Universidad de las Ciencias Informáticas

FACULTAD 3

Título: Desarrollo del módulo Administración y Gobierno para el Proyecto de Informatización para la Gestión de los Tribunales Populares Cubanos.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Ricardo Alberto Alejandre Nuevo

Tutor(es): Ing. Darián González Ochoa

Co-tutor: Ing. Arianna Leyva Campos

La Habana Curso 2012-2013

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Ricardo Alberto Alejandre Nuevo

_____
Firma del Autor

Ing. Darián González Ochoa

_____
Firma del Tutor

Ing. Arianna Leyva Campos

_____
Firma del Co-tutor

DATOS DE CONTACTO

Ing. Darián González Ochoa: (dochoa@uci.cu) Graduado de la Universidad de las Ciencias Informáticas en el año 2009, Diplomado en Docencia e Innovación Universitaria y Diplomado en Gobierno Electrónico. Profesor Instructor del Centro de Gobierno Electrónico, Jefe del Departamento de Gestión Gubernamental.

Ing. Arianna Leyva Campos: (alcampos@uci.cu) Graduada de Ingeniera en Ciencias Informáticas en el año 2010 en la Universidad de las Ciencias Informáticas (UCI). Profesor interno. Ha desempeñado los siguientes roles: analista, programadora, planificadora y jefa de subsistema del Proyecto de Informatización de los Tribunales Populares Cubanos. Ha cursado todos los cursos obligatorios del diplomado de Docencia e Investigación Universitaria y de Gobierno Electrónico.

AGRADECIMIENTOS

Antes que todo el agradecimiento a mis padres por el apoyo que me han brindado toda mi vida; a mi madre por traerme al mundo y por aguantar mis malacrianzas muchas veces; a mi padre por ser ejemplo en mi vida y por presentarme este mundo que es la Informática; a mis abuelos por complacerme en todo lo que podían; a mi abuela por ser mano dura en mi vida y darme gran parte de la educación que hoy tengo, ya hoy tienes un nieto ingeniero; a mi hermano por obligarme a ser ejemplo y por ser mi amigo además de mi hermano; a mis tías y tíos, mis primos y toda mi familia que por una razón u otra no pudieron estar aquí hoy; a Mario por guiarme en el mundo de la Informática; a los amigos del pre que me han apoyado durante la carrera, estando o no en la UCI; a la gente de mi grupo 3501, sin dudas el mejor grupo de la UCI, a la gente del edificio 9 por hacer que estos 5 años parecieran 2; a la familia del 9208, Josué, Luis, Miguel Ángel, El Bati, Yosviel, Raidel, Remberto, Carlito, Mark, Randy y Wilian; a los profesores que me han impartido clases durante estos 5 años, en especial a la profe Dariela, Mónica y Dina; a todos mis compañeros del proyecto por la ayuda que me han dado este año; a mis tutores, Darian y Arianna por todo el apoyo que me han dado con la tesis y con el proyecto. En fin a todas aquellas personas que me han apoyado y creído en mí, durante toda mi vida, de corazón gracias.

DEDICATORIA

A mis padres y a mi abuela por todo el apoyo, cariño y comprensión sin ellos este logro no hubiera sido posible.

RESUMEN

Los Tribunales Populares Cubanos (TPC) precisan de un sistema informatizado que, desde un enfoque técnico interdisciplinario, contribuyan a hacer eficiente, efectiva y eficaz su gestión, salvando el atraso que se observa al comparar el estado actual de su funcionamiento en este campo, con los logros obtenidos en el primer mundo. Por esta razón, el Tribunal Supremo Popular de Cuba (TSPC) inició un proyecto de cooperación con la UCI el cual tiene como objetivo informatizar los procesos que se gestionan en los TPC. En el funcionamiento de los TPC actualmente todos los procesos se llevan a cabo manualmente y con gran dependencia del documento escrito en papel. Por otra parte el control de los vencimientos de término es una ocupación tediosa y requiere un alto grado de desgaste por la cantidad de procesos que se atienden.

Es por ello que el presente trabajo tiene como objetivo el diseño e implementación del módulo Administración y Gobierno perteneciente al Proyecto Sistema de Informatización para la Gestión de los Tribunales Populares Cubanos (SITPC), como solución a los problemas identificados en los Tribunales Populares Cubanos.

Para la realización del sistema se utilizó RUP[1] como metodología de desarrollo, UML como lenguaje de modelado, Visual Paradigm como herramienta CASE[2], PHP como lenguaje de programación, PostgreSQL como gestor de base de datos, Apache como servidor web y como marcos de trabajo Jquery, Symfony2.0 y Doctrine. Como resultado se obtuvo un módulo funcional que está en fase de pruebas de liberación en el centro CALISOFT en la UCI.

Palabras clave: *SITP, módulo Administración y Gobierno, SITPC, vencimiento de términos.*

[1] Rational Unified Process (Proceso unificado de Rational)

[2] Computer Aided Software Engineering (Ingeniería de software asistida por computadora)

ÍNDICE

DECLARACIÓN DE AUTORÍA II
DATOS DE CONTACTO III
AGRADECIMIENTOS 1
DEDICATORIA 2
RESUMEN 3
INTRODUCCIÓN 10
CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 14
Introducción 14
1.1. Definiciones relacionadas con el objeto de estudio y el campo de acción 14
1.2. Módulo Administración y Gobierno 14
1.3. Estado del arte de sistemas informáticos relacionados con el campo de acción 16
1.3.1. Sistemas estudiados en el mundo 16
1.3.2. Sistemas estudiados en Cuba 18
1.4. Fundamentación de las herramientas, metodología y tecnologías a utilizar 20
1.4.1. Metodología de Desarrollo de Software: RUP 20
1.4.2. Proceso Unificado de Rational 20
1.4.3. Herramienta de Ingeniería de Software Asistido por Ordenadores (CASE) para el modelado 21
1.4.3.1. Visual Paradigm 8.0 21
1.4.4. Lenguaje de programación para el desarrollo Web: PHP 5.3.3 22
1.4.5. Sistema gestor de base de datos: PostgreSQL 9.1 22
1.4.6. Marcos de trabajo(frameworks) 23
1.4.6.1. Symfony 2.1 23
1.4.6.2. Bootstrap 2.0 24
1.4.6.3. jQuery 24
1.4.6.4. Doctrine 26
1.4.7. Entorno de Desarrollo Integrado (IDE): NetBeans 7.2 26
1.4.8. Servidor web: Apache 2.2.22 26
1.4.9. Patrones de arquitectura 27
1.4.9.1. Modelo Vista Controlador (MVC) 27
1.4.9.2. Modelo Cliente-Servidor 27
1.4.9.3. Multicapas 28

Conclusiones parciales 28
CAPÍTULO 2: PROPUESTA DE SOLUCIÓN 30
Introducción 30
2.1 Arquitectura del Sistema 30
2.1.1 Vista 30
2.1.2 Controlador 31
2.1.3 Modelo 31
2.1.3.1 Capa de acceso a datos 31
2.1.3.2 Capa de abstracción de datos 33
2.2 Modelo de datos 34
2.3 Diseño e implementación 35
2.3.1 Modelo de diseño 35
2.3.1.1 Diagrama de clases del diseño 36
2.3.1.2 Diagrama de interacción 37
2.3.1.2.1 Diagrama de secuencia 38
2.3.1.3 Diagrama de despliegue 38
2.3.1.3.1 Descripción de los nodos físicos 39
2.4 Estándares de codificación 40
2.4.1 Identación 40
2.4.2 Cabecera del archivo 40
2.4.3 Comentarios en las funciones 40
2.4.4 Ubicación de los archivos 41
2.4.5 Denominación de archivos 41
2.4.5.1 Las clases de gestión del negocio usarán el sufijo Gtr: 41
2.4.5.2 Los table model definidos para los grid usarán el sufijo TableModel: 42
2.4.5.3 Las entidades de presentación usarán el sufijo EP: 42
2.4.5.4 En las páginas, plantillas html.twig definidas para la vista el nombre del archivo debe seguir el estándar de la denominación de las clases. Ejemplo: 42
2.4.5.5 Para la denominación de los servicios a crear se tendrán en cuenta los siguientes elementos: 43
2.4.6 Clases 43
2.4.7 Estilo y reglas de escritura de código PHP 44
2.4.7.1 Nombre de las variables 44

2.4.7.2 Las definiciones de la función 44
2.4.7.3 Llamadas a funciones 45
2.4.7.4 Siempre incluir llaves 45
2.4.7.5 Colocar correctamente las llaves 45
2.4.7.6 Poner espacios entre signos 45
2.4.7.7 Precedencia de operadores 46
2.4.7.8 No utilizar variables sin inicializar 46
2.4.7.9 Instrucción “switch” 46
2.5 Patrones de diseño 47
2.5.1 Patrones GOF 47
2.5.1.1 Factory Method (Método de fabricación) 49
2.5.1.2 Decorator (Envoltorio) 50
2.5.2 Patrones GRASP 51
2.5.2.1 Creador 51
2.5.2.2 Experto 52
2.5.2.3 Bajo acoplamiento 53
2.5.2.4 Alta cohesión 53
2.5.2.5 Controlador 53
2.6 Inyección de dependencia 55
2.7 Conclusiones parciales 55
CAPÍTULO 3: VALIDACIÓN DE LOS RESULTADOS 56
Introducción 56
3.1 Métricas de software. 56
3.1.1 Métricas aplicadas al diseño del software. 57
3.1.1.1 Tamaño Operacional de Clase (TOC) 57
3.1.1.2 Relaciones entre Clases (RC) 57
3.1.1.3 Carencia de cohesión en los métodos (CCM) 57
3.1.1.4 Árbol de profundidad de herencia (APH) 58
3.1.2 Resultados de la aplicación de las métricas de diseño 58
3.1.2.1 Resultado de la aplicación de la métrica Tamaño Operacional de Clase (TOC) 58
3.1.2.2 Resultado de la aplicación de la métrica Relación entre Clases (RC) 59
3.1.2.3 Resultado de la aplicación de la métrica Carencia de Cohesión en los Métodos (CCM) 61

3.1.2.4 Resultado de la aplicación de la métrica Árbol de Profundidad de Herencia (APH) . 63
3.2 Pruebas 64
3.2.1 Pruebas Funcionales 65
3.2.1.1 Pruebas de Caja Negra 65
3.2.1.1.1 Técnica de la Partición de Equivalencia. 65
3.2.1.1.2 Casos de Prueba 65
3.2.1.2 Resultados de las Pruebas de Caja Negra 66
3.3 Conclusiones Parciales 67
CONCLUSIONES GENERALES 68
RECOMENDACIONES 69
BIBLIOGRAFÍA REFERENCIADA 70
BIBLIOGRAFÍA CONSULTADA 71
GLOSARIO 72

ÍNDICE DE FIGURAS

Figura 1. Patrón de arquitectura Multicapas 28
Figura 2. Estructura de la carpeta Resources, donde se encuentran las carpetas public, la que contiene los archivos css y javascript, y views, donde aparecen las páginas de la aplicación. 31
Figura 3. Estructura de la carpeta Controller, donde se encuentran las clases Controller. 31
Figura 4. Estructura de la carpeta Negocio, donde se encuentra las clases Gtr (Gestore). 32
Figura 5. Estructura de la carpeta Repository, donde se encuentran las clases Repository. 32
Figura 6. Estructura de la carpeta Entity, donde se encuentran las clases entidades mapeadas por el ORM Doctrine. 33
Figura 7. Estructura de la carpeta Presentación, donde se encuentran las clases entidades de presentación. 33
Figura 8. Fragmento del Model de datos del subsistema Administración y Gobierno. Conjunto de tablas involucradas en CU Gestionar Tribunales. 34
Figura 9. Ejemplo de diagrama de clases del diseño del caso de uso Gestionar tribunales. 37
Figura 10. Ejemplo de diagrama de secuencia del caso de uso 38
Figura 11. Diagrama de despliegue 39
Figura 12. Ejemplo de comentario en las funciones. 41
Figura 13. Ejemplo de las entidades de gestión del negocio. 41
Figura 14. Ejemplo de las entidades table model. 42
Figura 15. Ejemplo de las entidades Presentación. 42
Figura 16. Ejemplo de las páginas y plantillas. 42
Figura 17. Ejemplo de rutas. 43
Figura 18. Ejemplo de parámetros y servicios. 43
Figura 19. Ejemplo de clase controladora. 44
Figura 20. Ejemplo de función con más de una palabra en el nombre 44
Figura 21. Ejemplo del funcionamiento del patrón de diseño Factory Method. 49
Figura 22. Ejemplo de petición al contenedor de servicios. 49
Figura 23. Ejemplo de servicio de la clase service.yml 50
Figura 24. Método getRepository() de la clase EntityManager que implementa el patrón Factory. 50
Figura 25. Página decorada con la plantilla global. 51
Figura 26. Uso del patrón creador en la clase Controller GestionarSala. 52
Figura 27. Uso del patrón Experto en la clase GestionarTribunalesController. 53
Figura 28. Llamada a un servicio a través del método get() y el objeto container. 55

