindado un mundo lleno de amor y felicidad. Madre aunque sea de vinagre.

A mi papá, por ser la persona a seguir, el ejemplo por el cual me he guiado para lograr todas las metas que me he trazado y las que faltan por llegar.

A mis segundas madres: Mima, Aly, Mely, Marce y Echia, que siempre han estado en las buenas y malas, guiándome por los mejores caminos.

A mi familia, por brindarme tanto apoyo e incondicionalidad, por ser todos el motor impulsor de este ingeniero.

A mi novia Flaky, por ser tan especial conmigo, tener la paciencia de soportarme y siempre tener tiempo para mí, en las buenas y malas, por hacer de mí una persona mejor.

A mis hermanos Yoss, Drigg, Daniel, Julito, Butyn , Reinier, Yaidel, Karly, Pucho y el Lokol por ayudarme con los estudios para lograr estar aquí.

Al personal del proyecto, por ayudarme en esta ardua tarea, desde los compañeros hasta los profesores, en especial al profe Joan Jon y a mi tutora Elsydania.

A la plebe del 9109, Irmel, Doa, Yoslenys, Joan Charles y Daylen por aguantarme durante tanto tiempo y ser parte de la familia que llevo conmigo.

A los amigos del grupo, del fútbol y todas las amistades que no han podido llegar hasta aquí, por haberme brindado la posibilidad de poder compartir con ustedes, y ser parte de esta familia UCI.

Rances Miguel Hernández Peraza.

Dedico mi trabajo de diploma a mis padres, que son la luz que ilumina mi vida en los momentos más oscuros.

Mónica

Dedico mi trabajo de diploma a mis padres, a mis abuelas Mima y Marcela, a mis tías Mely y Aly, a mi bella Echia, y a la familia que son los pilares que mueven este corazón.

Rances

RESUMEN

Los procesos Disciplina y Derecho de la materia Laboral que se desarrollan en la instancia municipal de los Tribunales Populares Cubanos, se inician ante la presentación de una demanda ya sea de forma verbal o escrita. En el caso de Disciplina Laboral se efectúa al producirse una violación a los derechos de un trabajador o representante de la administración de la empresa u organismo a la que pertenece. Mientras que Disciplina Laboral al surgir alguna inconformidad ante la imposición de una medida disciplinaria dictada por el órgano rector encargado de solventar estos conflictos. La tramitación de la demanda que se genera durante estos procesos actualmente presenta dificultades y es propensa a errores debido principalmente a la gran cantidad de volúmenes de información a manejar, lo que imposibilita su agilización. Para dar solución a esta situación, en el presente trabajo se elabora una solución de software, a partir del diseño e implementación de los requisitos, previamente especificados, que debe cumplir el sistema para la agilización de estos procesos. Con este fin se realiza una fundamentación de todas las herramientas y técnicas empleadas por la arquitectura de software definidas para el desarrollo de la aplicación. También se presentan los artefactos resultantes que conforman la propuesta de solución, los cuales serán evaluados para obtener su grado de calidad según las métricas y mecanismos propuestos. Con esta herramienta los Tribunales Populares Cubanos contarán con un producto funcional que permita gestionar de manera diligente los procesos Disciplina y Derecho de la materia Laboral.

PALABRAS CLAVES

Tribunales Populares Cubanos, Materia Laboral, Proceso Disciplina Laboral, Proceso Derecho Laboral, Diseño, Implementación, Métricas, Producto Funcional.

ÍNDICE DE CONTENIDOS

Introducción **2**
Capítulo 1: Fundamentación teórica **8**
1.1. Estado del arte de los sistemas de gestión procesal **8**
1.1.1 Los sistemas Minerva y Lexnet 8
1.1.2 El sistema Justicia.CAT 9
1.1.3 El sistema Sisprop 10
1.2. Proceso Laboral. Procedimientos Derecho y Disciplina Laboral **10**
1.3. Metodología de desarrollo de software: RUP **11**
1.4. Aplicación web para el desarrollo del SIT **12**
1.5. Arquitectura de software del SIT **13**
1.5.1 Patrones de diseño y de arquitectura 13
1.5.1.1 Patrones de diseño 14
1.5.1.2 Patrones de arquitectura 15
1.6. Marco de trabajo **16**
1.6.1 Sauxe 16
1.6.2 ExtJs 18
1.7. Object Relation Mapper (ORM): Doctrine **18**
1.8. Lenguajes de programación: PHP y JavaScript **18**
1.9. Entorno de desarrollo integrado: Netbeans **19**
1.10. Software para los servidores **20**
1.10.1 Software para el servidor web: Apache 20
1.10.2 Software para el servidor de base de datos: PostgreSQL 21
1.11. Herramienta de modelado: Visual Paradigm **22**
1.12. Métricas de software **23**
1.13. Conclusiones **25**
Capítulo 2: Solución propuesta **26**
2.1 Representación arquitectónica **26**
2.2 Vista lógica **26**
2.2.1 Estructura organizativa para la implementación del sistema 27
2.2.2 Ejemplificación del uso de los patrones de diseño 30
2.2.3 Modelo de diseño 34
2.2.3.1 Diagrama de clases del diseño 34

2.2.3.2 Diagrama de interacción 38
2.3 Vista de implementación 40
2.3.1 Modelo de implementación 40
2.3.1.1 Especificación de los componentes de Sauxe 40
2.3.1.2 Diagrama de componentes 42
2.4 Vista de despliegue 43
2.4.1 Modelo de despliegue 43
2.4.1.1 Especificaciones para el despliegue del sistema 44
2.4.1.2 Diagrama de despliegue 45
2.5 Conclusiones 47
Capítulo 3: Análisis de los resultados 48
3.1 Evaluación del modelo de diseño 48
3.1.1 Resultados del instrumento de evaluación de la métrica Árbol de Profundidad de Herencia (APH) 48
3.1.2 Resultados del instrumento de evaluación de la métrica Número de Descendiente (NDD) 50
3.1.3 Resultados del instrumento de evaluación de la métrica Tamaño de Clase (TC) 53
3.2 Evaluación de la calidad del producto: pruebas de Caja Blanca 56
3.3 Evaluación de la calidad del producto: pruebas de Caja Negra 60
3.4 Conclusiones 61
Conclusiones generales 63
Recomendaciones 64
Bibliografía 65
Glosario 68

ÍNDICE DE FIGURAS Y TABLAS

Figura 1: Sistema judicial cubano 3
Figura 2: Organización propuesta por Sauxe 27
Figura 3: Estructura de la capa de presentación 28
Figura 4: Estructura de la capa de control de la lógica del negocio y del modelo 30
Figura 5: Patrón Experto para el diseño orientado a objetos 31
Figura 6: Patrón Bajo Acoplamiento para la programación orientada a objetos 32
Figura 7: Patrón Controlador para la programación orientada a objetos 32
Figura 8: Diagrama de clases para el patrón IoC 33
Figura 9: Diagrama de clases del patrón Singleton para obtener una instancia única 34
Figura 10: Diagrama de clase del diseño "Registrar Escrito Inicial" 36
Figura 11: Diagrama de secuencia "Registrar Escrito Inicial", para el escenario Pasar a Definitivo 39
Figura 12: Diagrama de componentes de Sauxe de la materia Laboral 43
Figura 13: Despliegue del SIT a nivel nacional 46
Figura 14: Gráfico: niveles de herencia en el diseño 50
Figura 15: Gráfico: porcentaje de la cantidad de pruebas en el diseño 52
Figura 16: Gráfico: porcentaje del total de clases por tamaño 55
Figura 17: Gráfico: representación de los niveles de reutilización en el diseño 55
Figura 18: Método eliminarRowdemandanteAction 56
Figura 19: Grafo de flujo del método eliminarRowdemandanteAction 58
Figura 20: Gráfico: resultados por iteraciones de los casos de prueba de caja blanca 60
Figura 21: Gráfico: resultados por iteraciones de los casos de prueba de caja negra 61
Tabla 1: Asignación de responsabilidades en el patrón Experto 31
Tabla 2: Descripción del caso de uso “Registrar Escrito Inicial” 35
Tabla 3: Valores del nivel jerárquico de clases 48
Tabla 4: Jerarquía de clases por casos de uso 49
Tabla 5: Valores de nivel de descendencia de clases 50
Tabla 6: Descendencia de clases por casos de uso 51
Tabla 7: Parámetros de calidad para altos valores de TC 53
Tabla 8: Cantidad de clases por tamaño 53
Tabla 9: Representación del tamaño de clases de la materia Laboral 54

INTRODUCCIÓN

La información digital que el hombre contemporáneo necesita almacenar en todas las esferas de su vida, ya sea de manera indeleble o provisionalmente, constituye uno de los patrimonios digitales más trascendentales de la historia de la humanidad. Este fenómeno se ha extendido a tal punto que hoy en día se destina gran cantidad de recursos a su protección y procesamiento, lo cual ha sido posible gracias al auge alcanzado por la aplicación de las Tecnologías de la Información y las Comunicaciones (TICs) en los últimos años, posibilitando su administración y gestión.

Una de las áreas en la que se pone de manifiesto la necesidad de tramitar información de forma confiable, a fin de descartar la mayor cantidad de riesgos posibles, es en el área jurídica. En Cuba se hace necesario manejar grandes volúmenes de información en los tribunales, lo que ha provocado que el procesamiento de los datos que se manejan sea muy lento y se cometan disímiles errores, tales como la pérdida de escritos en el archivo y creación de los expedientes; así como tachaduras, borrones y repetición de palabras en el registro de documentos. En ocasiones los documentos se confeccionan varias veces como consecuencia de la falta de archivos actualizados y de la gran cantidad de datos a guardar, lo que provoca que la información sea redundante, el tiempo de tramitación sea bastante prolongado y la búsqueda de un determinado expediente se dificulte.

Con el fin de desarrollar la infraestructura tecnológica del Sistema de Informatización de Tribunales (SIT), asegurar su gestión, administración y explotación en los principios de la seguridad informática y como parte del proceso de informatización que se está realizando en el país; se concibió el proyecto Tribunales Populares Cubanos (TPC) en el año 2009, el cual está vinculado al Centro de Gobierno Electrónico (CEGEL), correspondiente a la facultad 3 de la Universidad de las Ciencias Informáticas (UCI).

Con la implantación del SIT se agilizará la gestión de los procesos desarrollados en los tribunales convirtiendo los expedientes judiciales en expedientes digitales. El sistema facilitará que el almacenamiento y actualización de la información sea más seguro y organizado, el control y cumplimiento de los términos procesales sea efectivo, los recursos materiales y el capital humano se racionalicen adecuadamente, y que el flujo de información entre las distintas instituciones del sector jurídico como: la Policía Nacional Revolucionaria (PNR), la Dirección de Prisiones, Fiscalía y Bufetes Colectivos, se realice de forma loable.

En Cuba el sistema judicial está estructurado de manera escalonada con relación a sus tres instancias (Figura 1):

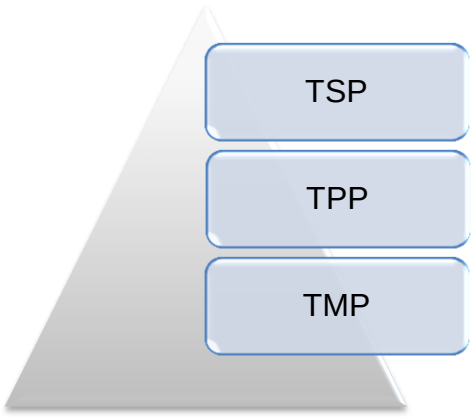

Figura 1: Sistema judicial cubano

TSP. Tribunal Supremo Popular.
TPP. Tribunales Provinciales Populares.
TMP. Tribunales Municipales Populares.

Los procesos están divididos en cinco salas de justicia o materias:

1. Materia Penal.
2. Materia Civil.
3. Materia Administrativa.
4. Materia Económica.
5. Materia Laboral.

Para la gestión de las materias se han concebido varios módulos o subsistemas entre los que se encuentra el Laboral, cuyo proceso se encarga de solventar por la vía judicial los conflictos derivados de las situaciones dadas en organizaciones y empresas en las que se quebranta el cumplimiento de una medida o se viola algún derecho. Este proceso se realiza regido por el Órgano de Justicia Laboral de Base (OJLB), y puede ser llevado a apelación. Actualmente la tramitación de los procesos Disciplina y Derecho Laboral que se desarrollan en los TMP, presenta varias deficiencias que dificultan su diligencia a pesar de que son procesos sencillos que implican como máximo dos jueces.

Para favorecer la celeridad de estos procesos, el equipo de análisis identificó los

requerimientos a tener en cuenta en el desarrollo de la solución informática, los que fueron llevados a un lenguaje entendible para los programadores y se especificaron las funcionalidades que precisa la aplicación. Teniendo en cuenta la arquitectura de desarrollo definida para el desarrollo del SIT y partiendo de los requisitos identificados para los procesos Disciplina y Derecho, se hace necesaria la implementación de las funcionalidades que permitirán:

- Agilizar las tareas realizadas manualmente por la secretaria, como el registro de los datos del escrito de la demanda presentada y de las pruebas que integran el proceso, y la elaboración de las actas de comparecencia, de diligencia de citación y de diligencia con la demanda establecida.
- Los abogados puedan registrar y visualizar los escritos de representación procesal que ellos presenten por alguna de las partes.
- Los jueces puedan realizar búsquedas y consultas de los expedientes en formato digital, a partir de criterios como el número de expediente, la fecha en que fue creado o el estado del acto procesal en el que se encuentra.
- Facilitar la interacción entre el tribunal y el órgano rector de la materia (OJLB).
- Viabilizar la búsqueda de los libros de presentación de escritos, de numeración de sentencia; así como la creación de las actas, providencias, notificaciones y citaciones realizadas a las partes, que se generan a lo largo de los procesos Disciplina y Derecho Laboral.
- Conocer el estado de los procesos Disciplina y Derecho Laboral desde que comienza hasta que se archiva.
- Registrar en los diferentes libros los escritos presentados en la sala de lo Laboral.

Para la solución de la problemática descrita anteriormente se plantea como **Problema a resolver**: ¿Cuál es la especificación de los artefactos necesarios para la informatización de los requisitos identificados que permitirán obtener un producto funcional, para los procesos Disciplina y Derecho de la materia Laboral en los Tribunales Populares Cubanos?

Se define como **Objeto de estudio:** Proceso de desarrollo de software de los procesos judiciales.

Para dar respuesta al problema planteado se ha determinado como **Objetivo general:** Desarrollar los procesos Disciplina y Derecho de la materia Laboral en los Tribunales

Populares Cubanos, para obtener un producto funcional que cumpla con los requisitos identificados.

Luego de definir el objeto de estudio se proyecta el siguiente **Campo de acción:** Diseño e Implementación de los procesos laborales en los Tribunales Populares Cubanos.

Como **Idea a defender** se define que: Con la especificación de los artefactos necesarios para la informatización de los requisitos identificados para los procesos Disciplina y Derecho de la materia Laboral en los Tribunales Populares Cubanos, se obtendrá un producto funcional que cumpla con los requisitos identificados.

Según el objetivo general planteado se trazan los siguientes **Objetivos específicos:**

1. Elaborar el marco teórico-referencial de la investigación para fundamentar el uso de las herramientas definidas para el desarrollo del Sistema de Informatización de Tribunales.
2. Realizar el modelo de diseño para transformar los requisitos identificados en una representación técnica del software que se va a desarrollar.
3. Desarrollar el modelo de implementación para proporcionar una visión general de lo implementado en el sistema.
4. Realizar la implementación del sistema para obtener un producto funcional.
5. Validar la solución propuesta para evaluar la calidad del trabajo realizado, corregir errores y asegurar que el producto obtenido satisface los requisitos acordados.

Como **Tareas de la investigación** a realizar para dar cumplimiento a los objetivos específicos se plantean:

1. Análisis del estado del arte de los sistemas para la gestión de procesos judiciales.
2. Caracterización de las herramientas, elementos de programación, patrones de diseño y arquitectónicos definidos para el desarrollo del Sistema de Informatización de Tribunales.
3. Definición de métricas y/o mecanismos para evaluar el diseño e implementación realizados.
4. Caracterización de la arquitectura definida para la implementación de la solución informática.
5. Elaboración de diagramas de clases del diseño.
6. Elaboración de diagramas de secuencia.
7. Elaboración del diagrama de despliegue.

8. Elaboración del diagrama de componentes.
9. Implementación de los componentes.
10. Diseño de los casos de prueba.
11. Validación de la solución a partir de los casos de prueba.

Los **métodos científicos**, como la forma de abordar la realidad, de estudiar la naturaleza, la sociedad y el pensamiento, con el propósito de descubrir su esencia y sus relaciones, constituyen una estrategia fundamental para afrontar el problema a resolver. Entre ellos están los métodos: t**eóricos y empíricos**. Durante la investigación realizada se aplicaron:

Métodos teóricos:

Histórico – Lógico: Se empleó para la comprensión del flujo de los procesos laborales para obtener las pautas a seguir relacionadas con el objeto de estudio.

Analítico – Sintético: Este método fue utilizado para realizar un análisis de los resultados obtenidos del estudio del estado del arte y seleccionar los elementos más importantes y sus relaciones.

Modelación: Se aplicó en la creación de los diagramas que representan abstracciones o modelos, para la creación de los artefactos correspondientes al modelo de diseño y de implementación.

Métodos empíricos:

Medición: Se utilizó para medir los principales atributos de calidad del software y para obtener información numérica acerca de los resultados obtenidos en la aplicación de las métricas y mecanismos de evaluación.

El presente trabajo se ha estructurado en tres capítulos resumidos del siguiente modo:

Capítulo 1: Fundamentación teórica

En este capítulo se abordarán aspectos teóricos sobre la metodología, herramientas, elementos de programación definidos y los patrones de diseño utilizados durante el desarrollo de la solución propuesta; así como las métricas y/o mecanismos que se emplearán para medir la calidad del producto obtenido.