Figura 29. Representación en porciento de los resultados obtenidos, agrupados en intervalos definidos tras aplicar la métrica TOC. 58
Figura 30. Representación de los resultados de la evaluación de la métrica TOC en los atributos: responsabilidad y complejidad. 59
Figura 31. Representación de los resultados de la evaluación de la métrica TOC en el atributo reutilización. 59
Figura 32. Representación en porciento de la aplicación de la métrica RC en intervalos definidos. 59
Figura 33. Representación de los resultados de la evaluación de la métrica RC en los atributos: acoplamiento y complejidad de mantenimiento. 60
Figura 34. Representación de los resultados de la evaluación de la métrica RC en los atributos: reutilización y cantidad de pruebas. 60
Figura 35. Representación en porciento de la aplicación de la métrica CCM. 63
Figura 36. Representación en porciento de la aplicación de la métrica APH. 64
Figura 37. Representación del número de no conformidades por iteración. 67

INTRODUCCIÓN

La construcción de la sociedad del conocimiento tiene en el empleo de las TIC's[3] el recurso dinamizador de los procesos de dirección de la vida social en la contemporaneidad. El avance hacia la eficiencia en la economía y en los servicios está signado por la gestión del conocimiento, que no puede concretarse sin una consecuente aplicación de las tecnologías de la información y las comunicaciones en los aspectos sustantivos del funcionamiento de la sociedad.

Respaldar, desde la ciencia y la tecnología, la funcionalidad de cualquier componente del sistema social, es una exigencia de primer orden para garantizar que todos los procesos vitales para el funcionamiento de dicho sistema se cumplan con el máximo de satisfacción para sus usuarios, lo que justifica que la Informática juegue cada día un mayor rol en el tratamiento automatizado de la información. Es por ello que la automatización oportuna de procesos mediante el uso de la Informática no solo favorece la disminución de los costos y el incremento de la productividad del trabajo, sino que preserva la memoria documental, agiliza la toma de decisiones y contribuye a una cultura organizacional más eficiente en la dirección de cualquier proceso social.

Dentro de las funciones públicas asociadas al ejercicio del poder en Cuba, la rama del derecho es parte interesada en la aplicación de los beneficios que supone la informatización de la sociedad, como parte de la actualización tecnológica de sus operaciones, por lo que en la actualidad se desarrolla la Informática Jurídica, como técnica interdisciplinaria que tiene por objeto el estudio e investigación de los conocimientos de la informática aplicables a la recuperación de información jurídica, así como la elaboración y aprovechamiento de los instrumentos de análisis y tratamiento de información jurídica necesarios para lograr la funcionalidad tecnológica de sus sistemas de administración y gobierno. (1)

En virtud de esta necesidad, los Tribunales Populares Cubanos precisan de un sistema informatizado que, desde un enfoque técnico interdisciplinario, contribuyan a hacer eficiente, efectiva y eficaz su gestión, salvando el atraso que se observa al comparar el estado actual de su funcionamiento en este campo, con los logros obtenidos en el primer mundo. En consecuencia, constituye un objetivo del país en el campo de la Informática Jurídica desarrollar soluciones internas a problemas de gran impacto en la sociedad y el funcionamiento de su sistema jurídico.

En función de esta necesidad, la Universidad de las Ciencias Informáticas (UCI) desarrolla sus proyectos productivos, que tienen repercusión nacional, para ofrecer soluciones propias a procesos que se realizan manualmente en los tribunales y otras entidades del sistema jurídico nacional, como contribución científico-tecnológica válida para el proceso de informatización de la sociedad cubana. En

[3] Siglas para: Tecnologías de la Información y las Comunicaciones.

la UCI existen varios centros dedicados al desarrollo de software, con especialización en ramas particulares, los que tienen como principal objetivo suplir las necesidades actuales y futuras tanto de clientes nacionales como internacionales, en función de demandas concretas de soluciones info-tecnológicas. En este campo, el Centro de Gobierno Electrónico (CEGEL) es el encargado de satisfacer necesidades de clientes gubernamentales mediante el desarrollo de productos, servicios y soluciones integrales de alta confiabilidad, calidad, competitividad, fidelidad y eficiencia, a partir de un personal altamente calificado. En la actualidad, constituye una necesidad conjunta del Ministerio de la Informática y las Comunicaciones (MIC) y los Tribunales Populares Cubanos (TPC), impulsar la informatización de los procesos jurídicos del país. Por esta razón, el Tribunal Supremo Popular de Cuba (TSPC) inició un proyecto de cooperación con la UCI el cual tiene como objetivo informatizar los procesos que se gestionan en los TPC. Lo anterior es responsabilidad directa del CEGEL como máximo encargado de la realización y puesta en funcionamiento del Proyecto Sistema de Informatización para la Gestión de los Tribunales Populares Cubanos.

La gestión jurídica en nuestro país descansa, en parte, en el funcionamiento orgánico de los TPC, que constituyen un sistema de órganos estatales cuya función es impartir la justicia que dimana del pueblo en sus tres instancias: suprema, provincial y municipal. En el funcionamiento de los TPC actualmente todos los procesos se llevan a cabo manualmente y con gran dependencia del documento escrito directamente en papel; por ejemplo el registro de documentos, la creación y archivo de los expedientes; cuestiones que posibilitan que se cometan errores de escritura, tachaduras, borrones y pérdida de documentos. Por otra parte los reportes estadísticos se emiten en plazos muy largos para la alta dirección, por lo que no es posible tomar decisiones operativas basadas en información actualizada.

El procesamiento ágil de los datos jurídicos y sus reportes documentales asegura el control de los vencimientos de términos, como criterio clave para garantizar el principio del derecho a un proceso sin dilaciones indebidas**.** En la Ley de los Procedimientos Civil, Administrativo, Laboral y Económico y la Ley Nº 5 del Procedimiento Penal se establecen términos procesales para los actos, estos tienen como fin que los litigios sean resueltos con la mayor rapidez posible.

La organización del trabajo en los tribunales tiene en la función del secretario un componente clave, ya que es el encargado de estar al tanto del vencimiento de los términos procesales y de su notificación al juez. Una vez notificado el vencimiento del término al juez por medio de diligencias, este procede según las disposiciones expresadas por la ley. Las partes también son responsables de conocer el tiempo que les fue impuesto por el tribunal para realizar sus trámites.

La ejecución de este proceso por parte de los secretarios se sustenta en la utilización de los siguientes datos: el término establecido por la ley para el acto procesal en cuestión y el calendario de días no

hábiles. Realizar esta tarea es una ocupación tediosa y requiere un alto grado de desgaste humano por la cantidad de procesos que se atienden a diario. La concepción actual de esta actividad tiene numerosas limitaciones: primero, se enfoca en notificar solo cuando el término ya se ha vencido sin tener en cuenta los términos que están próximos a vencerse y de esta forma alertar a los implicados, segundo el cálculo manual para determinar cuándo se vence el término implica un gran cúmulo de información que debe ser analizada por el secretario lo que lo hace sensible a errores; y tercero se centra en el control del cumplimiento de los términos por las partes.

En virtud de lo anterior, se ha evaluado que el cronograma de señalamientos de los actos públicos es otra de las tareas que resultan de gran dificultad en el funcionamiento actual de los tribunales, debido a que implica el análisis de una gran cantidad de información de todas las materias. La complejidad de este proceso lo hace susceptible a errores humanos y provoca dilación en la tramitación y los procesos jurídicos señalados.

Teniendo en cuenta las deficiencias anteriormente planteadas es que surge como **Problema a resolver:** ¿Cómo informatizar los requisitos acordados con el cliente para el Subsistema Administración y Gobierno de forma tal que se facilite la configuración y administración del sistema integral para la gestión procesal de los tribunales cubanos?

Teniendo, para ello, como **Objeto de estudio:** El proceso de desarrollo de software en la Informática Jurídica de Gestión.

De esta forma, se determina como **Objetivo general:** Desarrollar el módulo Administración y Gobierno perteneciente al Proyecto de Informatización para la Gestión de los Tribunales Populares Cubanos.

Identificándose como **Campo de acción**: El diseño e implementación del módulo Administración y Gobierno perteneciente al Proyecto de Informatización para la Gestión de los Tribunales Populares Cubanos.

Por tanto, se tiene como **Idea a defender:** Realizando el diseño e implementación del módulo Administración y Gobierno para la Gestión de los Tribunales Populares Cubanos, entonces se obtendrá la especificación de los artefactos necesarios para la informatización de los requisitos identificados en el proyecto.

Objetivos específicos:

- ✓ Elaborar el marco teórico de la investigación.
- ✓ Obtener la propuesta de solución.
- ✓ Validar los resultados obtenidos.

Para dar cumplimiento a los objetivos previamente expuestos, se tienen como **Tareas de investigación:**

1. Caracterización del proceso de desarrollo de software y su metodología.
2. Caracterización de la plataforma de desarrollo y de las herramientas utilizadas para diseñar, implementar y probar el sistema.
3. Caracterización de los Paradigmas de Programación.
4. Caracterización y selección de los Patrones de Diseño más factibles para esta propuesta de solución.
5. Realización de los Diagramas de Clases del Diseño.
6. Realización de Casos de uso del diseño (Diagramas de comportamiento).
7. Realización de los Diagramas de Secuencia.
8. Realización del Modelo de datos.
9. Realización del Diagrama de Despliegue.
10. Implementación de los componentes.
11. Diseño de Casos de Prueba.
12. Realización de Pruebas de funcionalidad.
13. Realización de Pruebas al código fuente.

El cumplimiento de estas tareas estuvo respaldado por una serie de métodos de investigación:

Teóricos:

Analítico-Sintético: Esta es la unión de dos métodos el analítico, el cual permitió la división del negocio en general en pequeñas partes para facilitar su estudio, y el sintético que basándose en los elementos analizados con el método anterior ayudó a realizar un resumen de los elementos más importantes del negocio.

Histórico-Lógico: Facilitó el estudio de algunos marcos de trabajo y arquitecturas de software existentes en el mundo, así como identificar los principales patrones de diseño e implementación más eficaces en el proceso de desarrollo de software.

Modelación: Este método es uno de los más importantes debido a que su uso permitió la creación de modelos, dígase diagramas, que explicitarán el flujo de acciones a seguir y las relaciones entre los componentes en los procesos del módulo Administración y Gobierno.

Empíricos:

Observación: Se usó en el diagnóstico del problema a investigar y es de gran utilidad en el diseño de la investigación.

Medición: Se desarrolló con el objetivo de obtener la información numérica acerca de la calidad del producto resultante mediante la realización de pruebas al sistema.

Capítulo 1: Fundamentación teórica

Introducción

En el presente capítulo se realiza un estado del arte de los sistemas informáticos para la gestión judicial similares al que se desarrollará y además se caracterizan las tecnologías y metodologías que se utilizan durante el desarrollo del sistema.