Capítulo 2: Solución propuesta

En este capítulo se especifican a través de ejemplos concretos los patrones de diseño y estilos arquitectónicos utilizados y la arquitectura definida para la implementación de la

solución informática. También se presentan el modelo de diseño y el modelo de implementación. Como principal resultado se evidencia la solución práctica propuesta a la investigación realizada.

Capítulo 3: Análisis de los resultados

En este capítulo se realiza la validación de los artefactos resultantes de la solución propuesta a partir del diseño de los casos de prueba y del modelo de diseño. Estos elementos son evaluados mediante la aplicación de métricas y/o mecanismos definidos para evaluar y medir la calidad del diseño y la implementación del software.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se referencian algunos de los sistemas existentes que se especializan en la gestión de los procesos judiciales tanto a nivel nacional como internacional. Se describen brevemente los procedimientos Disciplina y Derecho de la materia Laboral desarrollados en los Tribunales Populares Cubanos, en su instancia municipal, y las pautas a seguir durante el mismo. Se fundamenta sobre las tecnologías y herramientas que serán utilizadas para el desarrollo del sistema según la arquitectura definida y los beneficios que aportan en la implementación de los requisitos identificados; así como los patrones de diseño y estilos arquitectónicos utilizados en la realización de la solución informática. También se realizará un estudio sobre las métricas y/o mecanismos seleccionados para la posterior validación del diseño e implementación.

1.1. Estado del arte de los sistemas de gestión procesal

La informatización de los procesos judiciales constituye un elemento importante para el desarrollo de la infraestructura tecnológica de un país, pues la administración correcta de justicia constituye un eslabón fundamental en una sociedad. El Tribunal Supremo Popular cubano tiene como principal perspectiva "*contribuir notablemente a la seguridad jurídica del país y a la realización plena de los derechos, al actuar con profesionalidad y sentido de justicia, y alcanzar alta credibilidad, autoridad y reconocimiento social*" (Productiva, Junio 2010). Para analizar la situación actual de los sistemas de gestión procesal se tendrán en cuenta los exponentes más conocidos tanto a nivel nacional como internacional.

1.1.1 Los sistemas Minerva y Lexnet

A partir del 2001, el ministerio de justicia español crea el sistema informático de gestión procesal Minerva Nueva Oficina Judicial (NOJ) en su versión 1.0, para la gestión procesal integral en el Tribunal Supremo con el objetivo de garantizar un mayor control de las ejecutorias penales y facilitar la tarea en los juzgados de instrucción. Este sistema se integra con Lexnet, que es un sistema de gestión de notificaciones telemáticas desde los juzgados a los profesionales de la justicia (abogados y procuradores) utilizado en la administración de justicia española, y se especializa en la presentación de escritos y

documentos, el traslado de copias y la realización de actos de comunicación procesal por medios telemáticos. Su uso se basa en un programa similar a un sistema de correo web que permite, previa identificación con certificado y con una tarjeta criptográfica, enviar notificaciones a profesionales desde los juzgados con efectos legales plenos (Delgado García, y otros, 2007). Actualmente estos sistemas presentan dificultades que no permiten su correcto funcionamiento. En el caso de Lexnet uno de los problemas que presenta en su implementación es no es multiplataforma pues no permite ejecutarse en otros sistemas operativos como Linux o MacOS. Tampoco permite ejecutarse en otro navegador que no sea Internet Explorer, debido a que para realizar el proceso de firma de comunicación por quien envía las notificaciones requiere de la utilización del archivo binario ActiveX, el cual es propio de determinadas versiones de Windows y de Internet Explorer. Por otro lado el sistema Minerva, ha presentado problemas en su funcionamiento, debido a fallas en el servidor que no han permitido el envío de las notificaciones telemáticas a los procuradores de los procedimientos penales en el tiempo establecido.

1.1.2 El sistema Justicia.CAT

"*El proyecto e-justicia.cat es la garantía de un sistema integrado de gestión procesal judicial*" (Dotras, 2010). El objetivo de este sistema es lograr la administración de justicia en Cataluña, España; a partir de la ejecución de un nuevo sistema integrado de gestión procesal judicial que permitirá la revisión de los procesos en los juzgados, fiscalías, entre otros. La incorporación de los expedientes electrónicos a partir de la digitalización de los expedientes en papel, facilitará la protección de los actos procesales con la utilización de la firma digital que permite su procesamiento con las máximas garantías de seguridad, autenticidad e integridad. Con esto se pretende lograr paulatinamente la obtención de un juzgado sin papeles. Este proyecto se encuentra en estos momentos en la primera fase de desarrollo, que por el momento sólo incluye la jurisdicción del proceso civil en su primera instancia.

Este sistema no es idóneo para su despliegue en Cuba pues su sistema judicial no se ajusta a la legislación cubana, porque los procedimientos son diferentes en cuanto a los términos, los órganos que rigen los actos procesales y la forma de comunicarse entre estos.

1.1.3 El sistema Sisprop

Sisprop, por sus siglas Sistema para la tramitación de Procesos Penales, es un sistema creado en Cuba que persigue como objetivo la seguridad de la información que maneja, y la flexibilidad ante las modificaciones que pudiera sufrir la Ley Procesal Penal. Se plantea entre sus principales requerimientos el procesamiento de los expedientes correspondientes a procesos vistos anteriormente, facilitar la emisión de los documentos (autos, providencias, notificaciones, etc.), controlar y notificar el vencimiento de los términos, entre otros.

A pesar de todos los beneficios que aporta, presenta varias deficiencias, como que no aporta datos estadísticos al no permitir la emisión de reportes. No admite el registro de casi ningún dato de la fase judicial ni de la decisión del tribunal referente al proceso penal, lo que no permite el desarrollo del proceso judicial completamente. Las dificultades planteadas deben valorarse en el desarrollo del SIT, pues provocarían errores fatales en el sistema que influirían negativamente en la gestión de procesos vitales para la administración de la justicia.

Por lo que se concluye que se necesita la implementación del SIT para gestionar todos los procesos judiciales en sus diferentes instancias, de manera integrada y precisa, regidos por la legislación cubana y que elimine todas las barreras que obstaculizan su celeridad.

1.2. Proceso Laboral. Procedimientos Derecho y Disciplina Laboral

El proceso comienza con la presentación de la demanda verbal o escrita al TMP correspondiente por una de las partes (trabajador o administración) o por un miembro del OJLB. La demanda puede ser presentada en el primero de los casos, por el trabajador en reclamo de algún derecho laboral o por la administración solicitando al TMP la imposición de una medida disciplinaria. En el segundo de los casos es presentada por inconformidad del trabajador o de la administración por el fallo dictado por el OJLB en materia de derechos laborales o de disciplina laboral cuando la medida inicialmente aplicada sea la del traslado del trabajador a otra plaza con pérdida de la que ocupaba, o la de separación definitiva de la entidad.

El sistema de justicia laboral cubano, establecido por el Decreto-Ley 176, establece que en la solución de las reclamaciones o inconformidades sobre la imposición de medidas

disciplinarias o por la infracción de los derechos laborales, intervienen el OJLB y el TMP que corresponda y en los casos que la ley determine, el TSP para conocimiento de los procedimientos de revisión de sentencia que se ejecutan en esta instancia. Con este decreto fue posible la solución de conflictos laborales en los centros de trabajo y una correcta conciliación del uso de la vía judicial, cuyas precisiones procesales para su aplicación fueron dictaminadas por el Consejo de Gobierno del TSP con la Instrucción No. 197, la cual fue actualizada en el pasado año 2010 y es la que impera en estos momentos.

1.3. Metodología de desarrollo de software: RUP

Una metodología de desarrollo de software es una guía para estructurar, planear y controlar el proceso de desarrollo de una aplicación informática (Booch, et al., 2000). Estas están agrupadas en ágiles y tradicionales. La metodología Proceso Unificado de Desarrollo (RUP, en inglés Rational Unified Process), es una metodología tradicional, debido a que se centra en la planificación y control de los flujos de trabajo del proyecto en cuestión, obteniéndose una especificación detallada de los requerimientos necesarios para su desarrollo y modelado.

RUP es un proceso de desarrollo de software, pues es el conjunto de actividades realizadas por trabajadores que desempeñan un rol específico, para transformar los requisitos del usuario en un producto que cumpla con los mismos (Prince, 2005). Con este fin RUP divide el ciclo de vida del software en cuatro fases sobre las que se desarrollan varios flujos de trabajos, y cada fase termina con un hito, teniendo como resultado la disponibilidad de una secuencia de artefactos que pueden ser modelos o documentos que determinan el progreso del trabajo hasta alcanzar un estado de desarrollo predefinido. Para la solución propuesta se desarrollarán los modelos de diseño e implementación, como parte de los artefactos de los flujos de trabajo de Análisis y Diseño e Implementación correspondientes a las fases de Elaboración y Construcción respectivamente.

Entre las principales características que posee RUP se encuentra que es centrado en la arquitectura, pues esta involucra los elementos más significativos del sistema y está influenciada por otros factores como la plataforma en la que debe funcionar el software, sistemas operativos y consideraciones propias del desarrollo como requerimientos no

funcionales. Por lo que la arquitectura se representa mediante varias vistas que se centran en aspectos concretos del sistema, y que conforman el modelo 4+1 de la arquitectura, integrado por la vista lógica, de implementación, proceso y despliegue, incluyendo la de casos de uso que le proporciona cohesión a todas; lo que se representa a través de diagramas y modelos (Prince, 2005).

La selección de la metodología RUP para guiar el proceso de desarrollo del Sistema de Informatización de Tribunales, fue una decisión basada en varios aspectos, principalmente por ser una metodología robusta que se adapta a las características y complejidad del sistema. Genera gran cantidad de documentación, lo que se ajusta perfectamente al proyecto, el cual requiere una especificación en detalle de todo el trabajo realizado anteriormente, por su larga duración y el constante cambio del personal de desarrollo. Centra el proceso de desarrollo en la arquitectura definiendo para ella actividades y artefactos, provee calidad, reutilización de los componentes, seguridad y mantenimiento del software mediante una gestión sistemática de los riesgos.

1.4. Aplicación web para el desarrollo del SIT

Las funcionalidades definidas para el desarrollo del SIT deben implementarse siguiendo la arquitectura de desarrollo establecida por el equipo de arquitectura del proyecto TPC para la implantación de la solución informática, para lo que se concluyó que era necesario el desarrollo de una aplicación web.

Una aplicación web es un software o programa de cómputo que los usuarios utilizan accediendo a un servidor web a través de internet o de una intranet valiéndose del uso de un navegador web como Internet Explorer o Mozilla Firefox (Solano Vera, 2005). Con la aplicación de este recurso durante el desarrollo del SIT se cumplirá con los requisitos no funcionales definidos por el analista del sistema del proyecto TPC. En el caso de los requisitos de software, el sistema debe ejecutarse sobre un entorno web de manera que se permita su acceso desde un navegador Internet Explorer en la versión 6.0 o superior y Mozilla Firefox en la versión 3.0 o superior. También se obtendrán ventajas como que no ocupa espacio en el disco duro y consume pocos recursos, pues el usuario accederá a la aplicación a través de un navegador web. Esto es debido a que las tareas realizadas por el software provienen de otra computadora donde está ubicado el servidor web o de aplicaciones, el que interactúa con el servidor de base de datos; permitiendo que la

probabilidad de daño a los datos por algún virus informático u otra causa sea baja o casi nula. Este recurso posibilita lograr una administración centralizada y una fácil realización del despliegue, soporte y actualización, garantizando que esta última se realice a nivel central, es decir en el servidor, de tal forma que no sea necesario actualizar todos y cada uno de los clientes que acceden a la información del sistema (Productiva, Marzo 2011)**.**

1.5. Arquitectura de software del SIT

La arquitectura de software consiste en el diseño y especificación de un cierto número de elementos arquitectónicos para satisfacer los requerimientos funcionales y no funcionales que debe cumplir un sistema de software, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad; a fin de definir la línea que deben seguir los procesos relacionados con el desarrollo del software a través del uso de patrones y estilos arquitectónicos que tributen a la calidad del sistema (Camacho, y otros, Abril 2004). Permite a los desarrolladores e involucrados tener una idea clara de lo que se está implementando y debe ser diseñada por los arquitectos de software según las necesidades de las personas que van a utilizarlas. Una de las arquitecturas más comunes y la que se definió para el proyecto TPC, es la Arquitectura en Capas, específicamente la arquitectura en tres capas que está basada en un patrón de arquitectura conocido como Modelo Vista Controlador (MVC, en inglés Model View Controller).

1.5.1 Patrones de diseño y de arquitectura

Un patrón constituye una regla que consta de tres partes, la cual expresa una relación entre un contexto, un problema y una solución, logrando que esta última sea la óptima. Para que una de estas soluciones sea considerada un patrón debe ser reusable, es decir que su solución sea aplicable a diferentes problemas en distintas circunstancias. Entre la clasificación de los patrones se encuentran los patrones arquitectónicos y de diseño (Laman, Craig, 1999)**.**

- Patrones de diseño: Aquellos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software.
- Patrones de arquitectura: Aquellos que expresan el esquema organizativo estructural que es fundamental para sistemas de software.

1.5.1.1 Patrones de diseño

Entre los patrones de diseño se encuentran los patrones de diseño orientado a objetos, que forman parte del paradigma de la Programación Orientada a Objetos (POO). El diseño orientado a objetos implementado por este tipo de programación, pone en práctica la asignación de responsabilidades, es decir se distribuyen las funciones y las responsabilidades entre varios objetos de software en la aplicación, del mismo modo que se le asignan papeles a desempeñar a los trabajadores del negocio. Los objetos de software normalmente colaboran o interactúan para cumplir con sus responsabilidades, en analogía a como lo hacen las personas.

La descripción de la asignación de las responsabilidades y las interacciones de objetos a menudo se expresan gráficamente con diagramas de clase y con diagramas de colaboración o de secuencia. Los primeros muestran la definición de clases y los segundos el flujo de mensajes entre los objetos de software, razón por la que forman parte del diseño de la solución técnica propuesta por la presente investigación. La asignación de responsabilidades está englobada en un tipo de patrón nombrado Patrones Generales de Software para Asignación de Responsabilidades (GRASP, en inglés General Responsibility Assignment Software Patterns), entre los utilizados durante el desarrollo del SIT se encuentran:

- Experto: la asignación de la responsabilidad debe recaer sobre la clase que conoce toda la información necesaria para cumplir con la misma, como la creación de un objeto o la implementación de un método.
- Alta cohesión y Bajo acoplamiento: alta cohesión consiste en que la información que almacena una clase debe ser coherente y estar relacionada con ella de manera que esta tenga responsabilidades moderadas en un área funcional y sólo colabora con las otras clases necesarias para realizar las tareas que se le asignan. Una alta cohesión genera un bajo acoplamiento, este último plantea que una clase no depende de muchas clases solo las necesarias, por lo que al producirse una modificación tenga una repercusión mínima en las otras clases disminuyendo la dependencia entre estas.
- Controlador: es una clase a la que se le asigna la responsabilidad de atender un evento del sistema, ejerciendo como intermediario entre la lógica del negocio y la capa de presentación, recibe los datos del usuario, los procesa y los envía a las

clases según el método llamado. Para el desarrollo se implementa este patrón con la creación de un controlador por cada caso de uso[1], con el fin de dividir los eventos del sistema y aumentar la cohesión y disminuir el acoplamiento.

Otra clasificación de los patrones de diseño es el Grupo de los Cuatro (GOF, en inglés Gang of Four), que se dividen en grupos: Estructurales, de Comportamiento y Creacionales, de este último grupo se utiliza el patrón Singleton que garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella. Una clase Singleton tiene una variable estática (static) que hace referencia a la única instancia de la clase que se quiere utilizar, y que es creada cuando la clase es cargada en memoria. Para acceder a la única instancia de una clase Singleton, la clase proporciona un método estático (static), normalmente llamado getInstancia(), que retorna una referencia a la única instancia de la clase

El patrón Inversión de Control (IoC) es utilizado cuando se desea escribir clases que dependan de otras clases cuyas implementaciones no son conocidas en tiempo de compilación. Tiene el objetivo de integrar sistemas que tengan una única solución y mantengan comunicación e integridad de sus datos. También es empleado cuando se desee desacoplar las clases de sus dependencias de manera que las mismas puedan ser reemplazadas o actualizadas con muy pocos o casi ningún cambio en el código fuente. Una de las principales ventajas de usar el patrón IoC es que reduce el acople entre una clase y las clases de las cuales depende. Este patrón es utilizado para crear y brindar servicios. El módulo que implementa este patrón en el proyecto TPC es Administración y Gobierno, que se encarga de crear y brindar servicios comunes para que los restantes módulos los consuman. Aunque en el caso de que lo requiera, otro módulo también puede crear servicios y publicarlos.

1.5.1.2 Patrones de arquitectura

El MVC es un patrón de arquitectura de software que constituye una guía para el diseño de arquitecturas de aplicaciones que ofrezcan una fuerte interactividad con usuarios. Pertenece al estilo arquitectónico de llamada y retorno, porque este estilo se caracteriza

[1] Fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante.

porque: "*el sistema se constituye de un programa principal que tiene el control del sistema y varios subprogramas que se comunican con éste mediante el uso de llamadas"* (Camacho, y otros, Abril 2004). El funcionamiento del patrón MVC divide una aplicación interactiva en tres niveles:

- El modelo, que representa la información con la que trabaja la aplicación siguiendo las reglas del negocio.
- La vista, que muestra la información del modelo al usuario.
- El controlador, que maneja la lógica de negocio y toma las entradas del usuario, manipula el modelo, y hace que la vista se actualice para mostrar estos cambios.

Este patrón facilita la evolución por separado de la lógica de negocio y de la interfaz de usuario, lo que permite, que cuando se cambie la interfaz no se tengan que modificar necesariamente los componentes del negocio.

1.6. Marco de trabajo

Un marco de trabajo (en inglés framework) es un conjunto de bibliotecas y componentes centralizados en una aplicación o estructura de soporte definido, mediante el cual se puede organizar y desarrollar un proyecto de software (Frederick, y otros, 2008). Los objetivos principales que persigue un framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones de diseño y estilos arquitectónicos.

1.6.1 Sauxe

La Subdirección Tecnológica del Centro de Soluciones de Gestión de Entidades (CEIGE), de conjunto con la Unidad de Compatibilización, Integración y Desarrollo (UCID), desarrollaron el Marco de Trabajo Sauxe para desarrollar grandes aplicaciones web de gestión. Sauxe está estructurado por Doctrine, para el acceso a datos, ExtJs para la capa de presentación y ZendExt, que surge a partir de la extensión de algunos componentes de Zend Framework, para el manejo de la lógica del negocio a partir del patrón MVC (Tenrero Cabrera, y otros, 2011). Sauxe cuenta con un mecanismo para el manejo y configuración de excepciones, proporcionándole la información al usuario del error a través de la interfaz gráfica, la cual se registrará en un fichero de logs por cada excepción.

Este marco de trabajo provee un mecanismo de seguridad que permite:

- Restringir el acceso al sistema: cada usuario debe ser autenticado, de manera que se pueda determinar si tiene acceso al mismo y en caso de ser así, determinar las funcionalidades a las que tiene acceso y restringir su actividad al uso de las mismas, las cuales están determinadas por los roles asignados al usuario. Para autenticarse el usuario debe introducir nombre de usuario y contraseña.
- Definir una jerarquía de usuarios para el manejo centralizado de los mismos en el sistema: crear una jerarquía de usuarios, los cuales deben ser manejados por administradores de cada uno de los niveles. Las acciones que podrá realizar un usuario estarán determinadas por los roles que tendrá el usuario. Cada usuario pertenecerá a una entidad específica, existirán administradores en cualquier entidad que se desee.
- Permitir la creación de roles de usuarios: los roles serán creados por un administrador central y se podrán asignar a determinadas entidades de manera que los administradores de dichas entidades solo puedan crear usuarios con dichos roles. Los roles contendrán funcionalidades específicas del sistema que determinarán el acceso a las mismas por parte de los usuarios.
- Permitir la gestión y administración de usuarios en el sistema: el sistema permite la creación, modificación, activación y desactivación de los usuarios del mismo.

Se definió Sauxe como marco de trabajo porque es una alternativa muy económica para el desarrollo del sistema propuesto y posee toda la documentación y las funcionalidades de seguridad necesarias para dar cumplimiento a los requisitos establecidos con el cliente, con la calidad requerida y guiado por estándares internacionales. En el caso de la restricción del acceso al sistema permitirá definir que funcionalidades pueden realizar los usuarios que integran los procesos Disciplina y Derecho, la secretaria realiza la creación de los escritos de demanda y los jueces establecen la decisión sobre la misma. La creación de roles para usuarios según la entidad a la que pertenezcan permite que los procesos que conciernen a cada instancia se ejecuten en la entidad y en el nivel que corresponde.

1.6.2 ExtJs

ExtJs es una poderosa biblioteca construida con JavaScript desarrollada a través de objetos reusables, que ayuda a los desarrolladores a construir aplicaciones más atractivas. Su potencia radica en la rica colección de componentes personalizables que posee para el diseño, haciendo uso extensivo de la tecnología JavaScript Asíncrono y XML (Ajax) para el manejo de las peticiones al servidor, como cuadros de diálogo, menús, tablas editables, paneles y pestañas (Frederick, y otros, 2008). Principalmente es utilizada en aplicaciones que requieren un alto nivel de interacción con el usuario de manera más compleja que lo usual. Una de las grandes ventajas de utilizar ExtJs es que permite crear aplicaciones complejas utilizando componentes predefinidos así como un manejador de layouts[2].

1.7. Object Relation Mapper (ORM): Doctrine

Un ORM es un framework para convertir los datos que están almacenados en la base de datos relacional, en el que las tablas pasan a ser clases y los registros objetos que se pueden manejar con facilidad (Doctrine, 2009). Doctrine es el ORM que tiene definido el marco de trabajo Sauxe. Permite el trabajo con el lenguaje Preprocesador de Hipertexto (PHP, en inglés Hypertext Preprocessor) a partir de su versión 5.2.3, incorpora una capa de abstracción a base de datos y su característica más importante es la opción que brinda de escribir consultas a la base de datos en su propio lenguaje SQL orientado a objetos, llamado Doctrine Query Language (DQL). Este lenguaje provee a los desarrolladores una alternativa a SQL que evita crear consultas por concatenación y hacer frente a los ataques de inyección SQL**.**

1.8. Lenguajes de programación: PHP y JavaScript

Los lenguajes de programación son un "*conjunto de secuencias de instrucciones a través de un código que simboliza las operaciones a realizar*" (Ibarra, Agosto 2001). Estas instrucciones no son más que secuencias de símbolos y caracteres que cumplen con ciertas reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

[2] Capas que se utilizan para crear la estructura de una aplicación web.

PHP es un lenguaje interpretado de alto nivel embebido (incluido) en páginas que utilizan el Lenguaje de Marcado para Hipertexto (HTML, en inglés HyperText Markup Language) y ejecutado en el servidor. Soporta el uso de otros servicios que usen protocolos[3] como el Protocolo de Transferencia de Hipertexto (HTTP, en inglés HiperText Transport Protocol), permitiendo a los desarrolladores la generación dinámica de páginas. El código PHP se incluye entre etiquetas especiales de comienzo y final que permiten entrar y salir del modo PHP. A partir de PHP 5, se introduce el paradigma orientado a objetos en el lenguaje, además de ser gratuito, ligero y consumir poca memoria.

JavaScript es un lenguaje de programación del lado del cliente utilizado principalmente para crear aplicaciones web dinámicas. Es un lenguaje interpretado, por lo que no es necesario compilar los programas para ejecutarlos. Al utilizarlo de conjunto con el marco de trabajo ExtJs, incorporan acciones que se activan al pulsar botones y ventanas de alertas con mensajes de aviso. Esto permite que la interacción con el usuario se realice de la forma más amena posible, permitiendo que el sistema sea intuitivo y fácil de usar facilitando al máximo el trabajo por parte de los usuarios, pues estos presentan un nivel básico en conocimientos de computación (para el caso del SIT). Existen varias formas de incluir JavaScript, dentro de un documento HTML, una de estas es definir el código JavaScript en un archivo externo de tipo JavaScript, lo que provee una mejor organización, y enlazarlo mediante etiquetas <script> en los archivos HTML. Como principales ventajas que presenta este lenguaje: es dinámico, fácil de aprender, rápido, y orientado a objetos, pues permite trabajar con propiedades como las funciones del constructor.

1.9. Entorno de desarrollo integrado: Netbeans

Un Entorno de Desarrollo Integrado (IDE) es una herramienta que le permite a los programadores escribir, compilar, depurar y ejecutar programas (Corporation, Oracle, 2012). Pueden incluir varias funcionalidades en una misma suite como: un editor de código, un constructor de interfaz gráfica, un compilador y depurador de código e integración con sistemas controladores de versiones o repositorios. Pueden especializarse en el trabajo con un solo lenguaje de programación o con varios de ellos.

[3] Conjunto de reglas que controlan la secuencia de mensajes que ocurren durante una comunicación entre entidades que conforman una red.

Se definió como IDE Netbeans en su versión 6.9, porque es un entorno de desarrollo que permite la implementación con código PHP y posee un sistema para hacer reconocimiento y carga de clases, métodos y objetos. Permite la creación de aplicaciones web, propone una estructura para organizar el código fuente, es multiplataforma y el editor integra conjuntamente lenguajes como HTML, JavaScript, y un estilo llamado Hojas de Estilo en Cascada (CSS, en inglés Cascade Style Sheet), y PHP. Posee integración con Zend Framework, que es el framework base de la plataforma de desarrollo definido, y con sistemas de control de versiones, tales como Subversion, lo que es aprovechado para la realización de las actualizaciones diarias con el repositorio del proyecto TPC.

1.10. Software para los servidores

Para gestionar las funciones de lógica de negocio y de acceso a los datos durante el desarrollo de la aplicación, cada máquina cliente tiene la solución instalada en el servidor local (localhost). Esto significa que cada una actúa como un localhost de aplicaciones, al que se le envían las peticiones para su procesamiento, mientras que los cambios en la base de datos son administrados por un servidor de base de datos. Estos servidores requieren de un software específico para su uso.

1.10.1 Software para el servidor web: Apache

Un software para un servidor web es un programa informático que procesa aplicaciones en una red de computadoras, de manera que una aplicación del lado del servidor realiza conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente generando o cediendo una respuesta en un lenguaje o aplicación del lado del cliente.

Apache es un servidor web que implementa el protocolo HTTP, es software libre y soporta los sistemas operativos Windows y Linux, es de fácil configuración, pues su estructuración en módulos permite al usuario utilizar los servicios y funcionalidades que ofrece.

Se definió Apache como software para el servidor de aplicaciones en su versión 2.0, por su flexibilidad, rapidez, porque es gratuito, multiplataforma e interpreta varios lenguajes como PHP. Estas características promueven la eficiencia y rendimiento del sistema que debe ser capaz de dar respuestas a las peticiones con un nivel aceptable de desempeño,

teniendo en cuenta la concurrencia que pueda existir y debe prestar servicios sin que se amplíen los rangos de tiempo de repuesta.

1.10.2 Software para el servidor de base de datos: PostgreSQL

Un Sistema Gestor de Base de Datos (SGBD) es un tipo de software utilizado para los servidores de base de datos, que opera de interfaz entre la base de datos, el usuario y las aplicaciones que las utilizan. Su propósito es manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante para una organización, en este caso para los TPC. Los SGBD deben garantizar que esta información se encuentre segura, por lo que permiten otorgar diversas categorías de permisos a usuarios y grupos de usuarios. En cuanto a la independencia de los datos, se debe poder modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella (González Pérez, Febrero 2008)**.**

Las principales características que influyeron en la elección de PostgreSQL en su versión 8.3 como SGBD y como cliente el software PgAdmin III fueron:

- Alta concurrencia: mediante un sistema denominado Acceso Concurrente Multiversión (MVCC) permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos, esto permite el trabajo en conjunto entre los desarrolladores de cada módulo.
- Integridad de los datos: las claves primarias y llaves foráneas con capacidad de actualizar en cascada o restringir la acción y restricción no nula (not null), permiten el establecimiento de identificadores únicos que relacionen las tablas que lo necesitan, y la opcionalidad de campos vacíos para los datos que lo requieran.
- Resistencia a fallas y escritura adelantada de registros para evitar pérdidas de datos en caso de fallos por: energía, sistema operativo, hardware.
- Soporte para varios sistemas operativos entre ellos Linux y Windows.
- Capacidad para soportar cambios por su facilidad de mantenimiento, flexibilidad y facilidad de prueba.

- Permite definir niveles de acceso a la información en el sistema, que estará restringida de acuerdo a la entidad donde se originó y a las entidades rectoras de la misma.

1.11. Herramienta de modelado: Visual Paradigm

Una herramienta de modelado o de Ingeniería de Software Asistida por Computadora (CASE, en inglés Computer Aided Software Engineering) "*es un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación*" (Giraldo, y otros, 2005). Permite modelar los procesos de negocios que se desarrollan en una entidad determinada y facilitan la confección de artefactos y diagramas para la definición y recopilación de los requerimientos de los usuarios, el mejoramiento del diseño de los sistemas, y un mejor soporte en la documentación. Pueden estar enfocadas a un área de ingeniería de software más específica, como lo puede ser la ingeniería de información, el modelado de procesos, métricas, control de calidad, entre otros. Visual Paradigm es una herramienta CASE que como principales características posee: soporte UML a partir de su versión 2.1, generación del código a partir del modelo y viceversa, integración con la herramienta de control de versiones Subversion, generación de bases de datos a partir de la transformación de diagramas de Entidad-Relación en tablas de base de datos, ingeniería inversa de base de datos y distribución organizada automática de diagramas. Por estas razones y porque permite modelar todos lo diagramas como parte de los artefactos especificados por las fases de la metodología RUP para el ciclo de vida del software, se definió como Herramienta CASE a utilizar Visual Paradigm en su versión 3.4.

Las herramientas y elementos de programación establecidos anteriormente posibilitan el desarrollo de la aplicación partiendo de los requisitos funcionales y no funcionales que debe cumplir el sistema y que se establecieron con el cliente, por lo que es necesario que luego de su implementación, las funcionalidades del sistema sean probadas y validadas. Con este objetivo se hace necesario el empleo de ciertas métricas y/o mecanismos que posibilitan corregir errores y evaluar la calidad.

1.12. Métricas de software

Las métricas de software son utilizadas para la medición de elementos de suma importancia durante el proceso de ingeniería de software, proporcionan una visión útil de la calidad del software implementado. Esto es un proceso en el que se le asignan números o símbolos a atributos o entidades, con el fin de definirlas según reglas que ya están establecidas. Tiene como su principal finalidad construir software de alta calidad, lo que es definido como el "*cumplimiento de los requisitos de funcionalidad y desempeño explícitamente establecidos, de los estándares de desarrollo explícitamente documentados y de las características implícitas que se esperan de todo software desarrollado profesionalmente*" (Pressman, 2002)**.** Es necesario comprender los principios básicos de la medición, para poder obtener una evaluación con una óptima calidad. Por lo que el proceso de medición se caracteriza en cinco actividades fundamentales:

- Formulación: La derivación de medidas y métricas apropiadas para la representación del software que se considera.
- Recolección: El mecanismo con que se acumulan los datos necesarios para derivar las métricas formuladas.
- Análisis: El cálculo de las métricas y la aplicación de herramientas matemáticas.
- Interpretación: La evaluación de las métricas en un esfuerzo por conocer mejor la calidad de la representación.
- Retroalimentación: Las recomendaciones derivadas de la interpretación de las métricas de producto transmitidas al equipo de software.

La evaluación se realizará a partir de los datos obtenidos de los casos de prueba y del modelo de diseño. Las métricas que se utilizarán con este objetivo son:

Métricas para la validación del modelo de diseño:

Cuantifican los atributos del diseño de manera que permiten evaluar la calidad del mismo. Entre ellas se encuentran las métricas especializadas en el diseño orientado a objetos, cuya función consiste en medir las características correspondientes a la comunicación y la colaboración entre los objetos.

De esta colección el conjunto de métricas que más referencias poseen son las orientadas a clase o la colección de métricas de CK, que han sido propuestas por Chidamber y

Kemerer. Los autores han planteado seis métricas basadas en clases para sistemas como también se les conoce. De este conjunto las que serán empleadas son:

- Árbol de profundidad de herencia (APH): Esta métrica se basa en la longitud máxima desde el nodo hasta la raíz del árbol, por lo que a medida que crece el APH, las clases de nivel inferior heredarán más métodos y la complejidad del diseño será mayor. De igual manera un valor grande de APH indica que el nivel de reutilización de los métodos es alto.
- Número de descendiente (NDD): Un descendiente es una subclase que se encuentra inmediatamente subordinada a otra en la jerarquía de clases. Por lo que a medida que crece el NDD, se incrementa la reutilización y el número de pruebas requeridas para cada descendiente.

Otro conjunto de métricas para la evaluación del diseño orientado a objetos son las propuestas por Lorenz y Kidd, que separan las métricas basadas en clases en cuatro categorías: tamaño, herencia, valores internos y valores externos. Una de las más importantes y la que será aplicada es:

- Tamaño de Clase (TC): El tamaño general de una clase se determina a partir del número total de operaciones que están encapsuladas dentro de una clase y el número de atributos que están encapsulados por la clase.

Métricas para la validación de la calidad del producto:

Ayudan a diseñar casos de pruebas efectivos y a evaluar la eficacia de las pruebas. Son realizadas a partir de casos de pruebas:

- Métodos de caja blanca: se aplican en instrucciones de las operaciones definidas para una clase. Se realizan sobre un módulo concreto. También se les denomina estructurales, pues se centran en la definición de casos de prueba, para las instrucciones que conforman las funcionalidades de las clases, según el comportamiento interno de cada una.

- Método de caja negra: son realizadas sobre la interfaz de usuario y no se tiene en cuenta el funcionamiento interno del software, están orientadas a demostrar que las funciones del sistema son operativas, que los valores de entradas se aceptan adecuadamente y que se produce una salida correcta.

1.13. Conclusiones

Con la culminación de este capítulo se obtuvo como resultado que los sistemas de gestión procesal analizados presentan varias dificultades que no permiten cumplir con los requerimientos del SIT. Los extranjeros no se ajustan al sistema judicial cubano y los nacionales no responden a la materia Laboral. Las herramientas definidas por la arquitectura de desarrollo brindan al sistema flexibilidad, rapidez e integridad de los datos. El estudio de los patrones de diseño y de arquitectura permitió establecer una línea teórica, para su posterior uso en el diseño e implementación de la solución. Se definieron las métricas y/o mecanismos que se utilizarán posteriormente en la validación de la solución informática, lo que brindó una noción del procedimiento a seguir para alcanzar el objetivo de encontrar la mayor cantidad de errores, tanto en el diseño como en la implementación.

CAPÍTULO 2: SOLUCIÓN PROPUESTA

En el presente capítulo se profundizará sobre la estructura de la arquitectura del sistema y se ejemplificarán los patrones de diseño y estilos arquitectónicos utilizados. Se muestran el modelo de diseño, conformado por los diagramas de clases del diseño y de interacción. También se presenta el modelo de implementación del sistema y de despliegue, que forman parte de la especificación de los artefactos necesarios para la informatización de los requisitos identificados en los procesos Disciplina y Derecho de la materia Laboral en los Tribunales Populares Cubanos. Como principal resultado se evidencia la solución práctica de la investigación realizada.