1.1. Definiciones relacionadas con el objeto de estudio y el campo de acción

Con el objetivo de lograr un entendimiento del tema a tratar en la investigación se hace necesario tener conocimiento sobre una serie de conceptos:

Administración: La administración es "un proceso que consiste en prever, organizar, dirigir, coordinar y controlar". (2)

Informática Jurídica: "Conjunto de aplicaciones de la informática en el ámbito jurídico; es una técnica interdisciplinaria que tiene por objeto el estudio e investigación de los conocimientos aplicables a la recuperación de información jurídica, así como la elaboración y aprovechamiento de los instrumentos de análisis y tratamiento de información jurídica, necesaria para lograr dicha recuperación." (1)

Informática Jurídica de Gestión: "Es la aplicación de la informática a las tareas cotidianas de abogados, jueces, peritos, etc. a través del uso de computadoras y programas para realizar tareas de procesamiento de textos, de almacenamiento de datos, para efectuar comunicación mediante redes, etc." (3)

1.2. Módulo Administración y Gobierno

Los sistemas informáticos actuales son cada vez más flexibles y adaptables en dependencia de las necesidades de los clientes, por lo general están compuestos por varios módulos independientes que realizan determinadas funciones y todos ellos unidos por una estructura organizativa llamada módulo de administración para que el sistema funcione de forma ordenada, segura y consistente. Por esta razón generalmente la mayoría de los sistemas cuentan con un rector informático, el cual se encarga de permitir el control de la estructura organizativa del sistema, garantizando eficiencia en todos sus módulos y procesos, posibilitando la centralización de algunos procedimientos y configuraciones globales que garantizan el buen funcionamiento del mismo. Manejando diferentes conceptos en dependencia de las necesidades finales del cliente, por ejemplo: usuarios, puestos de trabajo, gestión de configuración de archivos y variables, control del flujo de eventos, entre muchos otros. (4)

En el módulo Administración y Gobierno se agrupan las funcionalidades relacionadas con la gestión de los diferentes tribunales con sus respectivas salas y las materias que atienden, la configuración del sistema, la gestión de usuarios, cada uno con roles predefinidos que podrán ser modificados en los respectivos tribunales para los que fueron creados, y gestión de fechas de trabajo a los jueces legos, el calendario de señalamiento con los servicios de vencimiento de términos, el calendario con los días no hábiles, la configuración de roles de acceso y restricción de acciones a usuarios. Además incluye los nomencladores donde se definen un conjunto de datos necesarios para llevar a cabo los procesos de los TPC, como por ejemplo el tipo de corrección disciplinaria, tipo de moneda, tipo de banco, la gestión de los diferentes ministerios, bufetes y fiscalías, entre otros. Además, se podrán visualizar las trazas recogidas en el sistema, mostrándose las acciones en correspondencia al usuario que realizó la acción. Se gestionará en este módulo lo referente a los nomencladores y se brindarán servicios que el resto de los módulos se encargarán de consumir.

Uno de los procesos llevados a cabo por el módulo es la asignación de roles, una vez creado un rol este puede ser asignado a un usuario en el momento de su creación. Permite además la activación y desactivación de usuarios, que será crítica para la seguridad del sistema ya que se mantendrá el registro de todos los usuarios que han trabajado en el mismo. Por ejemplo si un usuario cambiara de centro de trabajo el mismo sería desactivado en lugar de eliminar sus datos y las operaciones que realizó.

El desarrollo del módulo Administración y Gobierno dentro del proyecto Informatización de la Gestión de los TPC les proporcionará una serie de beneficios como los listados a continuación:

- ✓ La gestión de los términos de los trámites que se realizan en el tribunal alertando a los usuarios (jueces, abogados y fiscales) cuando un trámite está a punto de declararse extemporáneo.
- ✓ Existencia de un calendario de señalamientos de actos públicos donde los usuarios podrán comprobar la planificación de los mismos.
- ✓ Existencia de un calendario de días no hábiles, el cual hará más sencillo el proceso del cálculo del vencimiento de términos.
- ✓ La notificación automática de un conjunto de eventos de interés para los usuarios.
- ✓ Mecanismos de búsquedas de información.
- ✓ Configuración y asignación de fechas laborales en un tribunal para los Jueces Legos.
- ✓ Gestión, manejo y fácil acceso a información especializada con la consulta a los nomencladores.
- ✓ Ahorro de recursos materiales y humanos.
- ✓ Mejora de la seguridad en cuanto al nivel de acceso a documentos y a funcionalidades de las persona que laboran en un tribunal. (5)

Teniendo en cuenta lo anteriormente planteado la implementación de un sistema informático que optimice los procesos que se realizan en los TPC supondrá una mejora significativa en los servicios que estos brindan repercutiendo positivamente en la sociedad cubana.

1.3. Estado del arte de sistemas informáticos relacionados con el campo de acción

El proyecto del cual es parte esta investigación centra su desarrollo en la rama de la Informática Jurídica de Gestión por las aspiraciones y características del mismo, pues su principal objetivo es automatizar los procedimientos judiciales en cada instancia de los tribunales cubanos.

La introducción de las TIC´s en la administración de justicia puede permitir una justicia de mayor calidad y, al mismo tiempo, abierta, transparente y próxima al ciudadano. En la actualidad se promueve el uso de la informática en la rama del derecho, y aunque su avance es lento, existen numerosas soluciones informáticas en el mundo que hacen uso de sistemas informáticos para la gestión procesal, la mayoría de estas aplicaciones están enfocadas en la agilización de la tramitación procesal, sin embargo hasta el momento no existe una solución informática que facilite íntegramente el desarrollo de un proceso judicial similar a los que tienen lugar en los TPC debido a las diferencias en las leyes de cada país.

1.3.1. Sistemas estudiados en el mundo

Infolex, Software de gestión jurídica (España): Este software es realizado por la empresa Jurisoft la cual es líder en el sector de la Informática Jurídica. La misma se dedica a la creación de software aplicado al mundo del derecho lo que le permite conocer perfectamente las peculiaridades técnicas y legales que diferencian a la información jurídica y económica. Uno de sus productos es Infolex el cual es líder desde 1988 y decano del software de Gestión Jurídica. Este software trabaja en varios módulos: expedientes, seguimiento de expedientes de actuaciones, escritos, parte diario, procesos diarios y gestión de cobros, minutación-facturación, listín electrónico, agenda de despacho, contabilidad, tributación, listados e informes, bases de datos, herramientas. A pesar de las numerosas funcionalidades de este sistema, no es posible su utilización en los TPC debido a que los procesos que maneja no se ajustan a los que se necesitan informatizar, fue diseñado y construido para España por lo que sus procesos se rigen por las legislaciones de dicho país. Sin embargo es válido destacar que se tomaron ideas del mismo para la solución a desarrollar, por ejemplo, la forma en que implementan la seguridad, Infolex permite la creación de un esquema de seguridad dentro del despacho y el control de acceso de los usuarios a las distintas funciones del programa a partir de una clave personal, lo que garantiza la confidencialidad e integridad de la información que se maneja, aspecto de gran importancia para los clientes.

LEGAL NET, Sistema de Gestión de Contratación Pública y Privada (Colombia): Este sistema apoya cualquier tipo de proceso de contratación, desde la etapa precontractual, hasta la liquidación, realizando control sobre el valor del contrato, pagos, causación y pago de impuestos, duración y término de vencimiento, informa la necesidad de efectuar prórrogas, lleva el registro de garantías y control de vencimientos de vigencias; El usuario puede definir el tipo de proceso de contratación y sus etapas: preliminar, precontractual, elaboración del contrato, ejecución, prórrogas y liquidación. Permite programar agendas y avisos de alerta para controlar el vencimiento de términos por actividades. Presenta opción para el registro de las modificaciones al contrato, actualizando la información precedente y llevando el registro histórico. Los contratos y documentos son generados automáticamente a partir de plantillas predefinidas, los cuales igualmente pueden ser asociados mediante su digitalización.

Está modelado sobre un motor de Workflow, lo que permite la definición de las etapas de gestión contractual que aplican al interior de la organización en la cual se use. Además apoya integralmente los procesos de licitación, contratación, ejecución y liquidación, permitiendo su integración con los módulos de presupuesto y pagos. (6)

Este sistema está enfocado específicamente al proceso legal de una contratación lo que representa una parte muy pequeña dentro del sistema legal cubano por lo que no es la solución para los TPC. Además está desarrollado con software propietario lo que imposibilita la adaptabilidad a nuestro sistema legislativo.

Gedex, software jurídico (España): Realiza el seguimiento completo de los expedientes de un despacho, bufete o departamento jurídico. Uno de los más utilizados en España y Latinoamérica; sus inicios se remontan al año 1996 y actualmente cuenta con un gran número de licencias vendidas a diferentes despachos jurídicos. Gestiona expedientes así como sus contactos asociados, almacena, recupera y gestiona la documentación legal. Ofrece un sistema de contraseñas, con el que puede limitar el acceso a la información, ocultar expedientes y contactos a ciertos pasantes o empleados. El sistema Gedex está básicamente enfocado al seguimiento de los expedientes y la documentación en sentido general, la manera de realizar los procesos no se ajusta a cómo se hace en Cuba ya que este sistema tiene sus fundamentos en las legislaciones de España y no se ajusta a las cubanas, es decir el flujo por el que deben tramitar los expedientes no es el mismo, razón por la que no es factible para ser usado en los TPC.

Lex-Doctor, Sistema para la gestión jurídica (Argentina): Este completo sistema es utilizado en la amplia mayoría de los estudios jurídicos y asesorías letradas informatizados de la Argentina

desarrollado por Sistemas Jurídicos SRL, empresa que tiene un importante posicionamiento en el segmento de los sistemas aplicados a la actividad jurídica en Latinoamérica. Su tecnología de administración de datos, permite manejar grandes volúmenes de información, y organizar grupos de trabajo operando en redes de gran cantidad de terminales.

Principales funcionalidades:

- ✓ **Gestión de expedientes**: Procesos judiciales, extrajudiciales, mediaciones y todo tipo de expediente de índole jurídico. Partes, letrados, agendas, pruebas, gestiones, movimientos, audiencias, vencimientos. Manejo total de la procuración.
- ✓ **Gestión económica:** Cuentas por cliente, por expediente, y otros tipos de cuenta definibles por el usuario. Actualizaciones y liquidaciones, con conversión de monedas. Liquidaciones individuales o masivas.
- ✓ **Gestión documental**: Confección automatizada de escritos, cédulas, oficios, mandamientos, telegramas, cartas, y otros documentos, con posibilidad de confección seriada. Almacenamiento de documentos creados por el sistema y por otros programas instalados en la PC.
- ✓ **Reportes y estadísticas**: Confección de listados e informes, con diseños programados o propios, y con posibilidad de exportar a distintos formatos. Evaluación estadística gráfica.
- ✓ **Acceso remoto:** Opcionalmente, permite operar a través de Internet; así, disponiendo de una conexión y una PC, se puede trabajar en línea con la oficina desde cualquier lugar.

Lex-Doctor a pesar de ser un sistema con mucho prestigio y aceptación presenta diferentes inconvenientes para su utilización por parte de los TPC puesto que el mismo está desarrollado sobre software propietario, impidiendo las posibilidades de modificación y adaptación al medio nacional, además centra su trabajo sobre la gestión documental fundamentalmente, y para su utilización es necesario comprar la licencia de uso.

1.3.2. Sistemas estudiados en Cuba

Cuba no está ajena al avance de las tecnologías aplicadas al Derecho, en este sentido se han hecho intentos de informatizar las materias Penal y Económico con dos soluciones informáticas que no han cumplido las expectativas para las que fueron creadas.

SisProp: Sistema para la tramitación de procesos penales: Sistema informático desarrollado en la provincia de Villa Clara en el 2008. La propuesta inicial fue que abarcara la instancia suprema y provincial de la materia penal, desarrollándose solamente la tramitación de los procesos en la última instancia. Facilita la tramitación de los procesos penales, con agilidad y precisión, en las entidades del

sistema de tribunales populares del país. Dos de las características fundamentales del sistema son su seguridad dada la naturaleza de la información que se maneja, y la flexibilidad con respecto a cualquier modificación que pudiera sufrir la Ley Procesal Penal.

Principales deficiencias:

- ✓ No capta ningún dato de la fase judicial de la tramitación y decisión del tribunal.
- ✓ No aporta estadística, ni información alguna.
- ✓ No posee una exhaustiva validación de los datos.
- ✓ No se trabajó con un NIP[4] de cada proceso.
- ✓ Programado en Delphi, corre sobre SQL Server por lo que no es compatible con el software libre en el que se está programado los sistemas generales de cada materia judicial en la solución informática SIT.

Este sistema desarrollado en Cuba, no cumple con las expectativas para las que fue creado, no es posible integrarlo a la solución informática SIT debido a que las tecnologías de desarrollo no son compatibles con las definidas para el SIT, ya que estas son más avanzadas. Esta aplicación al no registrar ningún dato de la tramitación no emite informes estadísticos, quedando por debajo de las expectativas que se tienen con la creación del SIT.

SisEco: Sistema Económico: Sistema informático desarrollado en Ciudad de la Habana en el 2002, concebido para el área de la estadística en el procedimiento Económico. El sistema cuenta con funcionalidades muy básicas y es lento en cuanto a tiempo de respuesta. En él se insertan los documentos radicados manualmente y las salvas diariamente se guardan en disquetes. La secretaria de estadística recoge el libro de radicación de escritos (LRE) e inserta los datos en la aplicación y cada cinco años borra la información, quedando solamente asentada en los libros.

Principales deficiencias:

- ✓ Inserta documentos radicados manualmente. La secretaria de estadística recoge el libro LRE2 e inserta los datos en la aplicación y cada cinco años borra los datos de los años anteriores quedando solamente la información asentada en los libros.
- ✓ Las salvas se guardan en disquetes. (7)

Al igual que la anterior, esta aplicación no cumplió las expectativas iniciales de su creación. Es lento, característica que no va aparejada con la estadística en tiempo real. Prácticamente no informatiza nada, pues la radicación se realiza de forma manual, las salvas se guardan en disquetes, técnica que ha quedado obsoleta en nuestros días, la información almacenada es borrada cada cinco años, algo

[4]Número de Identificación Permanente o General.

incomprensible para un sistema judicial para el cual mantener registros es primordial, ya que la información es el mecanismo principal para la justicia, por lo que tampoco es posible integrar la solución al SIT.

1.4. Fundamentación de las herramientas, metodología y tecnologías a utilizar

1.4.1. Metodología de Desarrollo de Software: RUP[5]

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software.

Es como un libro de recetas de cocina, en el que se van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Además detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla. (8)

Las metodologías de desarrollo de software surgen a partir de la necesidad de organizar el trabajo de todas las partes de un equipo de desarrollo de software, estás se basan en técnicas, procedimientos, herramientas y soporte documental. Existen numerosas metodologías, clasificadas en dos grupos:

- ✓ **Metodologías Pesadas**: Son las más tradicionales, se centran en la definición detallada de los procesos y tareas a realizar, herramientas a utilizar, y requiere una extensa documentación, ya que pretende prever todo de antemano. Este tipo de metodologías son más eficaces y necesarias cuanto mayor es el proyecto que se pretende realizar respecto a tiempo y recursos que son necesarios emplear, donde una gran organización es requerida. (9)
- ✓ **Metodologías Ligeras/Ágiles**: Están especialmente indicadas en proyectos con requisitos poco definidos o cambiantes. Estas metodologías se aplican bien en equipos pequeños que resuelven problemas concretos, lo que no está reñido con su aplicación en el desarrollo de grandes sistemas, ya que una correcta modularización de los mismos es fundamental para su exitosa implantación. Dividir el trabajo en módulos abordables minimiza los fallos y el coste. (10)

1.4.2. Proceso Unificado de Rational

RUP es un modelo de software que permite el desarrollo de software a gran escala, mediante un proceso continuo de pruebas y retroalimentación, garantiza el cumplimiento de ciertos estándares de calidad. (11)

El Proceso Unificado tiene tres características distintivas. Estas características son:

[5] Siglas en inglés de Proceso Unificado de Rational (Rational Unified Process).

- ✓ **Dirigido por Casos de Uso**: El proceso utiliza Casos de Uso (CU) para manejar el proceso de desarrollo desde la Incepción[6] hasta el Despliegue[7].
- ✓ **Centrado en la Arquitectura**: El proceso busca entender los aspectos estáticos y dinámicos más significativos en términos de arquitectura de software. La arquitectura se define en función de las necesidades de los usuarios y se determina a partir de los Casos de Uso base del negocio.
- ✓ **Iterativo e Incremental**: El proceso reconoce que es práctico dividir grandes proyectos en proyectos más pequeños o mini-proyectos. Cada mini-proyecto comprende una iteración que resulta en un incremento. Una iteración puede abarcar la totalidad de los flujos del proceso. Las iteraciones son planificadas en base a los Casos de Uso. (12)

RUP identifica a los flujos de trabajo fundamentales que se producen durante el proceso de desarrollo de software. Estos flujos incluyen el Modelado de Negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba y Despliegue[8]. Además establece tres etapas para la realización de la documentación necesaria en el desarrollo de un sistema: Configuración y administración del cambio, Administración del proyecto y Ambiente.

Debido a la magnitud y complejidad del proyecto de Informatización para la Gestión de los Tribunales Populares Cubanos, la capacidad de mitigar riesgos identificados en cada iteración en iteraciones posteriores y la posibilidad de la gestión de la complejidad el equipo de arquitectura define en el documento de arquitectura el uso de RUP como metodología de desarrollo de software.

A pesar de que RUP define la documentación que es necesaria realizar en la construcción de un software el proyecto de Informatización para la Gestión de los Tribunales Populares Cubanos, centrado en el objetivo de la UCI de mejorar y asegurar la calidad de todos los productos desarrollados, se rige por las plantillas especificadas en el Expediente de Proyecto del Programa de Mejora. Estas plantillas son el resultado de la experiencia práctica y se guían por el modelo CMMI.

1.4.3. Herramienta de Ingeniería de Software Asistido por Ordenadores (CASE) para el modelado

1.4.3.1. Visual Paradigm 8.0

Visual Paradigm es una herramienta para desarrollo de aplicaciones utilizando modelado UML[9], el cual es ideal para ingenieros de software, analistas de sistemas y arquitectos de sistemas que están

[6] Primera de las cuatro fases del ciclo de vida de un sistema definido por RUP.
[7] Último flujo de trabajo de proceso del ciclo de vida de un sistema definido por RUP.
[8] Flujos de trabajo de proceso del desarrollo de un sistema según RUP.
[9] Siglas en inglés de Unified Modeling Language (Lenguaje de Modelado Unificado).

interesados en el desarrollo de sistemas de gran tamaño y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. Visual Paradigm ofrece ventajas como:

- ✓ Navegación intuitiva entre la escritura del código y su visualización.
- ✓ Potente generador de informes en formato PDF[10]/HTML[11].
- ✓ Documentación automática Ad-hoc[12].
- ✓ Ambiente visualmente superior de modelado.
- ✓ Sofisticado diagramador automático de diseños.
- ✓ Sincronización de código fuente en tiempo real.

Tiene una interfaz muy intuitiva y es de fácil aprendizaje para los desarrolladores. Permite además hacer descripción de los casos de uso dando una gran variedad de plantillas predeterminadas con la posibilidad de personalizarlas. Fue definido utilizar Visual Paradigm 8.0 como herramienta CASE en el proyecto TPC principalmente porque la UCI tiene licencia para su uso y entre las ventajas que ofrece actualmente cumple con las políticas de migración a software libre en Cuba, ya que es una herramienta multiplataforma que se puede utilizar tanto en Linux como en Windows.

1.4.4. Lenguaje de programación para el desarrollo Web: PHP 5.3.3

PHP[13] es un lenguaje de código libre "del lado del servidor" (esto significa que funciona en un servicio remoto que procesa la página web antes de que se muestre en el navegador) diseñado para desarrollar páginas web dinámicas. Es uno de los primeros lenguajes del lado del servidor que se puede incluir en el código HTML en vez de realizar una llamada a un archivo externo para procesar datos. Entre sus principales características sobresalen: que es gratuito, se integra de manera sencilla con múltiples gestores de bases de datos y tiene un gran número de funciones previamente definidas como por ejemplo funciones para el trabajo con fechas, arreglos y cadenas de texto, fueron utilizadas en la implementación del módulo Administración y Gobierno, entre otras. En la UCI existe una gran comunidad de desarrolladores que utilizan este lenguaje debido a la gran diversidad de ventajas que ofrece.

1.4.5. Sistema gestor de base de datos: PostgreSQL 9.1

[10] Siglas en inglés de Portable Document Format (Formato de Documento Portable): es un formato de almacenamiento de documentos digitales.
[11] Siglas en ingles de HyperText Markup Language (Lenguaje de marcado de hipertexto), hace referencia al lenguaje de marcado predominante para la elaboración de páginas web.
[12] Se usa pues para referirse a algo que es adecuado sólo para un determinado fin. En sentido amplio, ad hoc puede traducirse como específico o específicamente.
[13] Acronimo recursivo de Hypertext Preprocessor (Preprocesador de hipertexto).

PostgreSQL es un Sistema de Gestión de Base de Datos de Objetos Relacional (ORDBMS). Es ampliamente considerado como una de las alternativas de sistemas de base de datos de código abierto que se ha ganado una sólida reputación de confiabilidad, integridad de datos y corrección. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multi-hilos para garantizar la estabilidad del sistema. Es un sistema seguro y puede soportar grandes volúmenes de datos. (13)
Características fundamentales:

- ✓ Transacciones: Permiten el paso entre dos estados consistentes manteniendo la integridad de los datos.
- ✓ Integridad referencial: PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.
- ✓ Bloqueos de tablas y filas: PostgreSQL ofrece varios modos de bloqueo para controlar el acceso concurrente de los datos en tablas.
- ✓ Restricciones (constraints) y disparadores (triggers): Tienen la función de mantener la integridad y consistencia de la base de datos.
- ✓ Múltiples tipos de datos predefinidos: Como todos los manejadores de base de datos, PostgreSQL implementa los tipos de datos definidos para el estándar SQL3 y aumenta algunos datos.
- ✓ Soporte de tipos y funciones de usuarios: PostgreSQL soporta operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario. Incorpora una estructura de datos de arreglo.
- ✓ Interfaz con diversos lenguajes, entre ellos PHP.

1.4.6. Marcos de trabajo(frameworks)

Es una estructura tecnológica de soporte definida, normalmente con artefactos o módulos de software, a través de la cual otro proyecto de software puede ser más fácilmente organizado y desarrollado. Comúnmente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

1.4.6.1. Symfony 2.1

Symfony es un framework PHP basado en la arquitectura MVC (Modelo-Vista-Controlador). Fue desarrollado para ser utilizado sobre la versión 5 de PHP ya que hace ampliamente uso de la orientación a objetos que caracteriza a esta versión y desde la versión 2 de Symfony se necesita mínimamente PHP 5.3.3.

El concepto de Symfony es no reinventar la rueda, por lo que reutiliza conceptos y desarrollos exitosos de terceros y los integra como librerías para ser utilizados por nosotros. Entre ellos encontramos que integra plenamente uno de los frameworks ORM más importantes dentro de los existentes para PHP llamado Doctrine, el cual es el encargado de la comunicación con la base de datos, permitiendo un control casi total de los datos sin importar si estamos hablando de MySQL, PostgreSQL, SQL server, Oracle, entre otros motores ya que la mayoría de las sentencias SQL no son generadas por el programador sino por el mismo Doctrine.
Otro ejemplo de esto es la inclusión del framework Twig. (14)

1.4.6.2. Bootstrap 2.0

Bootstrap es un framework desarrollado por Twitter que simplifica el proceso de creación de diseños web combinando CSS y JavaScript. La mayor ventaja es que podemos crear interfaces que se adapten a los distintos navegadores apoyándonos en un framework potente con numerosos componentes webs que nos ahorrarán mucho esfuerzo y tiempo.
Bootstrap ofrece una serie de plantillas CSS y ficheros Javascript que nos permiten integrar el framework de forma sencilla y potente en nuestros proyectos webs. Entre sus principales características están:

- ✓ Permite crear interfaces que se adapten a los diferentes navegadores, tanto de escritorio como tablets y móviles a distintas escalas y resoluciones.
- ✓ Se integra perfectamente con las principales librerías Javascript, por ejemplo JQuery.
- ✓ Ofrece un diseño sólido y estándares como CSS3/HTML5.
- ✓ Funciona con todos los navegadores, incluido Internet Explorer usando HTML Shim para que reconozca los tags HTML5.
- ✓ Dispone de distintos layout predefinidos con estructuras fijas a 940 píxeles de distintas columnas o diseños fluidos.