2.1 Representación arquitectónica

Para la presentación de la arquitectura del sistema se utilizarán las vistas propuestas por RUP:

- Vista lógica: se presentarán la realización de los casos de uso a partir del modelo de diseño elaborado según los estilos arquitectónicos y patrones de diseño, para lo que se utilizan esquemas para representar gráficamente algunos conceptos predominantes en la estructura organizativa del sistema.
- Vista de implementación: se presentará el modelo de implementación conformado por el diagrama de componentes que especifica la relación de los componentes que integran la arquitectura del framework utilizado para el desarrollo.
- Vista de despliegue: se presentará el modelo de despliegue, conformado por la propuesta para el despliegue del sistema en las instancias de los tribunales.

2.2 Vista lógica

La vista lógica reúne los patrones de diseño y de arquitectura que proporcionan mayor solidez en la estructura del sistema y la reutilización de soluciones que permiten obtener resultados efectivos en el desarrollo del mismo; así como la estructura organizativa propuesta por Sauxe para la implementación del sistema. Esta vista incluye la descripción de las clases más importantes, su organización en paquetes y subsistemas, y la organización de estos subsistemas en capas. La descripción se realiza a través de

diagramas de clases para ilustrar la relación entre las clases arquitectónicamente significativas, paquetes y capas.

2.2.1 Estructura organizativa para la implementación del sistema

Teniendo en cuenta los principios establecidos por Sauxe (Figura 2), la estructura está basada en el modelo en tres capas (capa de presentación, capa de acceso a datos y capa de lógica del negocio). La misma define una organización jerárquica de manera tal que cada capa proporciona servicios a la capa superior y se sirve de las prestaciones que le brinda la capa inferior. Están integradas por elementos compuestos de varios paquetes o subsistemas. Según la materia, el módulo creado para su gestión posee la estructura necesaria para el desarrollo de las funcionalidades específicas de cada uno.

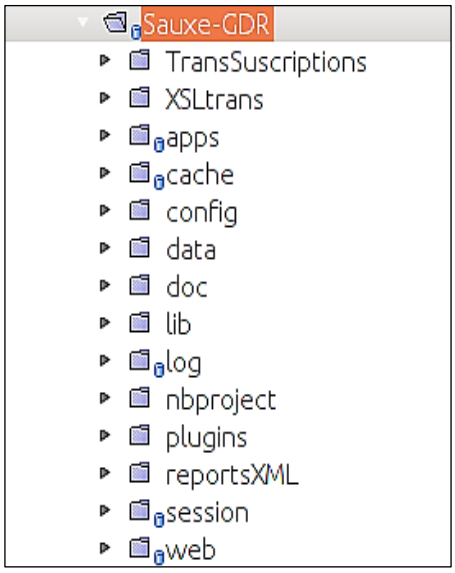

Figura 2: Organización propuesta por Sauxe

En la carpeta web (Figura 3) se controla la vista, específicamente en la carpeta views (vistas), se recopilan los ficheros que van a gestionar la capa de presentación:

- css: dentro de esta carpeta se encuentran las plantillas para el diseño del módulo, aunque ExtJs proporciona los componentes necesarios para la presentación.
- js: dentro de esta carpeta se encuentran los ficheros de extensión js donde se escribirá el código correspondiente a la capa de presentación, aquí es donde se carga el diseño de la aplicación, para cada clase controladora se crea una carpeta

que tendrá incluido el fichero js correspondiente a dicha clase control. En los casos en los que la complejidad del caso de uso a programar sea alta, se establece como una buena práctica de programación, a fin de lograr un rendimiento óptimo, la separación del código en varios archivos js que serán solicitados por la vista principal según el flujo a seguir.

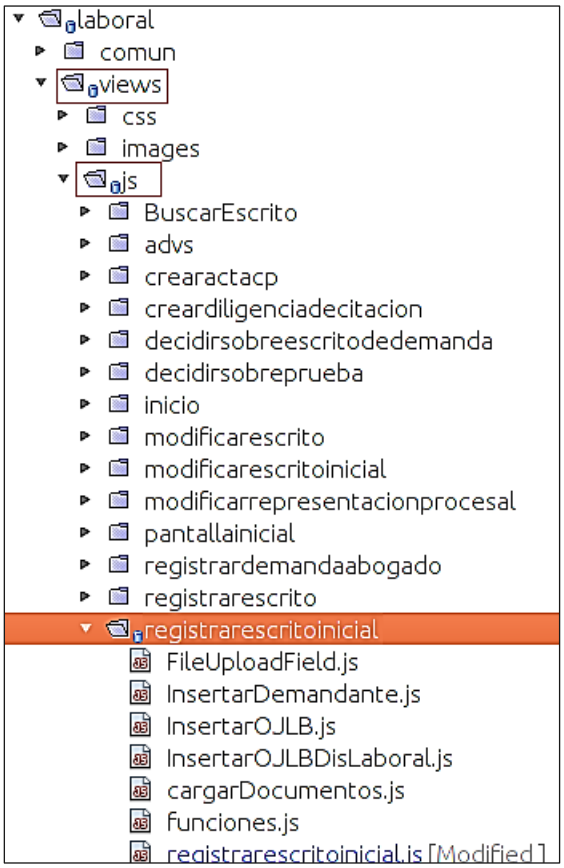

Figura 3: Estructura de la capa de presentación

En la carpeta apps (Figura 4) se maneja el control de la lógica del negocio y del modelo, así como la prestación y publicación de servicios, dentro de esta se encuentran las siguientes carpetas:

- común: contiene los ficheros xml de las excepciones del módulo (exception.xml) y para la publicación de los servicios creados (ioc-general.xml).
- controllers (Controladoras): contiene las clases que manejan el control de la lógica del negocio, en las que se programan las funcionalidades que son utilizadas

cuando el usuario hace alguna solicitud al sistema, a través de acciones sobre los componentes de la interfaz, como oprimir un botón o seleccionar un valor de una lista. Para las funcionalidades se emplea el sufijo Action después del nombre, por ejemplo registrarescritoinicialAction() y para las clases que las contienen, de extensión php, el sufijo Controller, ejemplo RegistrarEscritoInicialController.php.

- models (modelo): el acceso a los datos se realiza a partir de las tres clases generadas por el mapeador de datos Doctrine:
    - negocio (business), en las que se realizan los gestionar de los datos con las funcionalidades insertar, eliminar y modificar.
    - dominio (domain) donde se programan las consultas a la base de datos.
    - generadas (generated), que son las clases que reflejan la abstracción de las tablas de la base de datos y su relación con el resto a través de las llaves primarias que pasan a constituir llaves foráneas en caso de una relación de uno a muchos, o primarias en el caso de una relación de muchos a muchos conformando una nueva tabla.
- services: contiene los servicios que se utilizan, aunque el módulo Administración y Gobierno (A&G) es el encargado de la creación de los servicios que son utilizados en conjunto por el resto de los módulos, cada uno crea los servicios que requiera en la carpeta services y luego estos son publicados en su fichero ioc-general.xml, con un nombre que identifique la función que realiza el servicio.
- validators: esta carpeta contiene las clases de tipo php que van a realizar acciones de validación en el componente, como por ejemplo las precondiciones que se deben cumplir antes que un determinado método sea ejecutado.
- views: contiene los archivos de extensión phtml donde se especifica el título de la página que se gestiona y se carga el archivo js que mostrará la presentación y se especifica el orden en que estos serán utilizados pues la presentación generalmente se segmenta en varios js, para optimizar la organización del código.

Figura 4: Estructura de la capa de control de la lógica del negocio y del modelo

2.2.2 Ejemplificación del uso de los patrones de diseño

Los patrones de diseño fueron utilizados para lograr un diseño e implementación eficaz del software orientado a objetos según la correcta asignación de responsabilidades a las clases según determinada categoría de problemas, y para la interacción de los módulos que estructuran el sistema.

- **Patrón Experto**

El patrón Experto fue utilizado como parte del diseño de la solución para delimitar las funcionalidades que debe cumplir cada clase, según la responsabilidad que posea. Para lo que a menudo se requiere que la información se encuentre distribuida en varias clases de objetos. Este patrón permite conservar el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Para su utilización en el diseño del caso de uso Registrar Prueba, es necesario conocer la cantidad de pruebas y los tipos de pruebas que presentan. Para lo que alguna de las clases (Figura 5) deberá contener la información para obtener los datos que se necesitan. Asignando las responsabilidades con una definición clara de cada clase se plantea la pregunta:

¿Quién es el responsable de conocer la cantidad de pruebas y los tipos de pruebas?

Desde el punto de vista del patrón Experto, la clase de objetos que posee la información necesaria es LabNprueba, que contiene la funcionalidad buscarTipoprueba() en la que se realiza la consulta en el lenguaje DQL a la Base de Datos.

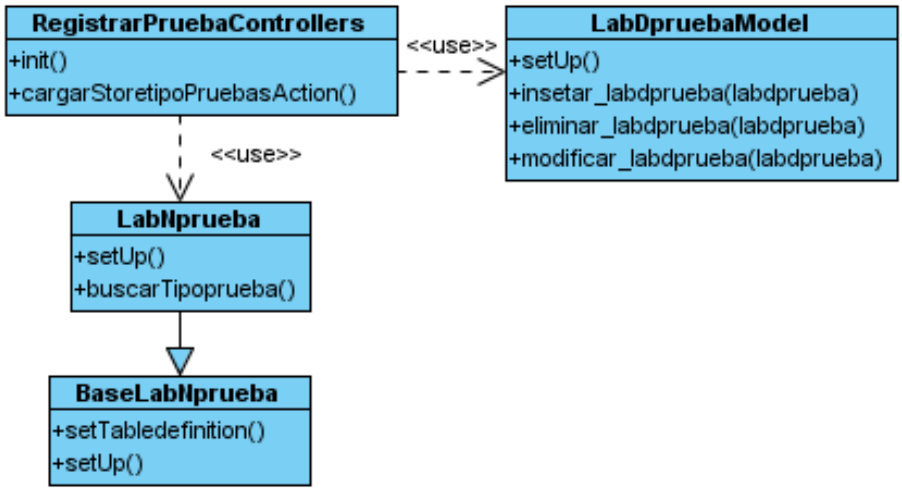

Figura 5: Patrón Experto para el diseño orientado a objetos

En general para cumplir con la responsabilidad de conocer toda la información y gestionar los datos de las pruebas, se asignaron cuatro responsabilidades a las cuatro clases de objetos de la siguiente forma:

Clase	Responsabilidad
RegistrarPruebaControllers	Conoce la clase js que hace la petición de la información
LabDpruebaModel	Conoce cómo gestionar los datos de las pruebas
LabNPrueba	Conoce los tipos de pruebas
BaseLabNPrueba	Conoce los atributos para la relación con otras tablas

Tabla 1: Asignación de responsabilidades en el patrón Experto

- **Patrones Alta Cohesión y Bajo Acoplamiento**

El patrón Alta Cohesión fue utilizado para evitar que una clase realice demasiadas operaciones según las tareas que se le asignan, para lo que se caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.

Una forma de lograr la utilización correcta de este patrón es a través de la creación de varias clases con extensión js, para separar el código implementado. El patrón Bajo Acoplamiento es utilizado para evitar que una clase dependa de otras de manera innecesaria, lo que permitirá que sean fáciles de entender por separado y de reutilizar. En este caso se logra a partir de la obtención de una subclase directa, lo que se representa en la Figura 6 con la herencia de la clase LabNprueba de BaseLabNprueba.

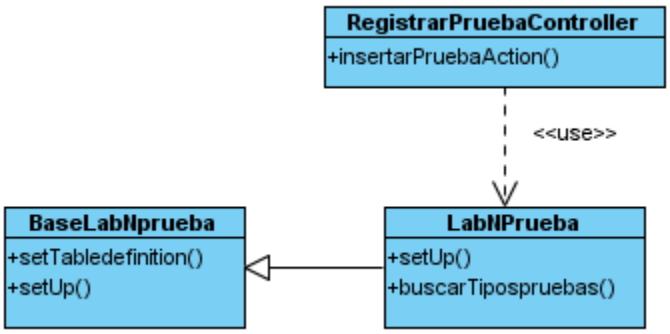

Figura 6: Patrón Bajo Acoplamiento para la programación orientada a objetos

- **Patrón Controlador**

El patrón Controlador fue utilizado a partir de la facilidad que brinda el framework para crear un controlador por cada caso de uso, el cual se encargará de manejar todas la acciones que requiera el caso de uso a partir de la funcionalidades que lo conforman (Figura 7).

```
class RegistrarescritoinicialController extends ZendExt_Controller_Secure {

    function init() { }

    function cancelarCompletoAction() { }

    function pasaraDefinitivoAction() { }

    function formarDocumentoAction() { }

    function insertarEscritoinicialAction() { }
}
```

Figura 7: Patrón Controlador para la programación orientada a objetos

- **Patrón IoC**

El patrón de Inversión de control se empleó para la utilización de los servicios publicados provenientes de otros módulos, con el fin de lograr un menor acoplamiento entre los mismos y permitir la reutilización de las funciones comunes. Cada módulo del proyecto cuenta con un esquema propio en la base de datos, por lo que cuando un módulo necesita acceder al esquema de otro, crea y publica un servicio para que el módulo que lo necesite lo consuma, por los que estos son publicados en el fichero ioc-general.xml de cada módulo.

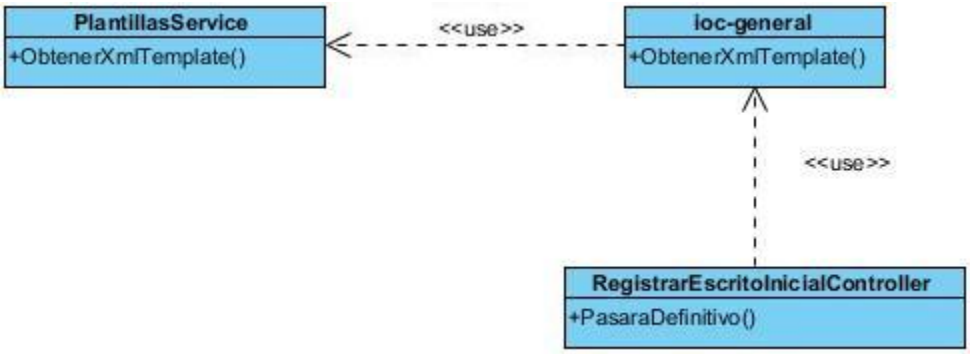

Figura 8: Diagrama de clases para el patrón IoC

En la Figura 8 se utiliza el servicio ObtenerXmlTemplate, publicado en el ioc-general.xml en la función PasarDefinitivo() de la clase RegistrarEscritoInicialController, para guardar en la base de datos definitivamente los documentos que se generan, es decir para que estos no se puedan cambiar nuevamente.

- **Patrón Singleton**

Para la utilización del patrón Singleton en la Figura 9 se representa la clase ZendExt_Event, que tiene como atributo una instancia que se declara privada con el objetivo de que dicha clase sea la única que tenga permisos para modificarla. También contiene la funcionalidad getInstance(), en la que se verifica si esta instancia ha sido creada en el momento en que se solicitó la función, si no ha sido creada se crea y se retorna, en caso contrario solamente se retorna. Mientras que en la clase RegistrarEscritoInicialController, a partir de la funcionalidad insertarEscritoinicialAction(),

se solicita el método getInstance() para obtener los servicios publicados de un módulo. La utilización de este patrón en la implementación del sistema permitió que los usuarios que poseen un conjunto de configuraciones diferentes según el nivel de acceso que tengan al sistema, puedan acceder de manera sincronizada a la clase que brinda un servicio común para todos.

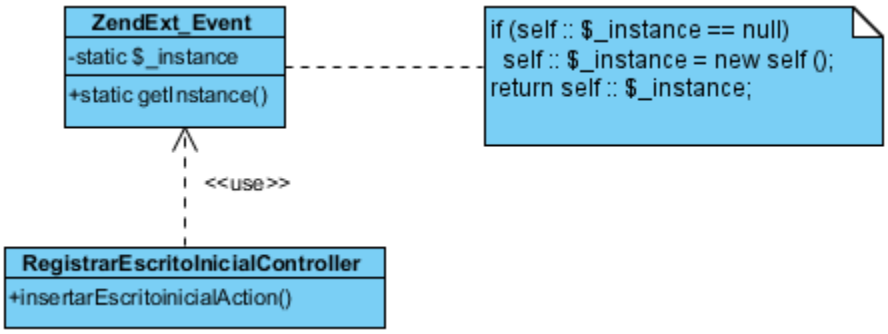

Figura 9: Diagrama de clases del patrón Singleton para obtener una instancia única

2.2.3 Modelo de diseño

El modelo de diseño es un modelo de objetos que describe la realización de los casos de uso, y proporciona una abstracción del modelo de implementación y su código fuente. Este modelo se utiliza como entrada a las actividades de los flujos de trabajo Implementación y Prueba. Está integrado por los diagramas de clase del diseño y por los diagramas de interacción, que forman parte del modelo de diseño del subsistema Laboral del Anexo 1 de la solución propuesta por el presente trabajo.

2.2.3.1 Diagrama de clases del diseño

Un diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces de la aplicación, brinda una perspectiva desde el punto de vista del diseño de las entidades de software. Para su confección es necesario identificar todas las clases que participan en la solución del software y conocer la información sobre los métodos y tipos de atributos de cada una. Para las Aplicaciones Web específicamente los diagramas de clase del diseño se conforman a partir del uso de estereotipos web. A continuación se muestra una descripción de un caso de uso de prioridad alta dentro del sistema:

Caso de Uso:	Registrar Escrito Inicial
Actores:	Secretaria (inicia), juez ponente
Resumen:	El caso de uso se inicia cuando la Secretaria o Registrador de escrito necesita registrar un escrito inicial. Consiste en que el Registrador de escrito selecciona la opción Registrar demanda. El caso de uso termina con el escrito inicial registrado.
Precondiciones:	▪ La secretaria o el abogado hayan sido autenticados en el sistema
Referencias	RF.01, RF.15, RF.38, RF.39, RF.40, RF.41, RF.56, RF.63, RF.64, RF.83, RF.84 CU Registrar datos del representante (Extendido) CU Registrar pruebas (Extendido) CU Registrar tercero posible afectado(Extendido)
Reglas del negocio	13, 22
Prioridad	Crítico

Tabla 2: Descripción del caso de uso "Registrar Escrito Inicial"

El Diagrama de clases del diseño correspondiente a la descripción del caso de uso Registrar Escrito Inicial muestra las clases principales que intervienen en el flujo principal del proceso, en el que resulta la inserción en la base de datos del escrito que inicia el proceso y que forma parte del expediente al que responde el mismo.

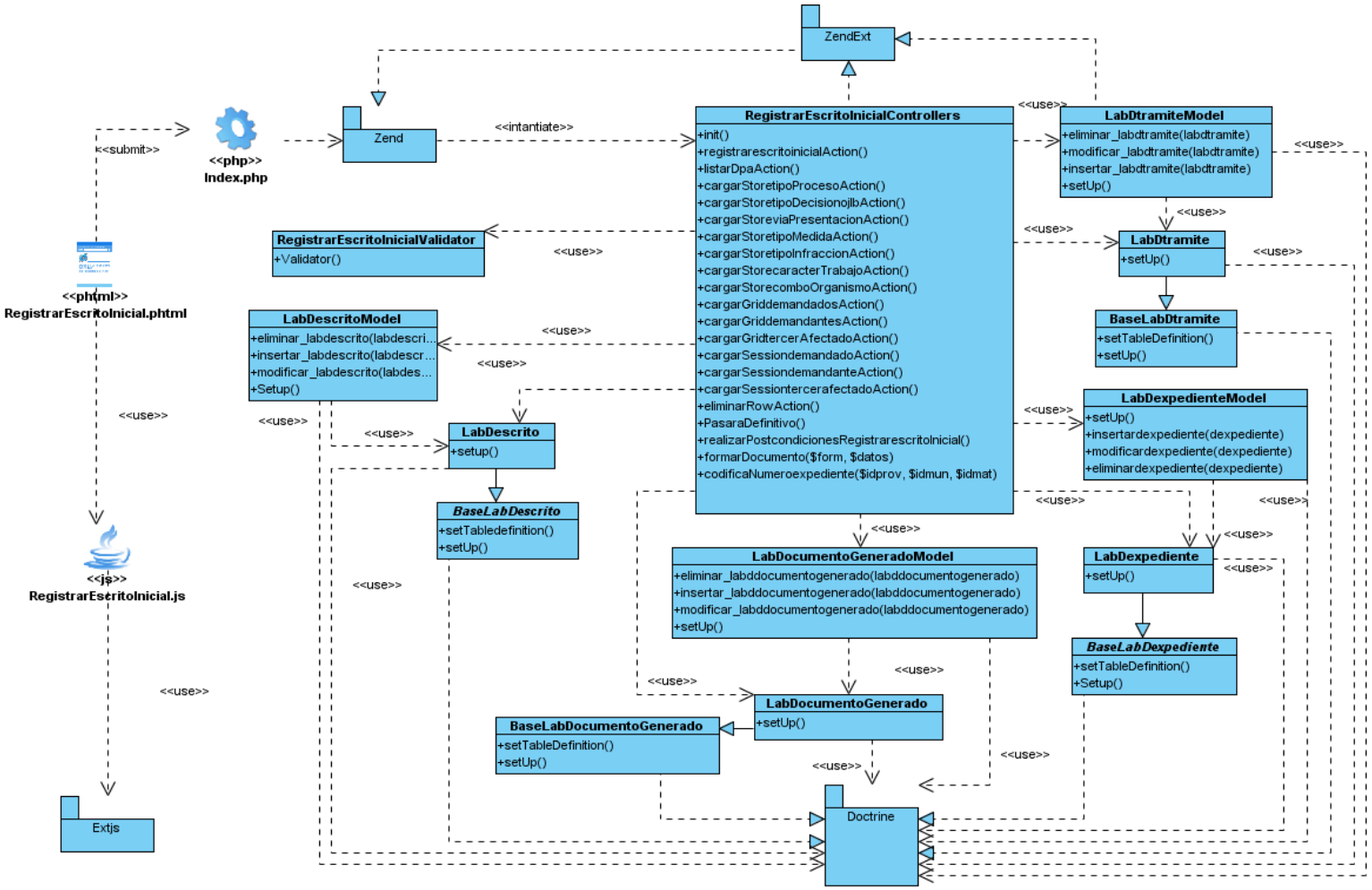

Figura 10: Diagrama de clase del diseño "Registrar Escrito Inicial"

Las clases que conforman el diagrama anterior son las siguientes:

- RegistrarEscritoInicial.js: Fichero js que utiliza los componentes ofrecidos por la librería ExtJs y el lenguaje JavaScript para crear la vista con la que interactúa el usuario. Obtiene los datos que necesita a través de peticiones Ajax que realiza al controlador, en la que especifica el tipo de método utilizado, “POST” o “GET” para enviar lo datos, los parámetros que requiere, el nombre de la funcionalidad que llamará, la acción que realizará al tener éxito y la que realizará si falla, esta última puede ser un mensaje indicándole al usuario la causa.

- RegistrarEscritoInicialControllers: Contiene todas las funcionalidades que debe realizar el caso de uso, hereda del componente ZendExt_Controller_Secure todos sus atributos y funcionalidades incluyendo la función init(), la que es redefinida. Cada controlador responde al menos a una vista o página cliente, una vez que se solicite la visualización de alguna funcionalidad del sistema, la petición llega hasta el controlador que le brinda una respuesta a la misma. El controlador contiene un método que lleva el mismo nombre, pero con letras minúsculas, que contiene la función render() que se encarga de solicitarle al componente Zend_View que muestre la vista solicitada.
- BaseLabDescrito: Tiene la función setTableDefinition() que define las relaciones con el resto de las clases. Pertenece a las clases generated y es una de las generadas por el ORM Doctrine.
- LabDescrito: En esta clase se deben realizar las consultas, pero por cuestiones de optimización se realizan en la clase que hereda de esta LabDescritoExt (esta clase no se representa en el diagrama), para evitar el traslado del código cada vez que se mapeen las clases de la base de datos. Pertenece a las clases domain y es una de las generadas por el ORM Doctrine.
- LabDescritoModel: Realiza las operaciones principales para gestionar (eliminar, modificar, insertar), teniendo como parámetro un objeto de tipo de la clase LabDescrito (para el caso que se requiera gestionar un escrito). Pertenece a las clases business y es una de las generadas por el ORM Doctrine.
- RegistrarEscritoInicial.phtml: Define el orden de los js y es usado por el componente Zend_View para representar la vista que se le muestra al usuario.
- Index.php: Funciona como un controlador frontal entre el archivo phtml y el paquete de componentes de Zend, específicamente tiene una relación de *use* con el componente Zend_Controller_Front.
- ExtJs: Paquete que representa los componentes del framework o librería ExtJs, los cuales son utilizados para crear la vista que se le muestra al usuario.
- Zend: Es el paquete de componentes de Zend Framework que utiliza Sauxe, para el manejo de la dinámica entre las vistas y la lógica del negocio. Está integrado por los componentes Zend_View, Zend_Controller_Action, Zend_Controller_Front, Zend_Loader.

- ZendExt: Paquete de componentes que utiliza Sauxe internamente creado a partir de la extensión de algunos componentes de Zend, está integrado por los componentes ZendExt_Controller_Secure y ZendExt_Model. La clase controladora y las clases que tienen terminación Model heredan del ZendExt.
- Doctrine: Está conformado por los componentes que utiliza el ORM Doctrine, Doctrine_Query y Doctrine_Record. Las clases con prefijo Base poseen una herencia del componente Doctrine_Record, mientras que las clases del dominio utilizan Doctrine_Query para la realización de consultas en el lenguaje DQL.

2.2.3.2 Diagrama de interacción

Los diagramas de interacción explican gráficamente cómo interactúan los objetos a través de mensajes para realizar las tareas, existen dos tipos, los diagramas de colaboración y los de secuencia. Los diagramas de secuencia muestran la organización que deben tener estos objetos, lo que depende en gran medida de la arquitectura de desarrollo y de los patrones utilizados, de manera que la comunicación entre sus elementos mantenga una lógica fundamentada en la relación que se establece entre los mismos. Se elaboran a partir de las descripciones de los casos de uso en las que cada evento en el que el usuario realiza una solicitud al sistema, ocasiona una acción determinada como respuesta. Cada respuesta es procesada por los componentes y clases que conforman las capas de la arquitectura, en la que cada una cumple su función. En el diagrama de secuencia correspondiente al caso de uso descrito se representará solamente uno de los escenarios que se realizan.

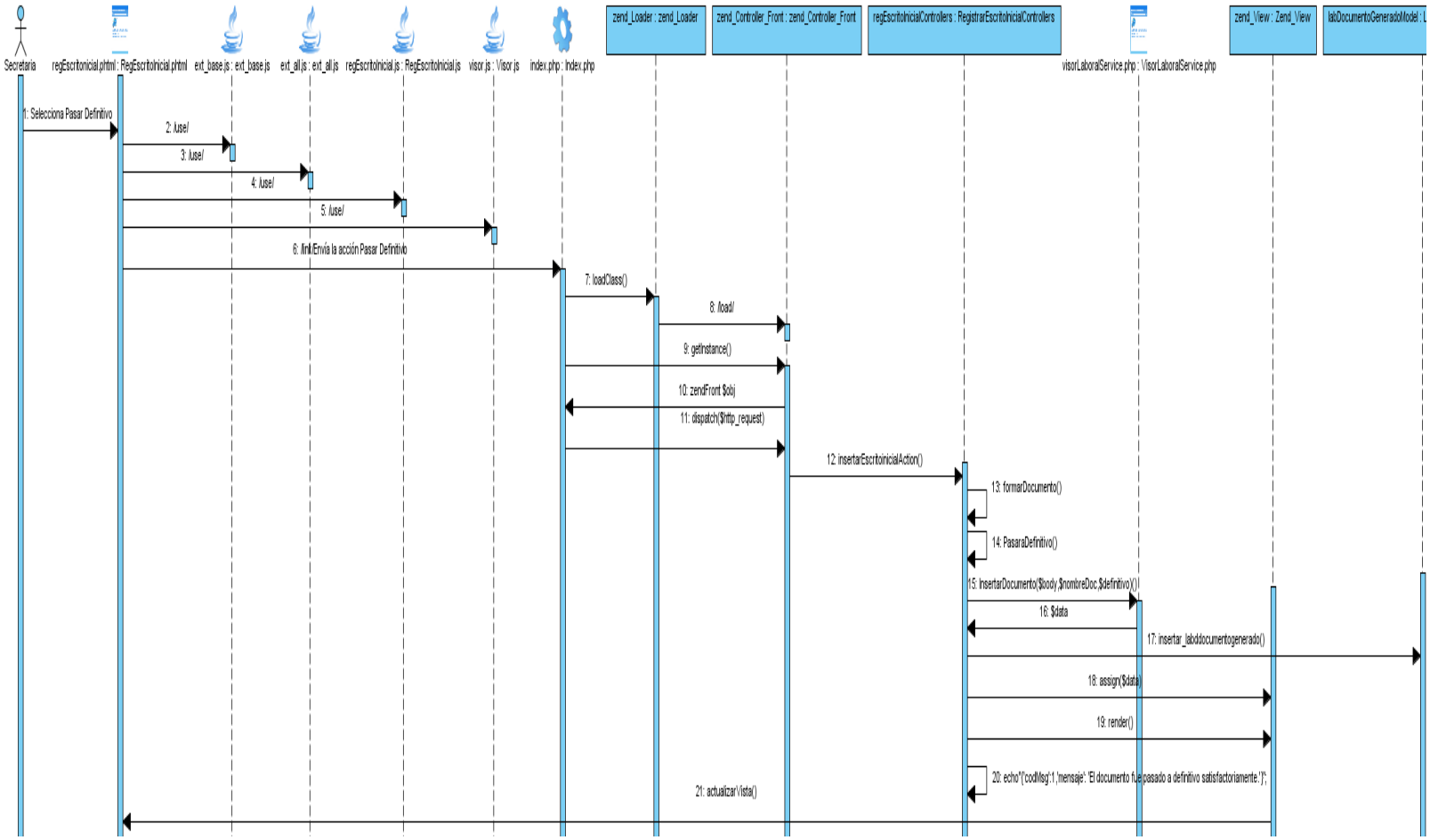

Figura 11: Diagrama de secuencia "Registrar Escrito Inicial", para el escenario Pasar a Definitivo

Entre los elementos del diagrama anterior se incluye el actor del sistema que da inicio a la acción del caso de uso, que en este caso es la secretaria, de esta forma se inicia el flujo básico de operaciones. Se representa la secuencia de acciones para guardar el escrito inicial como un documento en la base de datos y pasarlo a definitivo, para que no se le puedan realizar más cambios. Esto se realiza a partir de la utilización de la clase VisorLaboralService.php, que forma parte de los servicios que se crean en el módulo para la prestación de servicios comunes para todos los casos de usos. La funcionalidad que se creó para insertar el documento y se publicó en el fichero ioc_general.xml se denomina InsertarDocumento(), y lleva como parámetros el cuerpo del documento que está conformado por la plantilla con los datos necesarios, el nombre del documento y si es definitivo y o no, en este caso sería porque se estaría conformando el documento para

visualizarlo primeramente antes de pasarlo a definitivo. Finalmente se muestra un mensaje con el texto “El documento fue pasado a definitivo satisfactoriamente”. De esta misma forma se ejecutan todas las peticiones que realice el usuario al sistema y las que este necesite internamente, desde la capa de presentación, pasando por la controladora hasta la capa de acceso a datos.

2.3 Vista de implementación

La vista de implementación se constituye de una colección de componentes y subsistemas de implementación mediante el modelo de implementación. En esta vista por lo general se definen ejecutables, ficheros, subsistemas, y las dependencias entre ellos asociados a cada una de las capas que conforman el sistema, según la arquitectura establecida para el mismo. Para este caso se distinguen la capa de presentación, la capa de negocio, y la de acceso a datos.

2.3.1 Modelo de implementación

El modelo de implementación está integrado por el diagrama de componentes, que se conforma a partir de los componentes de la arquitectura. Este modelo constituye uno de los artefactos fundamentales resultante del flujo de trabajo de Implementación correspondiente a la fase de Construcción de RUP. Su principal objetivo es proporcionar una visión general de lo implementado en el sistema.

2.3.1.1 Especificación de los componentes de Sauxe

Los componentes que originaron el surgimiento de ZendExt y que constituyen la base del núcleo del framework Sauxe son ocho de los 51 que posee Zend Framework, posibilitan la obtención de una poderosa aplicación, con los requerimientos y capacidades necesarias que debe cumplir todo sistema, principalmente de gestión gubernamental como es el caso del SIT. Sauxe es una integración de estos componentes, en conjunto con los de la librería ExtJs y los del ORM Doctrine, por lo que fueron utilizados en la elaboración de la solución propuesta por el presente trabajo, específicamente en el modelo de implementación como parte de los componentes que se representan en el diagrama de componentes. A continuación se muestra una relación de los componentes utilizados con este fin y una breve descripción de su funcionalidad (Andux, Junio 2010):

- Zend_View: Es la clase que permite trabajar con la vista en el patrón MVC. El uso de esta clase, ocurre en dos grandes pasos: su controlador de secuencia de comandos crea una instancia de Zend_View y asigna variables a esa instancia, luego el controlador le indica al Zend_View que emita una vista, y devuelve el control al script de la vista, lo que genera la vista saliente. Su principal responsabilidad es mantener el script de la vista separado de los scripts del modelo y de los controladores e identificar cual es la vista o phtml que se está instanciando, una vez identificada procede a su ejecución.
- Zend_Controller_Front: Todas las peticiones que se ejecutan en un subsistema son validadas por el controlador frontal, el mismo contiene un método llamado init() que implementa un modelo de controlador frontal usado en aplicaciones que utilizan la arquitectura MVC. Su propósito es inicializar el entorno de la solicitud, rutear la solicitud entrante, y luego hacer un envío de las acciones; le agrega las respuestas y las regresa cuando se completa el proceso.
- Zend_Loader: Garantiza que a partir de una ruta de inclusión, este sea el responsable de la inclusión de los recursos requeridos en el proceso.
- Zend_Controller_Action: Es una clase abstracta que se utiliza al implementar controladores de acción para usar con el Controlador Frontal. La operación más elemental es hacer una subclase, y crear métodos de acción que corresponden a las posibles acciones que se desee que el controlador maneje, que serán cualquier método que termine en 'Action'.
- ZendExt_Controller: Permite lograr que todas las peticiones sean procesadas únicamente por un archivo index.php, lo que ofrece un punto central para todas las páginas de la aplicación y asegura la instalación de un ambiente correcto para ejecutar la aplicación.
- ZendExt_Validation: Este componente debe validar todos los datos y acciones que se activan ya sean por un usuario o por un sistema externo, que serán ejecutadas en un controlador de una aplicación en específico. La acción será validada antes de llegar al controlador para no hacer peticiones innecesarias a este.
- Doctrine_Query: Representa una consulta DQL que se utiliza para consultar bases de datos en una forma orientada a objetos.

- ext-base.js: Componente encargado del manejo de las solicitudes y respuestas para el trabajo con Ajax y el manejo de los componentes de ExtJs. Está incluida dentro de las clases que trae ExtJs.
- ext-all.js: Componente encargado de la creación de los componentes visuales de la vista. Está incluida dentro de las clases que trae ExtJs.

2.3.1.2 Diagrama de componentes

Un componente constituye una parte modular de un sistema, desplegable y reemplazable, que encapsula la implementación y un conjunto de interfaces que proporciona la realización de los mismos. Un diagrama de componentes es utilizado para reflejar la relación que se establece entre los componentes y/o elementos que conforman la arquitectura de un sistema de software, su estructura y la dependencia entre los mismos.