1.4.6.3. jQuery

jQuery es una librería JavaScript rápida y concisa que simplifica la forma de interactuar con el documento HTML, manejo de eventos, animación, y las interacciones Ajax para el desarrollo web rápido. jQuery es software libre y de código abierto, lo que significa que puede ser utilizado en proyectos libres. Entre sus principales ventajas se encuentran:

- ✓ Alterar el contenido de un documento.
- ✓ Responder a la los eventos de un usuario.
- ✓ Recuperar información de un servidor sin refrescar una página.

- ✓ Animar cambios realizados a un documento.

1.4.6.4. Doctrine

Doctrine es un potente y completo sistema ORM para PHP 5.2 o mayor, con una capa de abstracción de la base de datos (DBAL del inglés Database Abstraction Layer) incorporada. Entre otras ventajas ofrece la posibilidad de exportar una base de datos existente a sus clases correspondientes y también a la inversa, es decir convertir clases a tablas de una base de datos. Su principal ventaja radica en poder acceder a la base de datos utilizando la programación orientada a objetos (POO). Doctrine utiliza el patrón Active Record para manejar la base de datos, tiene su propio lenguaje de consultas de datos (DQL) y trabaja de manera rápida y eficiente. Es fácilmente integrado a los principales frameworks de desarrollo utilizados actualmente.

1.4.7. Entorno de Desarrollo Integrado (IDE): NetBeans 7.2

Un IDE es una aplicación de software que ofrece servicios integrales a los programadores de computadoras para el desarrollo de software. Un IDE está compuesto por un editor de código, un compilador o intérprete, un depurador y un constructor de interfaz gráfica. NetBeans es un IDE especialmente diseñado para el desarrollo de aplicaciones en Java, pero que acepta otros lenguajes de programación. Consta de una gran base de usuarios y una comunidad en constante crecimiento, lo que le ha permitido, al igual que muchos otros sistemas libres, el progreso paulatino de sus prestaciones y la eliminación de errores de programación (bugs) que pudiesen existir.

Ventajas del uso del NetBeans:

- ✓ La plataforma NetBeans permite que las aplicaciones se desarrollen a partir de un conjunto de módulos o componentes de software.
- ✓ NetBeans IDE es fácil de instalar y de uso instantáneo, se ejecuta en varias plataformas incluyendo Windows y Linux.

1.4.8. Servidor web: Apache 2.2.22

Un servidor web es una aplicación, programa o software instalado en una computadora que se mantiene a la escucha de peticiones que le realizará un cliente, y que responde a estas peticiones a través de una página web que será mostrada en el navegador del cliente o a través de un mensaje de error si ha ocurrido alguno. Apache es el servidor web de millones de servidores en el mundo debido a su robustez, configurabilidad y estabilidad.

Algunas de las características de Apache son:

- ✓ Corre en diferentes Sistemas Operativos (Windows, Linux).
- ✓ Es una tecnología gratuita de código abierto.
- ✓ Es un servidor altamente configurable de diseño modular.

- ✓ Apache permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor.
- ✓ Tiene una alta configurabilidad en la creación y gestión de logs.

1.4.9. Patrones de arquitectura

1.4.9.1. Modelo Vista Controlador (MVC)

Modelo–Vista–Controlador es un patrón de arquitectura de software. Separa conceptualmente la representación visual de la aplicación, las acciones que intercambian datos y el modelo de negocio y su dominio. En el Sistema de Informatización para la Gestión de los Tribunales Populares Cubanos se identifica con 3 elementos diferentes: la vista implementada en Java Script o HTML a través de las plantillas Twig reside del lado del cliente en tiempo de ejecución, el Controlador, y el Modelo que junto al controlador reside del lado del servidor, la interacción entre la vista y el controlador se realiza a través de una solicitud AJAX o peticiones HTML y la respuesta dada por el controlador puede encontrarse en JSON, XML o HTML según corresponda la solicitud. La gran ventaja del MVC es desacoplar la vista del modelo y así lograr una mayor reusabilidad.

Ventajas del MVC

- ✓ Es posible tener diferentes vistas para un mismo modelo (ej. representación de un conjunto de datos como una tabla).
- ✓ Es posible construir nuevas vistas sin necesidad de modificar el modelo subyacente.
- ✓ Proporciona un mecanismo de configuración a componentes complejos muchos más tratable que el puramente basado en eventos (el modelo puede verse como una representación estructurada del estado de la interacción). (15)

1.4.9.2. Modelo Cliente-Servidor

La tecnología Cliente/Servidor es el procesamiento cooperativo de la información por medio de un conjunto de procesadores, en el cual múltiples clientes, distribuidos geográficamente, solicitan requerimientos a uno o más servidores centrales. (16)

El principio fundamental de este modelo son los servicios, en el cual el cliente toma un papel de consumidor de servicios y el servidor juega el rol de proveedor de servicios. Los 3 elementos fundamentales en este modelo son: el proceso cliente que es el que comienza la comunicación, el proceso servidor que es el que escucha pasivamente las peticiones de servicios para luego dar una respuesta y el middleware[14] que no es más que la interfaz que permite la comunicación entre el cliente y el servidor para poder intercambiar mensajes. Normalmente existe uno o más servidores donde se

[14] Software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones.

encuentra el servicio que se ofrece y varios clientes que consumen dicho servicio, en el Sistema de Informatización para la Gestión de los Tribunales Populares Cubanos este estilo responde al hecho de que se trate de una aplicación web y se concreta con un servidor de bases de datos (PostgreSQL) un servidor web (con función de servidor de aplicaciones, Apache 2) y varios clientes que acceden al sistema a través de un navegador.

1.4.9.3. Multicapas

La arquitectura multicapas está compuesta por 3 tipos de capas: Interfaz o Presentación, Reglas de negocios y Gestión de Datos

Figura 1. Patrón de arquitectura Multicapas

La primera capa (Interfaz o Presentación), es la página inicial donde el usuario observa el diseño de la aplicación.

La segunda capa (Reglas de negocio), es la capa intermedia que se encarga de manejar todas las peticiones realizadas a la tercera capa (Datos) a través de la interfaz de usuario de la aplicación.

La tercera capa (Gestión de Datos), esta es la capa más importante del programa, ya que maneja la información basada en una plataforma potente permitiendo así una consistencia en la información. (17)

La arquitectura multicapas se caracteriza por organizar los componentes del sistema en capas con responsabilidades bien definidas y donde las capas del nivel más alto invocan los servicios de las del nivel inferior.

Conclusiones parciales

En este capítulo se realizó un estudio de diferentes sistemas existentes en el mundo que realizan procesos similares a los que se desean desarrollar con la investigación, se exponen características de los mismos lo que aporta conocimientos e ideas para la realización de un sistema. Con el estudio del arte se concluye que no existe en el mundo una solución informática que facilite el desarrollo de un

proceso judicial como el que tiene lugar en los Tribunales Populares Cubanos, por lo que es una necesidad la creación de un sistema que responda a las necesidades de los mismos.

Se realizó un análisis de las diferentes herramientas informáticas que se van a utilizar en el desarrollo del sistema, las cuales se listan a continuación y fueron previamente definidas en el Documento de Arquitectura del Proyecto Sistema de Informatización para la Gestión de los Tribunales Populares Cubanos.

- ✓ Lenguaje de programación: PHP 5.3.3
- ✓ Marco de trabajo: Symfony 2.0
- ✓ Herramienta CASE para el modelado: Visual Paradigm 8.0
- ✓ Servidor web: Apache 2.2.22
- ✓ Sistema Gestor de Base de Datos: PostgreSQL 9.1
- ✓ IDE de desarrollo: NetBeans 7.2

Capítulo 2: Propuesta de solución

Introducción

En este capítulo se realizará una propuesta de solución técnica a la investigación. Se presenta el Modelo de diseño conformado por los Diagramas de clases y los Diagramas de secuencia (Diagrama de interacción), el Modelo de implementación con los Diagramas de componentes y el Diagrama de despliegue. Se exponen además los patrones de diseño utilizados en la solución propuesta, los estándares de codificación definidos para la implementación y el diseño de la base de datos mediante los modelos físico y lógico.

2.1 Arquitectura del Sistema

No es algo nuevo que ninguna de las definiciones de Arquitectura de Software (AS) esté respaldada unánimemente por la totalidad de los arquitectos. La cifra de definiciones existentes en el mundo alcanza un orden de tres dígitos.

La definición “oficial” de AS se ha acordado que sea la que brinda el documento de IEEE Std 1471-2000, adoptada también por Microsoft, que reza así:

La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución.

El desarrollo de SITPC está basado en una arquitectura por capas(n-layers) permitiendo dividir los problemas a resolver y que cada capa contenga solo las funcionalidades relacionadas con sus tareas, esto proporciona una alta reutilidad y un bajo acoplamiento. Se hace uso del patrón arquitectónico Modelo–Vista–Controlador (MVC), el cual permite la reutilización e independencia entre las capas, además permite que se puedan realizar cambios en las mismas sin tener que modificar las otras capas, facilita la estandarización, la utilización de los recursos y la administración.

2.1.1 Vista

En esta capa se encuentra todo lo que se refiere a la visualización de la información, el diseño, colores, estilos y la estructura visual de nuestras páginas. En Symfony 2, el framework utilizado para desarrollar la solución, queda evidenciada esta capa a través del uso del motor de plantillas Twig y el framework Jquery, que en conjunto con los archivos CSS y JavaScript son los encargados de crear las páginas de la aplicación.

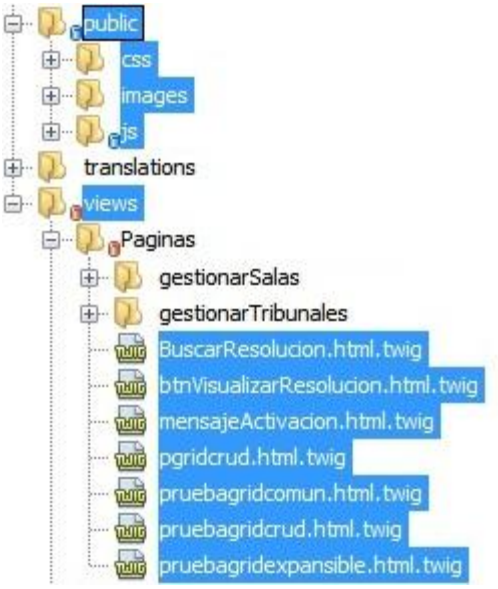

Figura 2. Estructura de la carpeta Resources, donde se encuentran las carpetas public, la que contiene los archivos css y javascript, y views, donde aparecen las páginas de la aplicación.

2.1.2 Controlador

La responsabilidad de esta capa es procesar y mandar a mostrar los datos obtenidos por la capa de acceso a datos. Es decir, trabaja de intermediario entre la capa de presentación y la capa de acceso a datos. En el SITPC el uso de esta capa se observa a través de las clases Controller que son las encargadas de realizar el procesamiento de los datos y lógica de presentación asociada a las peticiones realizadas desde la vista por los clientes.

Figura 3. Estructura de la carpeta Controller, donde se encuentran las clases Controller.

2.1.3 Modelo

En el SITPC esta capa está dividida en 2, la capa de acceso a datos y la de abstracción de datos.