El diagrama de componentes para la materia Laboral específicamente Figura 12, en el que se presentan los componentes que conforman la estructura interna del framework Sauxe integran las capas que conforman la arquitectura MVC, pues son utilizados por cada una para realizar las funcionalidades correspondientes y permiten que la interacción entre las mismas se realice de forma integrada y dependiente:

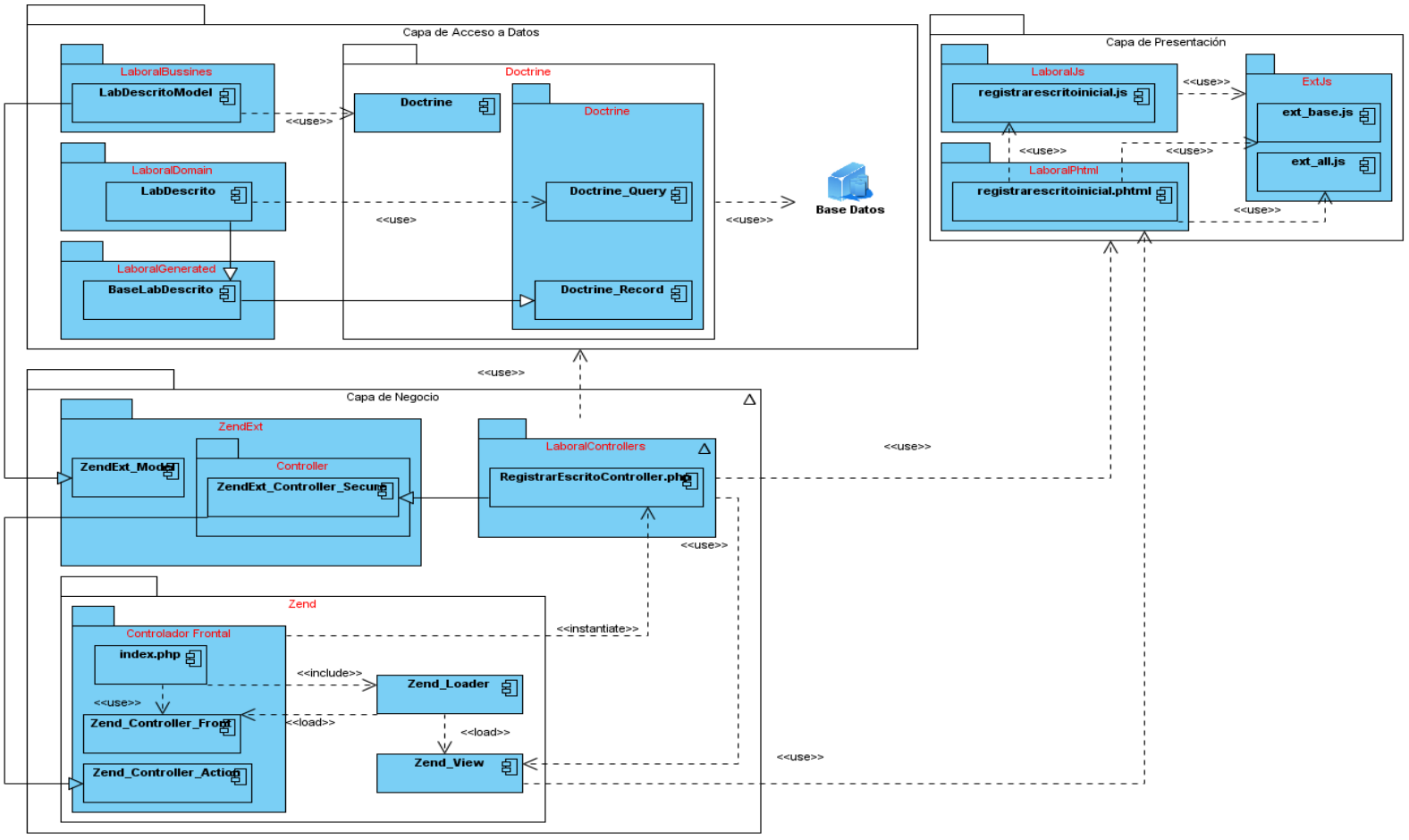

Figura 12: Diagrama de componentes de Sauxe de la materia Laboral

2.4 Vista de despliegue

La vista de despliegue se elabora a partir del modelo de despliegue, representa la disposición de las instancias de componentes de ejecución en instancias de nodos, se elabora a partir de la descripción de los nodos físicos para la mayoría de las configuraciones tanto para usuarios finales como para desarrolladores y probadores.

2.4.1 Modelo de despliegue

El modelo de despliegue está integrado por el diagrama de despliegue, este modelo define la arquitectura física del sistema por medio de nodos interconectados. Estos nodos son elementos de hardware sobre los que pueden ejecutarse elementos de software, permitiendo conocer una aproximación de cómo será la arquitectura física del sistema

antes de comenzar su desarrollo. Permite dar una vista arquitectónica más visible al modelo de diseño del sistema propuesto.

2.4.1.1 Especificaciones para el despliegue del sistema

Para la elaboración del diagrama de despliegue se consideraron los requisitos de hardware que requiere el sistema para su implantación; así como la estructura organizativa de cada instancia de los tribunales, por lo que es necesario garantizar que el sistema se despliegue en todos los municipios y provincias del país. El diagrama de despliegue que se presenta deberá cumplir con las especificaciones siguientes:

Se requiere de varias computadoras Clientes, en las que se procesará toda la información que se genera como parte de la tramitación de los procesos según la instancia y la materia, donde al menos una estará conectada a una impresora por el puerto bus universal en serie (USB, en inglés Universal Serial Bus) de la máquina para la impresión de los documentos. Por cada instancia el sistema contará con un servidor de base de datos y con un servidor Web Apache2. Este último ejecutará la función de servidor de aplicación, y su configuración en clúster permitirá dividir la carga de peticiones y aumentar la latencia a fallos del sistema permitiendo que si un servidor presenta algún problema otro pueda hacerse cargo de las peticiones. Para esto se establece un sistema de réplica de sesiones, que estará implementado a nivel del centro de datos. Para guardar los datos que requieran gran capacidad, como pruebas visuales, se instalará en las instancias provinciales y supremas un servidor que utiliza el Protocolo de Transferencia de Archivos (FTP, en inglés File Transfer Protocol), que se conectará con el servidor de BD.

Teniendo en cuenta la cantidad de usuarios conectados concurrentemente, así como la cantidad de procesamiento del mismo y con el objetivo de lograr mayor rendimiento y disponibilidad de este en la base de datos, se concibe un sistema para el centro de datos instalado en un clúster formado por cinco servidores.

La composición del clúster estaría formada por tres computadoras o nodos de base de datos, que se comunicarán a través de una red privada virtual (VPN), un servidor balanceador de carga, y un servidor de replicación. La base de datos estará distribuida en tres computadoras o nodos, que son las máquinas que procesan las demandas reales que entran. Siempre que una demanda se reciba del cliente, el clúster de base de datos

tiene que determinar si está enfrentando una demanda de lectura o escritura. En adición al acceso de lectura se encuentra el servidor balanceador de carga que asegura que todos los nodos del banco de datos puedan trabajar eficazmente al mismo tiempo, pues se utiliza para distribuir la carga dentro del conjunto de nodos, la máquina con el número más bajo de preguntas activas será escogida para realizar una nueva demanda. El servidor de la replicación es el componente central del sistema, pues toma demandas entrantes y copia esos cambios a todos los nodos del banco de datos en el sistema. Si el servidor de replicación descubre un problema en un nodo del banco de datos elimina la máquina de la lista de banco de datos activos y se escribe al disco un log que contiene una descripción detallada del error (Productiva, Octubre 2010).

Los requerimientos de hardware definidos por la arquitectura de desarrollo para el correcto funcionamiento del sistema consistirán en:

- Las máquinas clientes, serán clientes ligeros que utilizarán el sistema operativo Linux en la distribución Debian, con al menos 512 MB de memoria RAM.
- Los servidores de aplicación de las instancias del tribunal poseerán Micro Dual Quad Core Xeon E5440 con 12 megas de cache, 8GB de memoria RAM y 40GB de disco duro y usarán el sistema operativo Linux en la distribución Ubuntu.
- Para los servidores de base de datos y del centro de datos se requiere que posean microprocesador Core I3, 2 GB de memoria RAM, 1 terabyte de disco duro y el sistema operativo Linux en la distribución Ubuntu.

2.4.1.2 Diagrama de despliegue

El Diagrama de despliegue describe la distribución física del sistema en términos de cómo se distribuyen las funcionalidades entre los nodos de cómputo. Representa la correspondencia entre la arquitectura del software y la arquitectura del sistema y muestra la configuración de los nodos de proceso en tiempo de ejecución y los enlaces de comunicación entre estos.

De manera general constituye un grafo de nodos unidos por conexiones de comunicación. Un nodo puede contener varias instancias de componentes de software, objetos o procesos. En general un nodo será una unidad de computación de algún tipo, desde un sensor a un servidor. El Diagrama de despliegue representa la interacción de los componentes que según su funcionalidad establecen relaciones de conexión a nivel

de protocolos, que para el caso de la relación de la computadora cliente y el servidor (PC_cliente – servidor) será el protocolo HTTP, mientras que en la conexión entre servidores se utilizará el Protocolo de Control de Transmisión/Protocolo de Internet (TCP/IP, en inglés Transmission Control Protocol/Internet Protocol). Dado el carácter institucional de los TPC y su estructuración en forma jerárquica, se determinan tres escenarios de despliegue en el que se debe instalar el sistema, además del centro de datos: instancia municipal (incluido Isla de la Juventud), instancia provincial e instancia suprema, por lo que se presentará la integración de los escenarios mencionados a nivel nacional (Figura 13).

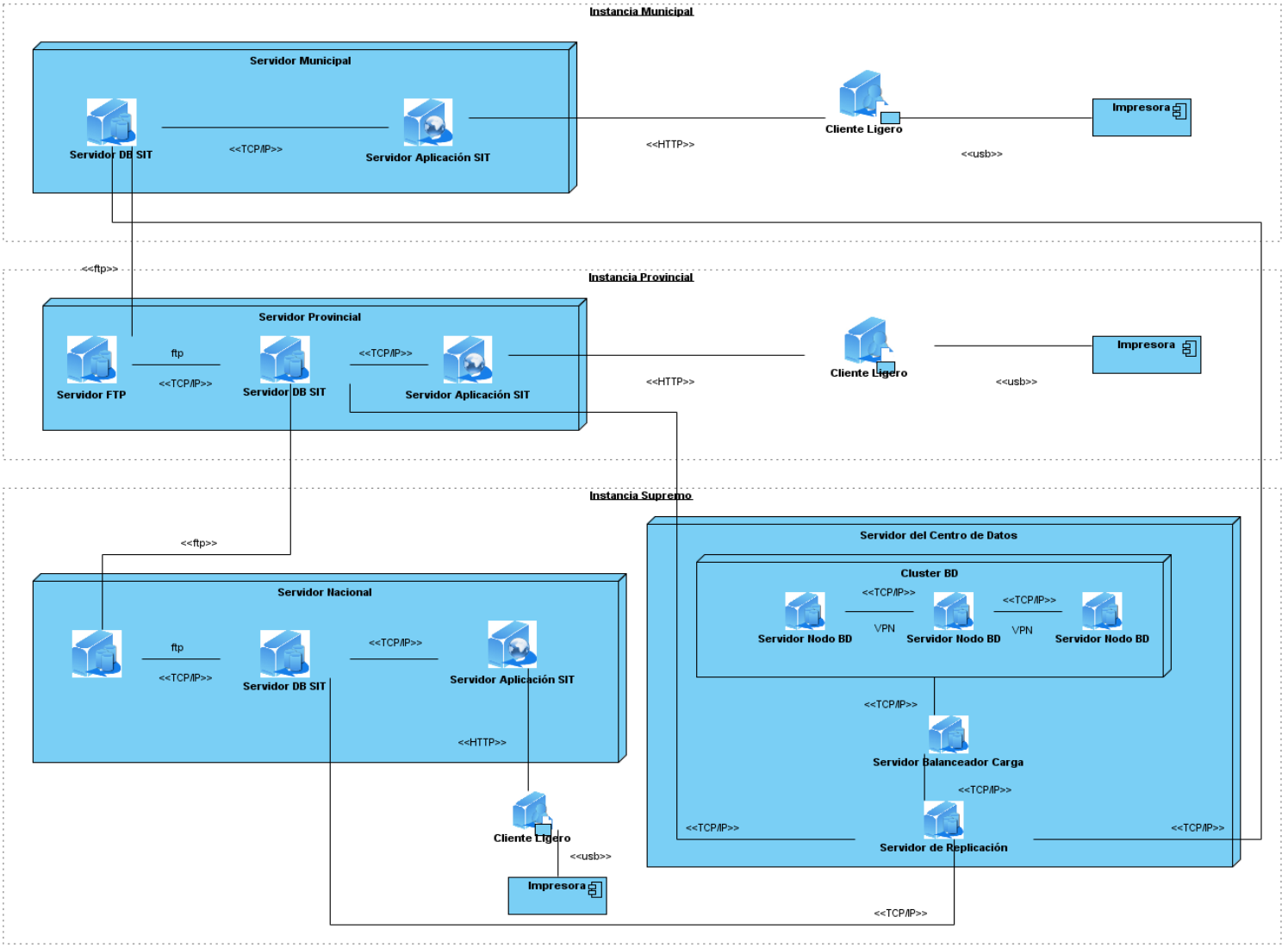

Figura 13: Despliegue del SIT a nivel nacional

2.5 Conclusiones

Los artefactos generados permitieron una mejor estructuración de la solución propuesta y el establecimiento de una guía para la implementación, lo que facilitó este proceso con la organización de las formas de relaciones entre clases y las funcionalidades a utilizar. Estos artefactos se conformaron a partir de los diagramas que propone RUP y que integran los modelos de diseño, implementación y despliegue mediante el uso de patrones. Estos patrones permitieron darle solución a los problemas surgidos durante el desarrollo, delimitar las funcionalidades a cumplir por cada clase, evitar la sobrecarga según las tareas que se le asignan y definir las responsabilidades correspondientes a los requisitos en cada caso de uso.

CAPÍTULO 3: ANÁLISIS DE LOS RESULTADOS

En el presente capítulo se evalúa el grado de calidad de la solución propuesta, a partir de la utilización de las métricas y/o mecanismos definidos previamente para la validación de los artefactos del diseño e implementación. Se realiza un análisis de los resultados obtenidos como parte de las actividades que integran el proceso de medición, tomando los datos arrojados por las métricas utilizadas.

3.1 Evaluación del modelo de diseño

Para la validación de los diagramas que integran el modelo de diseño se utilizarán las métricas APH y NDD correspondientes a la serie CK propuestas por Chidamber y Kemerer y la métrica TC perteneciente a las propuestas por Lorenz y Kidd, para determinar el grado de calidad y fiabilidad del diseño correspondiente a la solución presentada. Estas métricas se aplicarán a los diagramas de clase del diseño que se encuentran en el documento del modelo de diseño del subsistema Laboral del proyecto Tribunales Populares Cubanos y que constituye el Anexo 1 del presente trabajo.

3.1.1 Resultados del instrumento de evaluación de la métrica Árbol de Profundidad de Herencia (APH)

Para la medición del diseño con la métrica APH, se establece el siguiente umbral de medición:

Nivel de jerarquía de clases	Valor
Bajo	<=1 y <=3
Medio	>3 y <=5
Alto	>5 y <=10

Tabla 3: Valores del nivel jerárquico de clases

.

En la Tabla 4 se toman como muestra los casos de uso de importancia crítica para la materia, debido a que por su complejidad representan el comportamiento general que tendrá la métrica al aplicarse al resto de los diseños, tomándose por cada uno la cantidad de clases que poseen un nivel de jerarquía común.

Caso de uso	Cantidad de clases	Valor de jerarquía de clases
Registrar Escrito Inicial	4	1
	5	2
Registrar Prueba	9	1
	11	2
Registrar Tercer Afectado	5	1
	9	2
Registrar Representación	5	1
	6	2
Crear Acta Comparecencia	3	1
	7	2
Decidir Escrito de Demanda	4	1
	5	2
Crear Diligencia de Citación	2	1
	3	2
Buscar Expediente	6	2
Crear Sentencia	2	1
	8	2

Tabla 4: Jerarquía de clases por casos de uso

Para obtener un resultado tangible del comportamiento del Árbol de Profundidad de Herencia, se procederá a la realización del cálculo de la media de la siguiente forma:

$$\sum_{1}^{N}(CC * VJC) / \sum_{1}^{N} CC$$

Teniendo que:

N: cantidad de casos de uso

CC: valor de la cantidad de clases por caso de uso con determinado VJC

VJC: valor de jerarquía de clases

Sustituyendo los valores correspondientes: Media=154/94=1.638

Resultado: A partir de los datos obtenidos después de aplicar la métrica APH se concluyó que el nivel más alto de herencia tiene valor dos, tomando como referencia los valores del umbral establecido en la Tabla 3 para la media calculada de 1.638. El nivel jerárquico de las clases es bajo, por lo que no fue necesario hacer mucho uso de la herencia (representado en la Figura 14), con lo que se determinó que el diseño no es complejo y existe un bajo acoplamiento.