2.1.3.1 Capa de acceso a datos

Esta capa tiene la responsabilidad de conectar la capa Controladora con el gestor de base de datos y manejar la lógica del negocio. Las clases Repository forman parte de esta capa, en ellas es donde se

encuentran las consultas más complejas a la base de datos. Estas clases se encuentran en la carpeta vendor de Symfony2 que es donde se ubican las dependencias de terceros. En esta carpeta estará el módulo Común (ComunBundle) que va a recoger las clases Repository junto con otras clases necesarias para el funcionamiento de todos los módulos. Las clases Gtr son las encargadas del manejo de la lógica del negocio, estas recibirán a través del controlador toda la información enviada desde la vista. En esta capa también se encuentra el ORM Doctrine que posibilita la separación de la aplicación respecto al gestor de base de datos mediante su lenguaje propio de consultas DQL.

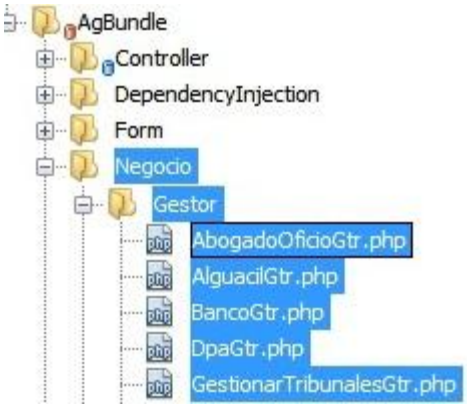

Figura 4. Estructura de la carpeta Negocio, donde se encuentra las clases Gtr (Gestore).

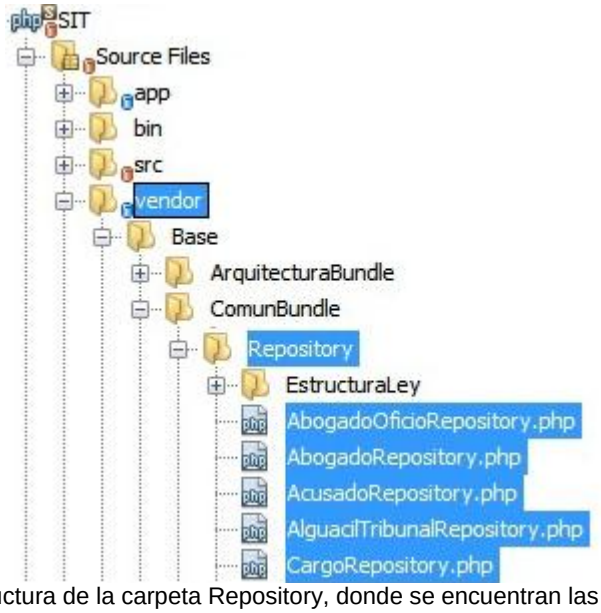

Figura 5. Estructura de la carpeta Repository, donde se encuentran las clases Repository.

2.1.3.2 Capa de abstracción de datos

Esta capa está formada por las clases entidades y las entidades de presentación. Las clases entidades no son más que una representación de las tablas de la base de datos que son mapeadas previamente por el ORM Doctrine y las entidades de presentación representan una combinación entre dos o más entidades. Las clases entidades al igual que las clases Repository se encuentran en el módulo Común (ComunBundle) en la carpeta de dependencias de terceros (vendor) de Symfony2. En esta capa estará ubicado como servidor de base de datos PostgreDQL.

Figura 6. Estructura de la carpeta Entity, donde se encuentran las clases entidades mapeadas por el ORM Doctrine.

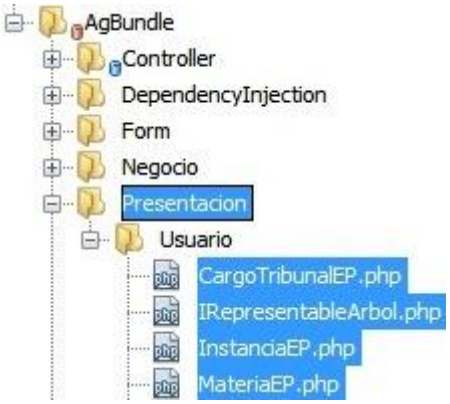

Figura 7. Estructura de la carpeta Presentación, donde se encuentran las clases entidades de presentación.

2.2 Modelo de datos

El modelo de datos es una representación abstracta de los datos de una organización y las relaciones entre ellos. En esencia un modelo de datos describe una organización. Su principal objetivo es representar los datos y ser comprensible.

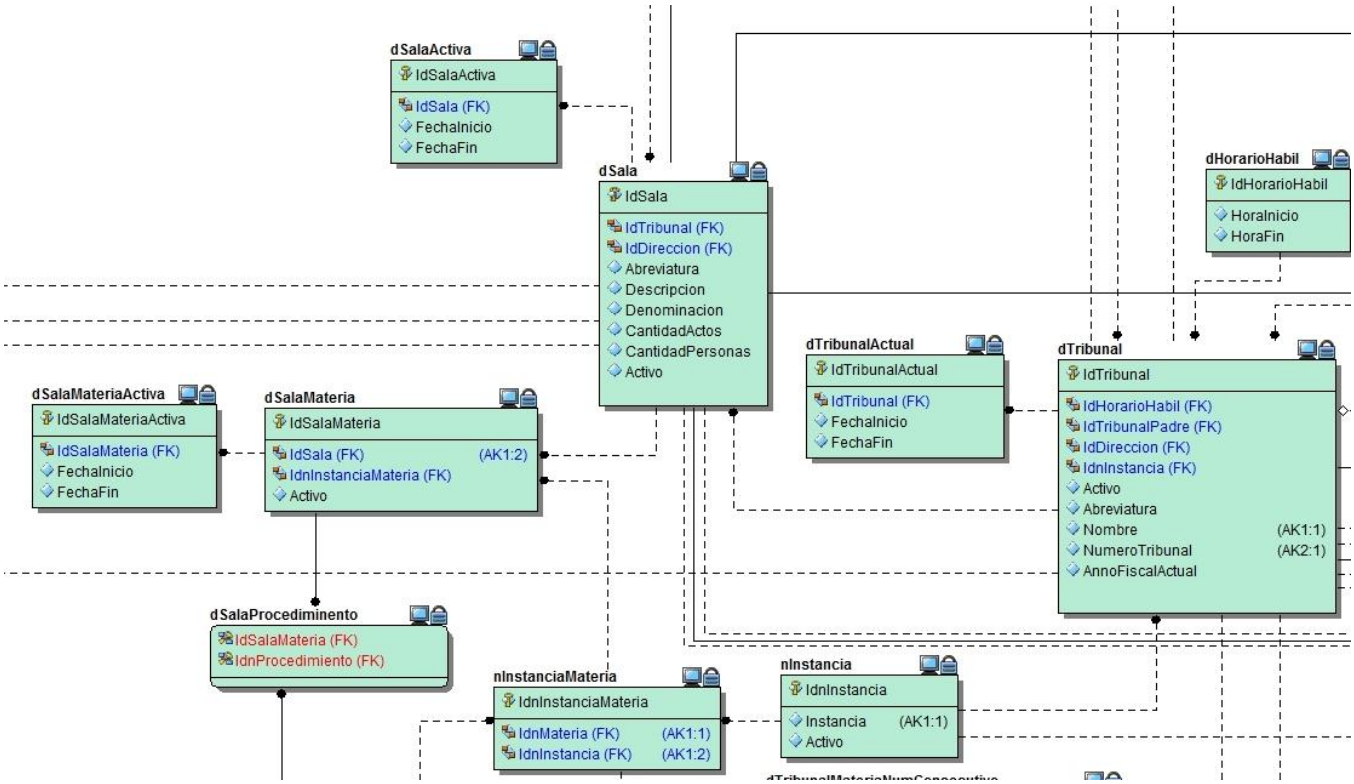

Figura 8. Fragmento del Model de datos del subsistema Administración y Gobierno. Conjunto de tablas involucradas en CU Gestionar Tribunales.

El modelo de datos del subsistema Administración y Gobierno está formado por 151 tablas, 4 de ellas son nomencladores y las restantes almacenan la información correspondiente al flujo de trabajo del módulo. La nomenclatura de las entidades anteriores es la siguiente: las tablas nomencladoras comienzan con la letra n, el resto de las tablas comienzan con la letra d y las especializaciones se nombraron así: nombre de la tabla padre + nombre de la tabla hija.

2.3 Diseño e implementación

La etapa de implementación del desarrollo de software es el proceso de convertir una especificación del sistema en un sistema ejecutable. Siempre implica los procesos de diseño y programación de software, pero, si se utiliza un enfoque evolutivo de desarrollo, también puede implicar un refinamiento de la especificación del software. (18)

En el proceso de diseño se crean los modelos de una posible implementación de la especificación de requisitos dados en el Análisis. El principal objetivo del diseño es sustentar todo el trabajo técnico de codificación, de pruebas y posteriormente, de mantenimiento. Si el diseño no se hace y nos lanzamos directamente a la codificación, veríamos seriamente afectada la calidad del producto: *Nos arriesgaríamos a construir un sistema inestable, un sistema que falle cuando se realicen pequeños cambios, un sistema que pueda ser difícil de probar; un sistema cuya calidad no pueda ser probada hasta más adelante en el proceso de Ingeniería de Software, cuando quede poco tiempo y ya se haya gastado mucho dinero.* (19)

La Implementación comienza con el resultado del diseño donde este es tomado y refinado introduciendo elementos del lenguaje de programación. Este refinamiento se realiza repetidamente hasta que se pueda derivar el código. Luego se implementan los diferentes componentes que presenta el sistema en su diseño.

2.3.1 Modelo de diseño

El modelo de diseño muestra los objetos o clases en un sistema y, donde sea apropiado, los diferentes tipos de relaciones entre estas entidades. El modelo de diseño es esencialmente el diseño mismo. Es el puente entre los requerimientos y la implementación del sistema. Esto significa que existen requerimientos en conflicto en este modelo. Tiene que ser abstracto con el fin de que el detalle innecesario no oculte las relaciones entre ellos y los requerimientos del sistema. Sin embargo, también tiene que incluir suficiente detalle para que los programadores tomen las decisiones de implementación. (18)

2.3.1.1 Diagrama de clases del diseño

El Diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y las interfaces en una aplicación. Normalmente contiene la siguiente información:

- ✓ Clases, las cuales se representan mediante una caja subdividida en tres partes: en la parte superior se muestra el nombre, en el medio los atributos y en la inferior las operaciones o métodos; y Asociaciones, que se representan mediante una línea que las une. La línea puede tener una serie de elementos gráficos que expresan características particulares de la asociación.
- ✓ Interfaces, con sus operaciones y constantes.
- ✓ Información sobre los tipos de atributos.
- ✓ Navegabilidad.
- ✓ Dependencia.