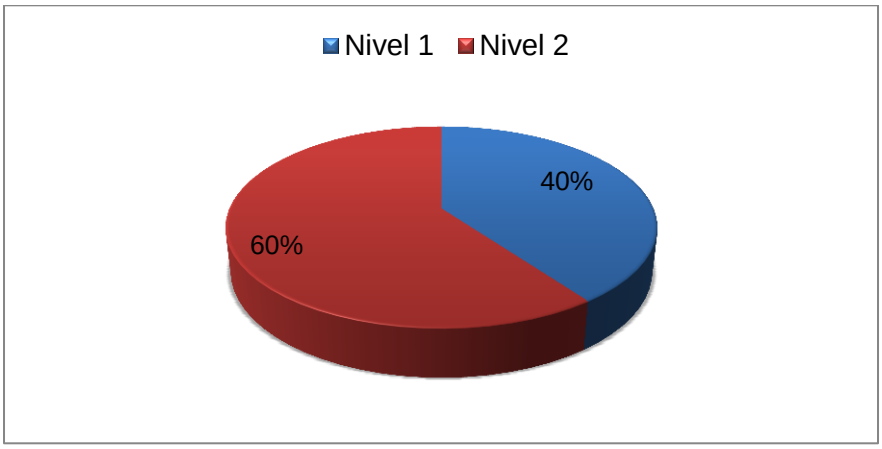

Figura 14: Gráfico: niveles de herencia en el diseño

3.1.2 Resultados del instrumento de evaluación de la métrica Número de Descendiente (NDD)

La métrica Número de Descendiente brinda resultados numéricos para determinar la complejidad de prueba del diseño, según los valores establecidos para medir los niveles de dependencia entre clases, se medirá el NDD para los principales casos de uso de mayor prioridad para la materia. Con este fin se establece el siguiente umbral de medición:

Nivel de descendencia de clases	Valor
Bajo	<=1 y <=3
Medio	>3 y <=5
Alto	>5 y <=10

Tabla 5: Valores de nivel de descendencia de clases

En la Tabla 6 se toman como muestra los casos de uso de importancia crítica para la materia, ya que por su complejidad representan el comportamiento general que tendrá la métrica al aplicarse al resto de los diseños, tomándose por cada uno la cantidad de clases que poseen una cantidad común de descendientes.

Caso de uso	Cantidad de clases	Valor del nivel de descendencia
Registrar Escrito Inicial	2	4
	6	1
Registrar Prueba	1	10
	1	9
	12	1
Registrar Tercer Afectado	1	8
	1	5
	10	1
Registrar Representación	2	5
	7	1
Crear Acta Comparecencia	6	1
	1	8
	3	1
Decidir escrito de demanda	2	4
	6	1
Crear Diligencia de Citación	2	2
	4	1
Buscar Expediente	1	5
	2	1
Crear Sentencia	7	1
	2	1
	1	9

Tabla 6: Descendencia de clases por casos de uso

Se procede al cálculo de la media para obtener el comportamiento del Número de Descendencia de los diseños utilizados, de la siguiente forma:

$$\sum_{1}^{N}(CC * VND) / \sum_{1}^{N} CC$$

Teniendo que:

N: cantidad de casos de uso

CC: valor de la cantidad de clases por caso de uso con determinado VND

VND: valor del nivel de descendencia.

Sustituyendo los valores correspondientes: Media=149/80=1.862

Resultado: A partir de los datos obtenidos después de aplicar la métrica NDD se concluyó que el nivel más alto de descendencia toma valor diez, tomando como referencia los valores del umbral establecido en la Tabla 5 para la media calculada de 1.862. El nivel de descendencia de las clases es Bajo, por lo que se reduce la reutilización de las mismas, obteniéndose niveles medios. El número de pruebas requeridas a realizar a cada descendiente en la mayoría de los casos se obtuvieron niveles bajo y medios, resultado que se representa en la Figura 15.

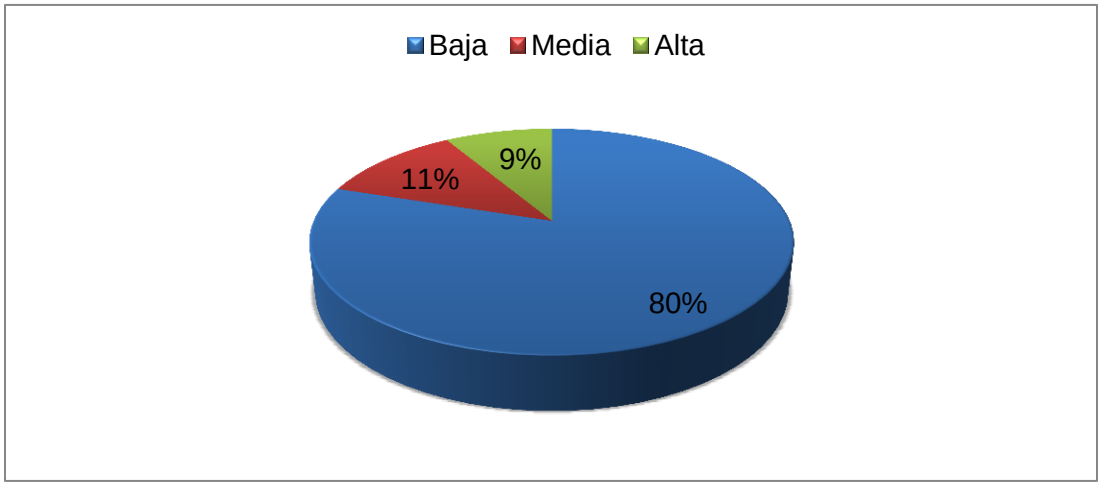

Figura 15: Gráfico: porcentaje de la cantidad de pruebas en el diseño

3.1.3 Resultados del instrumento de evaluación de la métrica Tamaño de Clase (TC)

La métrica para la evaluación del diseño a partir del Tamaño de Clase consiste como se planteó anteriormente en medir el tamaño de una clase a partir de las siguientes medidas:

- Total de operaciones (operaciones tanto heredadas como privadas de la instancia), que se encapsulan dentro de la clase.
- Número de atributos (atributos tanto heredados como privados de la instancia), encapsulados por la clase, en el caso que posean.

Esta métrica plantea que un alto valor de TC influye de manera directa en ciertos indicadores de calidad del software, descritos en la Tabla 7:

Parámetros de calidad	Altos valores de TC
Reutilización	Reduce la posibilidad de reutilización
Implementación	Complica la Implementación del sistema
Prueba	Hace compleja las pruebas
Responsabilidad	Aumenta la responsabilidad de la clase

Tabla 7: Parámetros de calidad para altos valores de TC

Generalmente el establecimiento de umbrales para la medición de estos parámetros ha constituido una polémica, por lo que algunos especialistas del tema, entre los que se encuentra Pressman como uno de los más referenciados, plantean el siguiente rango, que será el adoptado para la medición del diseño de este sistema, con el que se obtuvieron los resultados de la cantidad de clases por tamaño de clase:

Umbral de medición	Tamaño de clase	Cantidad de clases
>=0 y <=20	Pequeño	12
>20 y <=30	Medio	2
>30	Grande	0

Tabla 8: Cantidad de clases por tamaño

La métrica TC fue aplicada a la capa controladora porque es la más compleja, y maneja toda la lógica del negocio pues de una forma u otra utiliza todas las clases de la vista y del

modelo. Para las principales clases definidas en el diseño realizado se obtuvieron los valores del total de operaciones. A continuación se resume en la Tabla 9 todas las clases controladoras que integran el diseño, las operaciones o funcionalidades que posee cada una y el tamaño que tendrán según el umbral planteado anteriormente:

No.	Clase controladora	Cantidad de operaciones	Tamaño
1	RegistrarEscritoInicialController.php	22	Medio
2	RegistrarPruebaController.php	16	Pequeño
3	RegistrarEscritoController.php	9	Pequeño
4	RegistrarRepresentaciónProcesalController.php	11	Pequeño
5	RegistrarDemandaAbogadoController.php	3	Pequeño
6	DecidirEscritoDemandaController.php	13	Pequeño
7	DecidirPruebaController.php	6	Pequeño
8	DecidirEscritoRepresentacionProcesalController.php	6	Pequeño
9	CrearDiligenciaCitacionController.php	6	Pequeño
10	CrearActaCPController.php	21	Medio
11	CrearADVSController.php	2	Pequeño
12	BuscarEscritoController.php	4	Pequeño
13	BuscarExpedienteController.php	3	Pequeño
14	CrearSentenciaController.php	2	Pequeño

Tabla 9: Representación del tamaño de clases de la materia Laboral

Se trabajó con un total de 14 clases controladoras que engloban las funcionalidades de los 33 casos de usos con los que cuenta la materia Laboral, para un promedio de 8.8 operaciones por clases. Los resultados de la cantidad de clases se muestran a continuación en la Figura 16, en las que se muestran las representaciones de los porcentajes obtenidos.

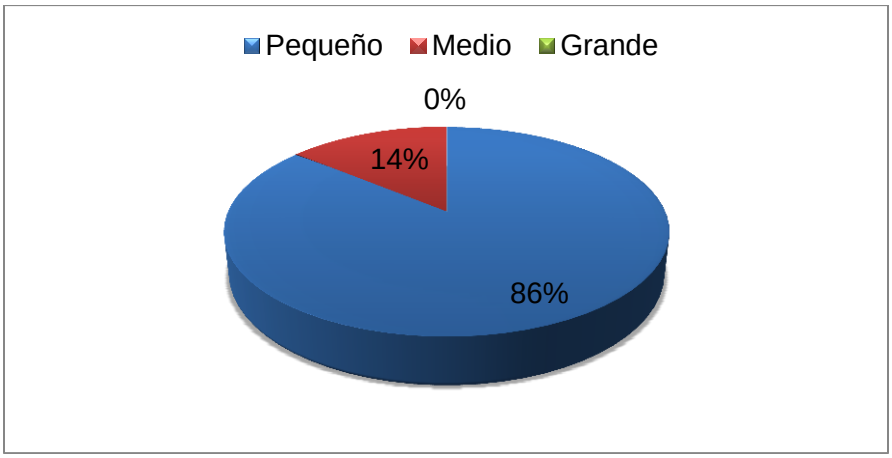

Figura 16: Gráfico: porcentaje del total de clases por tamaño

Resultados: La aplicación de la métrica TC, concluyó que la mayoría de las clases quedaron en la clasificación de pequeño y algunas en medio, por lo que este indicador demuestra que los valores de TC no son altos. La mayoría de las clases son reutilizables Figura 17, por lo que el sistema no tendrá una compleja implementación y la realización de las pruebas se podrá realizar de manera factible.

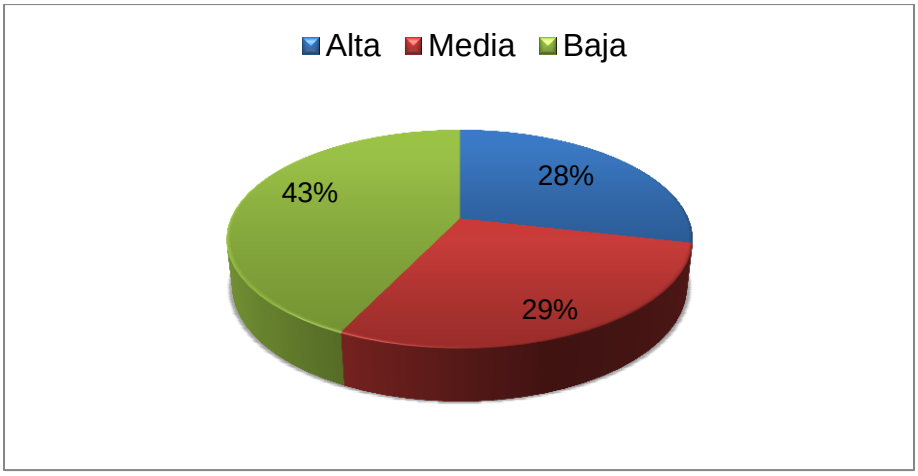

Figura 17: Gráfico: representación de los niveles de reutilización en el diseño

3.2 Evaluación de la calidad del producto: pruebas de Caja Blanca

Las pruebas de Caja Blanca deben garantizar que se ejerciten por lo menos una vez todos los caminos independientes para cada módulo, se realicen todas las decisiones lógicas en sus vertientes verdaderas y falsas, se ejecuten todos los bucles con sus límites operacionales y que se efectúen las estructuras internas de datos para asegurar su validez. La técnica más utilizada es la del Camino Básico debido a que con el cálculo de una medida de la complejidad ciclomática, posibilita obtener casos de pruebas factibles que tengan una alta probabilidad de mostrar un error no detectado hasta el momento.

Esta técnica es aplicada a las funcionalidades que realizan los casos de usos de manera independiente, los que integran el Anexo 4 Pruebas de Caja Blanca. De esta manera el procedimiento será aplicado a la clase controladora RegistrarEscritoInicialController.php, en la funcionalidad eliminarRowdemandanteAction(), en la que se elimina de la sesión que carga en un listado los demandantes que se han insertado en el sistema, los datos que se encuentren en el índice seleccionado. Con este objetivo primeramente se ejecutan las precondiciones necesarias, luego se sustituyen los valores de la posición que se desea eliminar por los de la inmediata superior, los cuales serán eliminados de la sesión, para finalmente hacer un llamado a la funcionalidad local cargarsesiondemandanteAction(), que cargará la sesión nuevamente.

Se presenta un ejemplo de este procedimiento, en la Figura 18 donde se muestra el código que ejecuta dicho procedimiento, con el orden enumerado de las instrucciones que realiza:

```
function eliminarRowdemandanteAction() {
    $indice = $this->_request->getPost('indice');  //1
    $persona = $this->_request->getPost('persona');   //1
    if ($persona == 'demandante') {   //2
        $cantidad = count($_SESSION['Escritoinicial']['Demandante']);      //3
        for ($i = $indice; $i <= $cantidad - 1; $i++) {   //4
            $_SESSION['Escritoinicial']['Demandante'][$i] = $_SESSION['Escritoinicial']['Demandante'][$i + 1];  //5
        }
        unset($_SESSION['Escritoinicial']['Demandante'][$cantidad - 1]);    //6
    }
    return $this->cargarsesiondemandanteAction();   //7
}
```

Figura 18: Método eliminarRowdemandanteAction

Para la enumeración de las instrucciones, es necesario tener en cuenta, las condicionales, los ciclos y las instrucciones lineales. En este caso se le da el valor uno a las instrucciones lineales en las que se guardan en las variables $indice y $persona los parámetros que son enviados por la petición Ajax, que se realiza al controlador, que indican la posición en la que se desea eliminar y el tipo de persona, que deberá ser un demandante. El valor dos lo tiene la condicional, que en caso de cumplirse se ejecutaría la instrucción que tiene valor tres en la que se obtiene la cantidad de personas que están guardados en la sesión del demandante. El valor cuatro es para el ciclo "for", que se ejecuta al igual que la instrucción seis, en esta última se procede a vaciar la sesión en la posición indicada. Mientras que el valor cinco lo toma el caso en que se cumplen las condiciones para que se ejecute el ciclo "for". Finalmente se elimina de un componente gridpanel[4], que es utilizado generalmente cuando se desea obtener una lista de valores, los datos del demandante seleccionado.

Los pasos para la elaboración de la técnica de Caja Blanca Camino Básico son:

- Paso 1: Dibujar el grafo de flujo asociado.

Para la confección del grafo de flujo asociado al método de la Figura 18 es necesario primeramente conocer los conceptos de los elementos que lo integran.

Grafo: es un par (V, A) donde V es un conjunto finito de elementos que se denominan vértices y A es un conjunto de pares no ordenados <x,y>, donde $x \in V$, $y \in V$, denominados aristas o arcos.

Nodo: es una o más secuencias de procesos, que pueden ser instrucciones o sentencias de decisión, pueden haber nodos sin asociar que sólo se utilizan al principio o al final del código.

Arista: representan el flujo de información.

Región: Área limitada entre nodos y aristas.

En la Figura 19 se presenta el grafo correspondiente al código de la funcionalidad planteada.

[4] Componente que provee Extjs para listar datos.

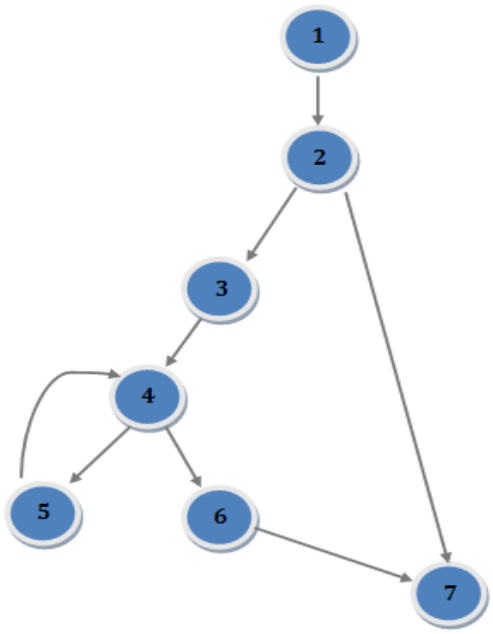

Figura 19: Grafo de flujo del método eliminarRowdemandanteAction

- Paso 2: Calcular la complejidad ciclomática del grafo.

La complejidad ciclomática define el número de caminos independientes, que a su vez proporciona el mínimo número de pruebas que se debe realizar, permitiendo que una secuencia se ejecute al menos una vez. Para esto se plantean las siguientes fórmulas:

V (G) = (A - N) + 2, teniendo que A es el número de aristas y N es el número de nodos del grafo de flujo asociado. Sustituyendo *V (G) = (8 - 7) + 2, V (G) = 3*

V (G) = P + 1, teniendo que P es el número de nodos predicados, es decir que de ellos parten al menos 2 aristas. Sustituyendo *V (G) = 2 + 1*, *V (G) = 3*

V (G) = R, teniendo que R es el número total de regiones del grafo de flujo asociado, incluyendo la región externa. Sustituyendo *V (G) = 3*

Por lo que se concluye que existen 3 caminos independientes por los que el flujo del grafo puede circular y el mínimo número de casos de pruebas que deberán ejecutarse.

- Paso 3: Determinar el conjunto básico de caminos independientes.