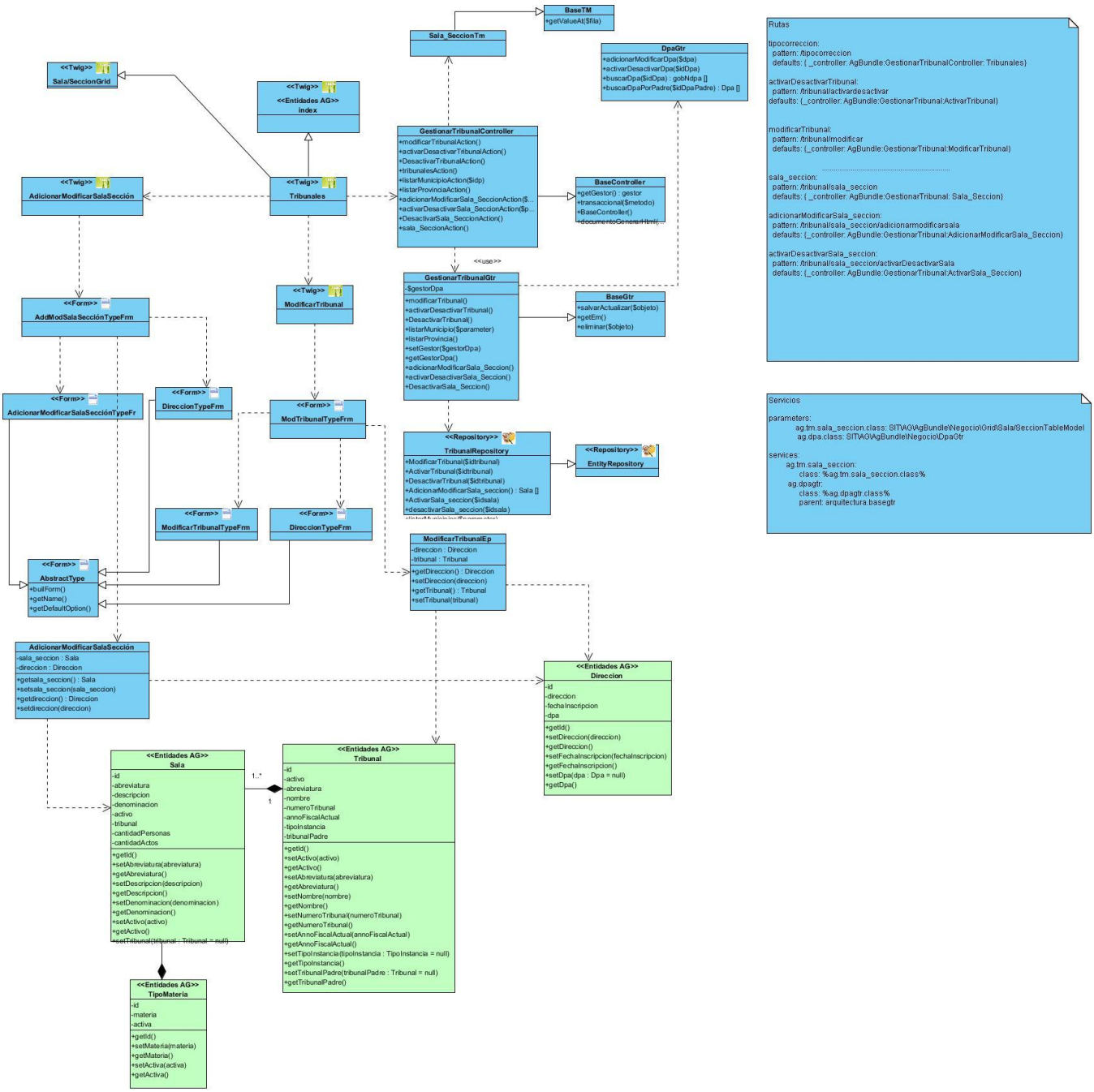

Figura 9. Ejemplo de diagrama de clases del diseño del caso de uso Gestionar tribunales.

2.3.1.2 Diagrama de interacción

Los diagramas de interacción describen el comportamiento de un sistema, para demostrar cómo los objetos interactúan dinámicamente en diferentes momentos durante la ejecución del sistema. Los objetos dentro de un sistema se comunican unos con otros, enviándose mensajes. Un mensaje es justo una operación donde un objeto llama a otro objeto. Así pues la dinámica de un sistema se refiere

a cómo los objetos dentro del sistema cambian de estado durante el ciclo de vida del mismo y también a cómo dichos objetos colaboran a través de la comunicación. Esta comunicación entre los objetos se representa mediante los diagramas de interacción, que a su vez agrupan a dos tipos de diagramas: secuencia y colaboración.

2.3.1.2.1 Diagrama de secuencia

El diagrama de secuencia muestra la interacción de un conjunto de objetos a través del tiempo. Esta descripción es importante porque puede dar detalle a los casos de uso, aclarándolos al nivel de mensajes de los objetos existentes. Es decir, proporciona la interacción entre los objetos en un escenario específico durante la ejecución del sistema. Gráficamente un diagrama de secuencia es una tabla con dos ejes: el eje horizontal muestra el conjunto de objetos y el eje vertical muestra el conjunto de mensajes.

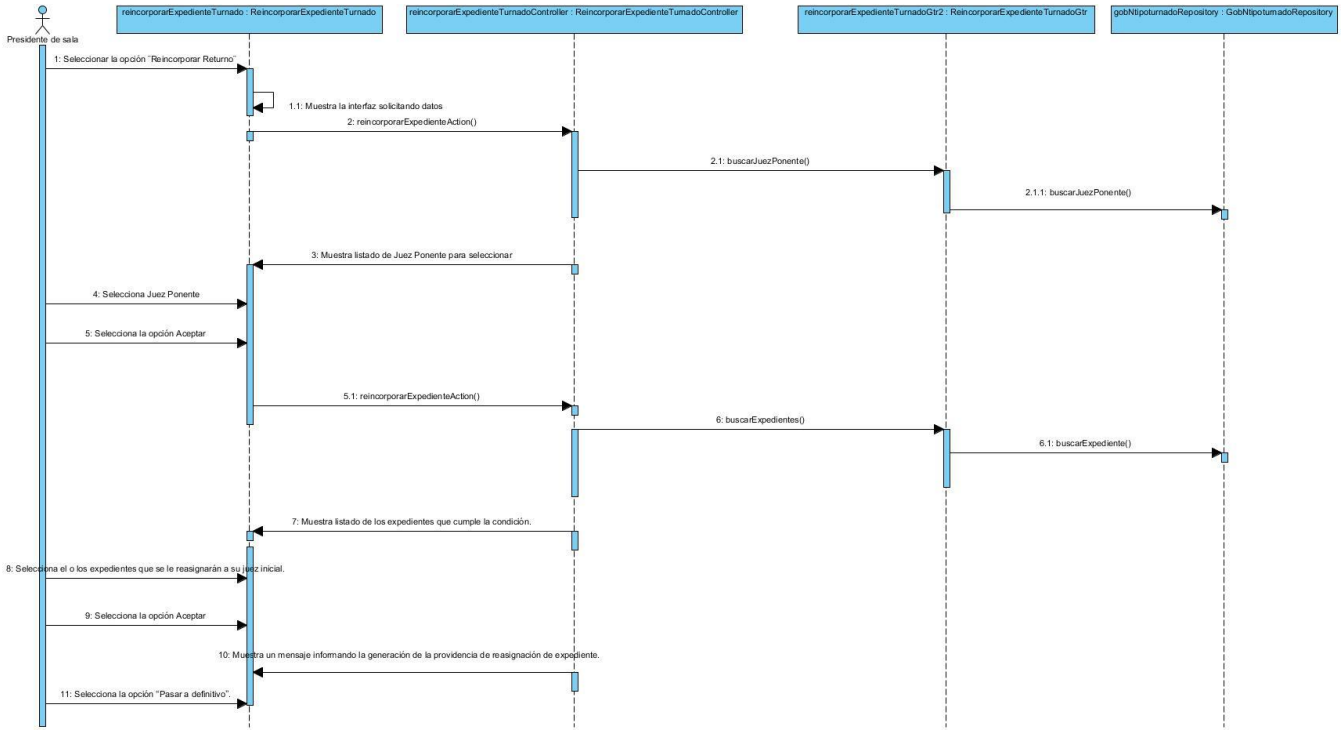

Figura 10. Ejemplo de diagrama de secuencia del caso de uso

2.3.1.3 Diagrama de despliegue

Los Diagramas de Despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como una computadora, un

dispositivo o memoria y son conectados por asociaciones de comunicación tales como enlaces de red y conexiones TCP/IP.

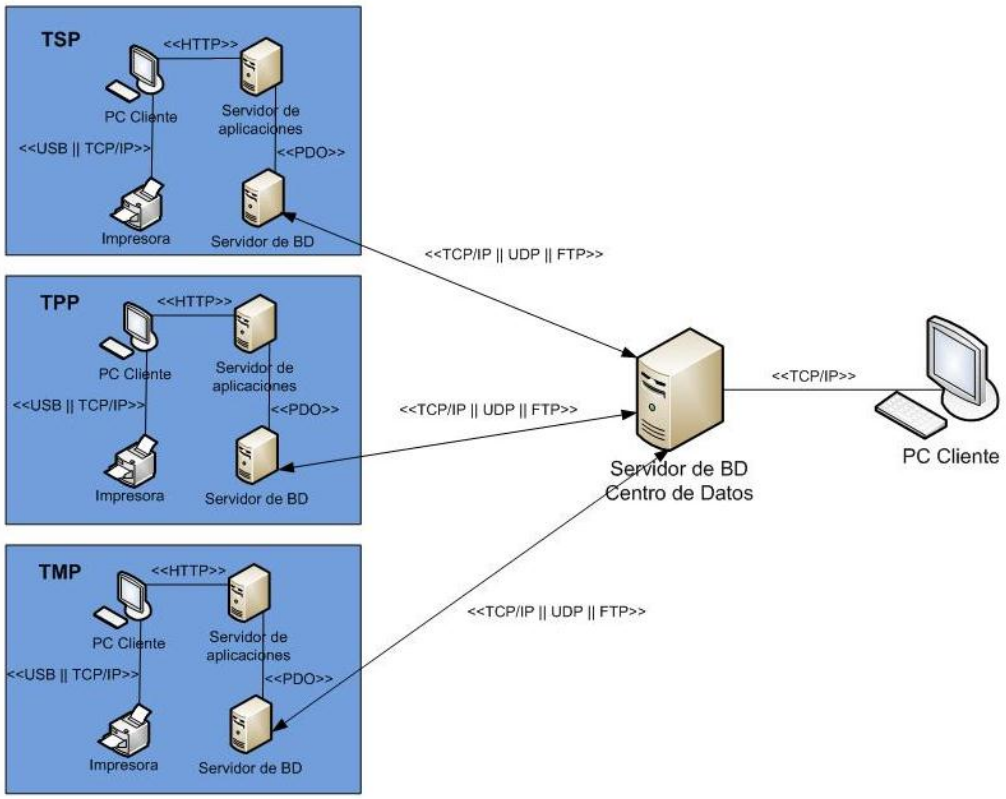

Figura 11. Diagrama de despliegue.

2.3.1.3.1 Descripción de los nodos físicos

Un nodo es un objeto físico en tiempo de ejecución que representa un recurso informático, generalmente con memoria y capacidad de procesamiento. Los nodos pertenecientes a la figura 2 se definen a continuación:

- ✓ **Nodo PC Cliente:** Computadora con el sistema operativo Linux y el navegador Mozilla Firefox, mediante el cual los usuarios podrán acceder al sistema y hacer uso del mismo.
- ✓ **Nodo Servidor de Base de Datos (Centro de Datos):** Servidor en que se encontrará la base de datos que contendrá la información de todas las instancias de los tribunales, es decir, TSP, TPP y TMP.
- ✓ **Nodo Servidor de bases de datos (Tribunal Supremo):** Servidor en que se encontrará la base de datos que contendrá toda la información del Tribunal Supremo Popular. Este servidor se sincronizará con el Centro de Datos para enviar la información del día en horario no laboral y una vez al día, específicamente de 12:00am a 6:00am.

- ✓ **Nodo Servidor de bases de datos (Tribunales Populares Provinciales):** Servidor en que se encontrará la base de datos que contendrá toda la información de los Tribunal Provinciales Populares, al igual que el tribunal supremo cada uno se sincronizará con el Centro de Datos en horario no laboral para enviar su información diaria.
- ✓ **Nodo Servidor de aplicaciones**: Servidor que contendrá todo lo referente al sistema: se incluirán los archivos necesarios para que el usuario pueda tener acceso; manejará las informaciones específicas de los registros, así como sus clases, almacenará la configuración general del sistema, las clases y librerías externas además de los plugins de instalación de la aplicación.
- ✓ **Nodo Impresora**: Dispositivo de impresión necesario para llevar a formato duro los diferentes documentos necesarios para el funcionamiento de los tribunales.

2.4 Estándares de codificación

2.4.1 Identación[15]

El contenido siempre se identará con tabs, nunca utilizando espacios en blanco.