Un camino independiente constituye cualquier camino o vía de ejecución del código o programa que introduce un nuevo conjunto de sentencias de procesos o una nueva condición, que deberá abarcar al menos una arista que aún no haya sido recorrida.

Camino básico #1: 1-2-7
Camino básico #2: 1-2-3-4-6-7
Camino básico #3: 1-2-3-4-5-4-6-7

- Paso 4: Preparar los casos de prueba que sigan la ejecución de cada camino.

Un caso de prueba es un conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para cumplir un objetivo o una funcionalidad, (siempre se ejecutan como una unidad desde el comienzo hasta el final). Para el diseño de los casos de prueba a partir de los caminos básicos o independientes obtenidos se planean:

- La descripción, que puntualiza el caso de prueba y sus datos de entrada.
- La condición de ejecución, que especifica las condiciones de la que depende la ejecución.
- La entrada, que muestra los parámetros que serán los valores que está recibiendo el procedimiento.
- Los resultados esperados, que expone la salida que debe devolver el procedimiento después de efectuado el caso de prueba.

El caso de prueba para el camino básico # 1 como parte de la primera iteración realizada es el siguiente:

Caso de prueba # 1

Camino: 1-2-7

Descripción: Se desea eliminar de una sesión que es cargada por el componente gridpanel los datos del demandante seleccionados por el usuario en el caso de uso Registrar Escrito Inicial.

Condición de ejecución: Para la ejecución del algoritmo es necesario que la variable $persona no tenga valor 'demandante'.

Entrada: La variable $persona no tiene valor 'demandante', tiene valor 'demandado'.

Resultados esperados: Teniendo en cuenta los datos pasados por parámetro se espera que en el gridpanel no se eliminen los datos del demandante. El resultado obtenido fue correcto.

Resultados: La aplicación de las pruebas de caja blanca, arrojaron como resultados que luego de realizar la primera iteración se obtuvo que el número de casos de pruebas con

resultados correctos es superior al número de casos de pruebas con resultados incorrectos (representados en el gráfico de la Figura 20). Estos últimos fueron aplicados nuevamente, luego de rectificar los errores, en la segunda iteración, obteniéndose resultados positivos.

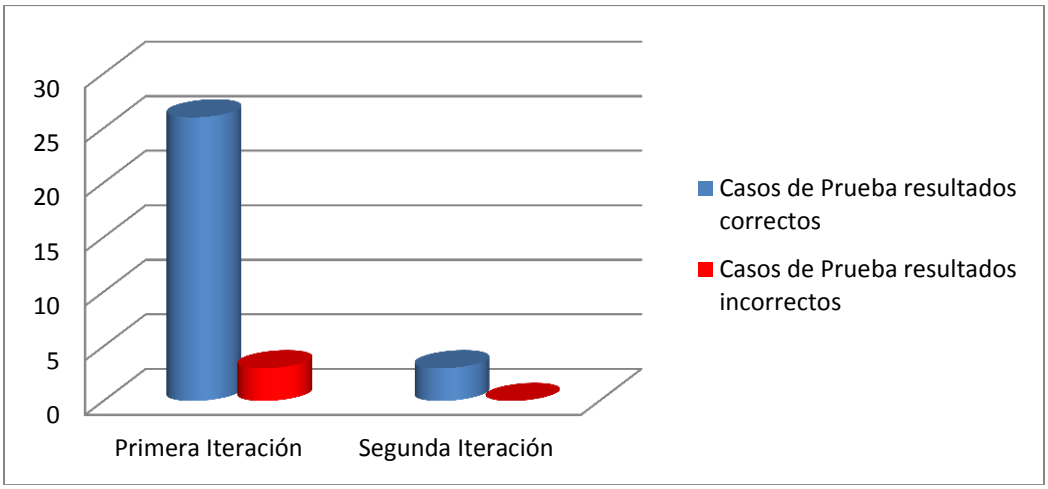

Figura 20: Gráfico: resultados por iteraciones de los casos de prueba de caja blanca

3.3 Evaluación de la calidad del producto: pruebas de Caja Negra

Las pruebas de caja negra son utilizadas para encontrar los siguientes errores:

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

Existen varias técnicas para su implementación, la utilizada es la de partición equivalente que plantea dividir el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de prueba, esforzándose por definir un caso que descubra ciertas clases de errores, reduciendo así el número total a desarrollarse.

Los diseños de casos de prueba elaborados a partir de la aplicación de esta técnica se encuentran en el Anexo 4 Pruebas de Caja Negra que conforman la solución del presente trabajo. Los resultados obtenidos se resumen en el gráfico de la Figura 21.

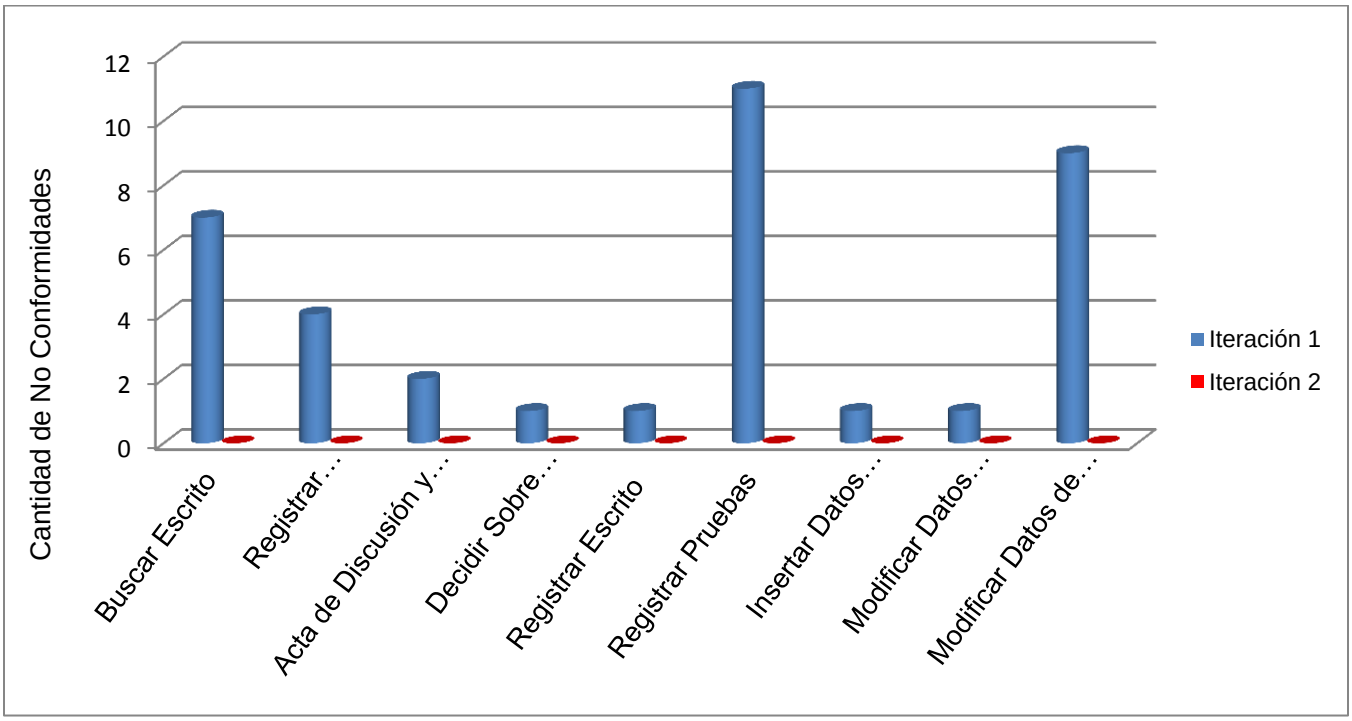

Figura 21: Gráfico: resultados por iteraciones de los casos de prueba de caja negra

Resultados: La aplicación de las pruebas funcionales de caja negra arrojó como resultados, que fue necesario realizar dos iteraciones, en la primera se detectaron no conformidades de funcionalidad, de correspondencia con la documentación y de validación, que posteriormente fueron corregidas satisfactoriamente en la segunda iteración, en la que no se encontró ninguna no conformidad.

3.4 Conclusiones

En este capítulo se determinó que el diseño propuesto no presenta una alta complejidad estructural, como resultado de la aplicación de las métricas APH, TC y NDD, que evidenciaron que la mayoría de las clases son reutilizables y tienen bajo acoplamiento. Se aplicaron las pruebas de caja blanca o estructural, con lo que se determinaron los errores

internos, y la pruebas de caja negra a la interfaz del sistema cuyos resultados permitieron obtener una aplicación a nivel de interfaz lo más confiable posible, como consecuencia de las iteraciones realizadas.

CONCLUSIONES GENERALES

Se realizó el estudio del estado del arte de algunos sistemas de gestión procesal, llegando a la conclusión que no satisfacen los requerimientos que demanda la informatización de los procesos Disciplina y Derecho Laboral, ya que los nacionales no abarcan estos procesos y los internacionales no se ajustan al sistema jurídico cubano.

- Las herramientas definidas por la arquitectura aportaron al desarrollo del sistema mayor rapidez en las respuestas a las solicitudes del usuario, integridad de los datos e independencia tecnológica.
- A partir de los diagramas de clase del diseño se implementaron los procesos del flujo principal de la materia Laboral para el sistema de tribunales en Cuba y con la elaboración del diagrama de despliegue se establecieron los entornos para el despliegue del sistema y su integración a nivel nacional.
- A partir del modelo de implementación se pudo obtener un producto funcional que cumple con las restricciones planteadas por el equipo de arquitectura.
- La aplicación de métricas para evaluar tanto el diseño como la implementación de la solución propuesta, concluyeron que presentan el nivel de calidad requerida.

RECOMENDACIONES

Realizar el diseño e implementación de los procesos de la materia Laboral de los Tribunales Populares Cubanos, Recurso de Apelación y Demanda por Seguridad para la instancia provincial, y en la instancia suprema de los procesos Revisión de Sentencia y Apelación de Seguridad Social.

BIBLIOGRAFÍA

1. **Andux, Yadira Piñera. Junio 2010.** *Formalización y estandarización de la documentación técnica de la arquitectura tecnológica del Marco de Trabajo Sauxe versión 2.0.* Universidad de Ciencias Informática, UCI. La Habana, Cuba : s.n., Junio 2010. Tesis de Diploma.
2. **Bakken, Stig Sæther y Aulbach, Alexander . Abril 2001.** *Manual de PHP.* [ed.] Rafael Martínez. Abril 2001. pág. 500.
3. **Blanco, Ignacio Carlos. 2001.** *Tesis de Ingeniería en Informática, Plataformas de desarrollo de aplicaciones Web orientadas a componentes reutilizables.* 2001. Tesis de Diploma.
4. **Booch, Grady. 1985.** *Análisis y Diseño Orientado a Objetos con Aplicaciones.* 1985. 311-73-0161-3.
5. **Booch, Grady, Jacobson, Ivar y Rumbaugh, James. 2000.** *El Proceso Unificado de Desarrollo de Software.* [ed.] Andrés Otero. [trad.] Salvador Sánchez y Miguel Ángel Silicia. Primera. Madrid : Fareso, S.A, 2000. pág. 469. 84-7829-036-2.
6. **Camacho, Erika, Cardeso, Fabio y Nuñez, Gabriel . Abril 2004.** *Arquitecturas de software, guía de estudio.* Abril 2004.
7. **Corporation, Oracle. 2012.** Manual de Netbeans. [En línea] 2012. http://netbeans.org/index_es.html.
8. **Delgado García, Ana María y Oliver Cuello, Rafael. 2007.** Iniciativas recientes de las e-justicia en España. [En línea] 4, Febrero de 2007. http://idp.uoc.edu. 1699-8154.
9. **Doctrine. 2009.** *Manual de Doctrine para PHP.* [ed.] Juan Ignacio Luca de T. 1.2. s.l. : Premier Press, Inc, 2009. pág. 200. Vol. 2. 84-4 15-1420-8.
10. **Dotras, Dídac. 2010.** 63, Cataluña : s.n., mayo de 2010, Revista Socinfo, pág. 1.
11. **ElDia.es. 2010.** *Eldia.es.* [En línea] 19 de 10 de 2010. http://www.eldia.es/2010-10-19/canarias/39-PSC-censura-Atlante-II-lleve-dos-anos-funcionar-juzgados-Canarias.htm..
12. **Frederick, Shea, Ramsay, Collin y Blades, Steve. 2008.** *Learning ExtJS.* Birmingham : Packt Publishing, 2008. 978-1-847195-14-2.

13. **García, Ana María Delgado. 2007.** E-Justicia en España. *Iniciativas recientes de la e-justicia en España.* [En línea] Febrero de 2007.
14. **Giraldo, Luis y Zapata, Juliana. 2005.** Herramientas de desarrollo de ingeniería de Software para Linux. [En línea] Septiembre de 2005. http://hugolopez.phi.com.co/docs/download/file=Giraldo-Zapata-Herramientas%20de%20ISW.pdf,_id=17.
15. **Gómez Gutiérrez , Alex . 2011.** *Diseño e Implementación de una solución informática para el proceso Administrativo en los Tribunales Provinciales Cubano.* Universidad de Ciencias Informáticas. La Habana, 2011. Tesis de Pregrado.
16. **González Pérez, Reisel . Febrero 2008.** *Introducción al Sistema de Gestión de Base de Datos PostgreSQL.* Universidad de Ciencias Informáticas, UCI. Ciudad de la Habana : s.n., Febrero 2008.
17. **Ibarra, Germán Castro. Agosto 2001.** Historia de la Informática. [En línea] Agosto 2001. http://eva.uci.cu/file.php/103/Bibliografia_Basica/Materiales_basicos/1_Historia_de_la_informatica.pdf.
18. **Laman, Craig. 1999.** *UML y Patrones, Introducción al análisis y diseño orientado a objetos.* [ed.] Pablo Eduardo Roig Vázquez. Primera. s.l. : Prentice Hall, 1999. pág. 112. 970-17-0261-1.
19. **Morell, M. D. E. C. and M. J. R. G. Guadarramas. 2008.** Sistema para la Tramitación de Procesos Penales. Facultad Matemática-Física-Computación. Universidad Central "Marta Abreu" de Las Villas.Villa Clara,Cuba, : s.n., 2008.
20. **Morín, Arley Triana. 2009.** DesarrolladorSenior. [En línea] 2009. http://desarrolladorsenior.blogspot.com/2009/09/el-patron-de-diseno-singleton-esta.html..
21. **Pérez, Javier Eguíluz. 2008.** Introducción a Ajax. [En línea] Febrero de 2008. www.librosweb.es.
22. **—. 2008.** Introducción a JavaScript. [En línea] Febrero de 2008. http://www.librosweb.es/javascript.
23. **Pressman, Roger. 2005.** *Ingeniería de Software. Un enfoque práctico.* España : Mc Graw Hil, 2005.
24. **—. 2002.** *Métricas del Producto para el Software.* 2002. Vol. Capítulo 15.

25. **—. 2002.** *Técnicas de Prueba del software.* s.l. : McGraw-Hill Companies , 2002. Vols. Capítulo 14, Parte 2. 8448132149.
26. **Prince, Rushton . 2005.** *Implementing RUP in 10 steps.* 2005.
27. **Productiva, Dirección de Calidad de la Infraestructura. Octubre 2010.** *Arquitectura de Software del Sistema de Informatización de los Tribunales Populares Cubanos.* Universidad de Ciencias Informáticas, UCI. La Habana, Cuba : s.n., Octubre 2010.
28. **—. Marzo 2011.** *Especificación de requisitos de software del Subsistema Laboral de los Tribunales Populares Cubanos.* Universidad de Ciencias Informáticas. La Habana, Cuba : s.n., Marzo 2011.
29. **—. Junio 2010.** *Plan de Desarrollo de Software de los Tribunales Populares Cubanos para el Sistema de Informatización de Tribunales versión 0.0.3.* Universidad de Ciencias Informáticas, UCI. La Habana., Cuba : s.n., Junio 2010.
30. **SIGETCU, Infraestructura Productiva. Abril 2010.** *Modelo de Negocio con BPM del Subsistema Laboral de los Tribunales Populares Cubanos.* Universidad de Ciencias Informáticas, UCI. La Habana, Cuba : s.n., Abril 2010.
31. **Solano Vera, H. 2005.** *Interfaz para monitoreo de redes de comunicación mediante una aplicación web.* Departamento de Ingeniería en Sistemas Computacionales, Universidad de las Américas Puebla. México : s.n., 2005. Tesis de Licenciatura.
32. **Tenrero Cabrera, Marianela y Morejón Borbon, Yoandry. 2011.** *Manual de instalación de las herramientas para Sauxe 1.5 para Windows y Linux.* Departamento de Tecnología del Centro de Soluciones de Gestión de Entidades (CESGE);Centro de Desarrollo y Asimilación de las Tecnologías de la Unidad de Compatibilización, Integración y Desarrollo (UCID), Universidad de las Ciencias Informáticas. 2011. pág. 50.

Glosario

OJLB: Órgano de Justicia Laboral de Base.

TMP: Tribunal Municipal Popular.

TPP: Tribunal Provincial Popular.

TSP: Tribunal Supremo Popular.

CP: Comparecencia Pública que se realiza luego de haberse admitido la demanda, donde la secretaria procede a citar a las partes, mediante una DDC.

DDC: Diligencia dando cuenta.

Negocio: Contexto en el que se desarrollará la solución de software, esencia de los procesos de la organización.

Proceso: Conjunto de tareas lógicamente relacionadas que existen para conseguir un resultado bien definido dentro de un negocio; por lo tanto, toman una entrada y le agregan valor para producir una salida. Los procesos tienen entonces clientes que pueden ser internos o externos, los cuales reciben la salida, lo que puede ser un producto físico o un servicio. Estos establecen las condiciones de satisfacción o declaran que el producto o servicio es aceptable o no.

Diligencia: Citación o notificación elaborada por la Secretaria del proceso, puede ser una citación a una de las partes implicadas o una notificación de la sentencia dictada.