2.4.2 Cabecera del archivo

Es importante que todos los archivos .php inicien con una cabecera específica que indique información de la versión, autor de los últimos cambios, etc. Es de cada equipo decidir si se quieren o no agregar más datos.

```
/**
* @Clase controladora acusado. "GestionarTribunalesController.php"
* @modificado: 1 de Septiembre de 2012
* @autor: Ricardo Alejandrez Nuevo
*/
```

2.4.3 Comentarios en las funciones

Todas las funciones deben tener un comentario, antes de su declaración, explicando que hacen. Ningún programador debería tener que analizar el código de una función para conocer su utilidad. Tanto el nombre como el comentario que acompañe a la función deben bastar para ello.

[15] En el contexto de informática significa mover un bloque de texto hacia la derecha insertando espacios o tabuladores para separarlo del texto adyacente, lo que en el ámbito de la imprenta se ha denominado siempre como sangrado o sangría.

```php
/**
 * Para acceder al Gestor correspondiente a este controllador
 *
 * @return GestionarTribunalesGtr
 */
private function getGestor() {
    if (!$this->container->has('ag.gestionartribunalesgtr')) {
        throw new \LogicException('Este servicio no esta registrado en la aplicacion');
    }
```

Figura 12. Ejemplo de comentario en las funciones.

2.4.4 Ubicación de los archivos

Se ubicarán los archivos según las convenciones establecidas por Symfony2 o según las especificaciones del equipo de arquitectura.

2.4.5 Denominación de archivos

Para la denominación de los archivos se seguirán las convenciones establecidas por Symfony2 o el equipo de arquitectura. **Ejemplo**:

2.4.5.1 Las clases de gestión del negocio usarán el sufijo **Gtr**:

Figura 13. Ejemplo de las entidades de gestión del negocio.

2.4.5.2 Los table model definidos para los grid usarán el sufijo TableModel:

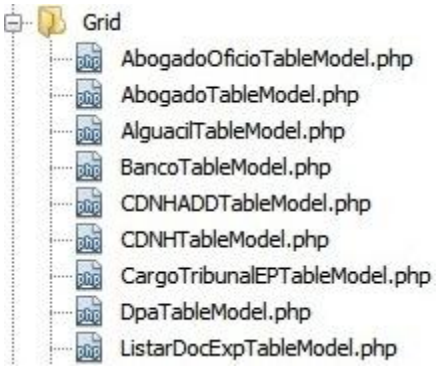

Figura 14. Ejemplo de las entidades table model.

2.4.5.3 Las entidades de presentación usarán el sufijo EP:

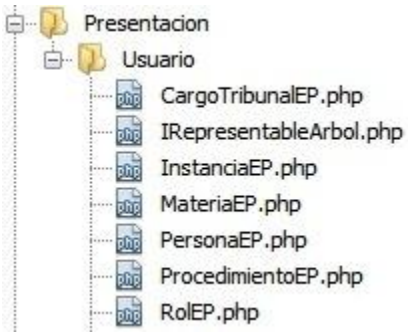

Figura 15. Ejemplo de las entidades Presentación.

2.4.5.4 En las páginas, plantillas html.twig definidas para la vista el nombre del archivo debe seguir el estándar de la denominación de las clases. Ejemplo:

Figura 16. Ejemplo de las páginas y plantillas.

2.4.5.5 Para la denominación de los servicios a crear se tendrán en cuenta los siguientes elementos:

- ✓ El nombre de los parámetros y servicios será todo en minúscula utilizando como separador el carácter punto (.)
- ✓ El nombre de las rutas será todo en minúscula utilizando como separador el caracter guión bajo (_).

```
# Rutas del Caso de Uso Gestionar Tribunales
ag_gestribu_index:
    pattern: /gestribu
    defaults: {_controller: AgBundle:GestionarTribunales:cargarTribunales }
ag_gestribu_ajax:
    pattern: /gestribu/ajax
    defaults: { _controller: AgBundle:GestionarTribunales:listarTribunales }
```

Figura 17. Ejemplo de rutas.

```
parameters:
    ag.pruebagtr.class: SIT\AG\AgBundle\Negocio\PruebaGtr
    ag.monedagtr.class: SIT\AG\AgBundle\Negocio\Gestor\MonedaGtr
services:
    ag.leyesgtr:
        class: %ag.leyesgtr.class%
        parent: arquitectura.basegtr
```

Figura 18. Ejemplo de parámetros y servicios.

2.4.6 Clases

Las clases serán colocadas en un archivo .php aparte, donde sólo se colocará el código de la clase. El nombre del archivo será el mismo del de la clase. Las clases siguen las mismas reglas de las funciones, por tanto, debe colocarse un comentario antes de la declaración de la clase explicando su utilidad. Los nombres de las clases deben de iniciar con letra mayúscula. Si un nombre de la clase se comprende de más de una palabra, la primera letra de cada nueva palabra debe capitalizarse. No se permiten las letras capitalizadas sucesivas; por ejemplo, una clase "SymfonyPDF" no se permite, mientras " SymfonyPdf" es aceptable.

Siempre utilizar las etiquetas <?php ?> para abrir un bloque de código. No utilizar el método de etiquetas cortas.

```php
namespace SIT\AG\AgBundle\Controller;

use SIT\AG\AgBundle\Form\AG\TribunalType;
use Base\ArquitecturaBundle\Controller\BaseController;
use Base\ComunBundle\Negocio\Componentes\Grid\RutasGrid;
use Base\ComunBundle\Entity\AG\Tribunal;
use Symfony\Component\HttpFoundation\Response;

/**
 * Clase controladora que contiene las funcionalidades referentes
 * a la gestión de los tribunales.
 * @author Ricardo Alejandrez <raalejandrez@estudiantes.uci.cu>
 */
class GestionarTribunalesController extends BaseController {
```

Figura 19. Ejemplo de clase controladora.

2.4.7 Estilo y reglas de escritura de código PHP

2.4.7.1 Nombre de las variables

Los nombres deben ser descriptivos y concisos. No usar ni grandes frases ni pequeñas abreviaciones para las variables. Siempre es mejor saber qué hace una variable con sólo conocer su nombre. Esto aplica para los nombres de variables, funciones, argumentos de funciones y clases. Los nombres de las variables y de las funciones pueden inician con letra minúscula, pero si estas tienen más de una palabra, cada nueva palabra debe iniciar con letra mayúscula. Las constantes deben de escribirse siempre en mayúsculas y tanto estas como las variables globales deben de tener como prefijo el nombre de la clase a la que pertenecen.

2.4.7.2 Las definiciones de la función

Los nombres de la función pueden contener sólo caracteres alfanuméricos. Los nombres de la función siempre deben empezar en letras minúsculas. Cuando un nombre de la función consiste de más de una palabra, la primera letra de cada nueva palabra debe capitalizarse.

```php
public function cargarSalasAction($id) {
    $idtribunal = $id;
    $modelo = $this->get('ag.tm.sala');
    $ruta = new RutasGrid();
    $ruta->setRutaDatosAjax('ag_gridSalas_ajax');
    $ruta->setRutaAdicionar('ag_gridSalasAdicionar');
    $ruta->setRutaActualizar('ag_gridSalasAdicionar');
    $ruta->setRutaEliminar('ag_desactivarSala');
```

Figura 20. Ejemplo de función con más de una palabra en el nombre

2.4.7.3 Llamadas a funciones

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

```
<?php
        $variable = $this->obtenerSala($idTribunal, $idMateria);
?>
```

2.4.7.4 Siempre incluir llaves

En todo momento a la hora de codificar un bloque de instrucciones, este debe ir encerrado entre llaves, aun cuando conste de una sola línea.

Si se iba a hacer esto:

```
        if($cosa) function();
```

Mejor se hace esto:

```
        if ($cosa) {
                function();
        }
```

2.4.7.5 Colocar correctamente las llaves

Las llaves de apertura irán al final de la sentencia que delimitan y la de cierre alineadas con el inicio de la sentencia en una nueva línea.

```
        if ($algo) {
                for(iteración) {
                        //código
                }
        }
        while (condición) {
                function();
        }
```

2.4.7.6 Poner espacios entre signos

Si se tiene un signo y operador binario, se colocan espacios a ambos lados. Si se tiene un signo unario, se colocan espacios a uno de sus lados. En términos más simples, se programa como si se escribiera bien en español. Este elemento es algo muy sencillo que ayuda a la legibilidad del código.

Incorrecto:

```
$a=0;
for($i=5;$i<=$j;$i++)
```

Correcto:

```
$a = 0;
for ($i = 5; $i <= $j; $i++)
```

Esto se realiza de manera automática en el caso del IDE “NetBeans”, al presionar las teclas Alt+Shift+f se autoformatea el contenido del fichero activo. Haciendo cumplir estas 2 últimas pautas.

2.4.7.7 Precedencia de operadores

Lo mejor es siempre usar paréntesis para estar seguro de la precedencia de los operadores. Básicamente, la idea es no codificar operaciones complejas y estar seguros que nuestros compañeros de equipo con menos “habilidad” comprendan todo sin problemas:

Incorrecto:

```
$bool = ($i < 7 && $j > 8 || $k == 4);
```

Correcto:

```
$bool = (($i < 7) && (($j < 8) || ($k == 4)));
```

Mejor:

```
$bool = ($i < 7 && ($j < 8 || $k == 4));
```

2.4.7.8 No utilizar variables sin inicializar

Si no se tiene control sobre el valor de una variable, se debe verificar que esté inicializada. Esto lo permite PHP de la siguiente manera:

Incorrecto:

```
if ($cliente == 5)
```

Correcto:

```
if (isset($cliente) && $cliente == 5)
```

Es importante aclarar que sólo se debe usar está opción cuando no se tenga el control o no se esté seguro del valor que pueda tener la variable (Como en variables que llegan por parámetro o por GET, etc.).

2.4.7.9 Instrucción “switch”

En el momento de utilizarla se debe seguir el siguiente estilo:

```
switch ($modo) {
```

```
        case 'modo1':
            // Instrucción 1
            break;
        case 'modo2':
            // Instrucción 2
            break;
        default:
            // Código a ejecutar si todo falla
            break;
}
```

2.5 Patrones de diseño

Un patrón de diseño es una solución que se repite a un problema que ocurre varias veces en el diseño de software. Esta solución no es un diseño terminado que puede traducirse directamente a código, sino más bien una descripción sobre cómo resolver el problema, la cual puede ser utilizada en diversas situaciones. Los patrones de diseño reflejan todo el rediseño y remodificación que los desarrolladores han ido haciendo a medida que intentaban conseguir mayor reutilización y flexibilidad en su software. Los patrones documentan y explican problemas de diseño, y luego discuten una buena solución a dicho problema. Con el tiempo, los patrones comienzan a incorporarse al conocimiento y experiencia colectiva de la industria del software, lo que demuestra que el origen de los mismos radica en la práctica misma más que en la teoría.

2.5.1 Patrones GOF[16]

En el año 1944 aparece el libro "*Design Patterns: Elements of Reusable Object Oriented Software*" escrito por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides, ellos recopilaron y documentaron 23 patrones de diseño aplicados usualmente por expertos diseñadores de software orientado a objetos.

Los patrones de diseño el grupo de GOF clasifican en 3 grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento.

- ✓ **Creacionales**: tratan con las formas de crear instancias de objetos. El objetivo de estos patrones es de abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados. Dentro de este grupo se encuentran:
    - Object Pool (Conjunto de Objetos)

[16] Del inglés Gang Of Four, en español Pandilla de los Cuatro

- Abstract Factory (Fábrica abstracta)
- Builder (Constructor virtual)
- Factory Method (Método de fabricación)
- Prototype (Prototipo)
- Singleton (Instancia única)

✓ **Estructurales**: Los patrones estructurales describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos. Dentro de este grupo se encuentran:

- Adapter (Adaptador)
- Bridge (Puente)
- Composite (Objeto compuesto)
- Decorator (Envoltorio)
- Facade (Fachada)
- Flyweight (Peso ligero)
- Proxy

✓ **Comportamiento**: Los patrones de comportamiento ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos. Dentro de este grupo se encuentran:

- Chain of Responsibility (Cadena de responsabilidad)
- Command (Orden)
- Interpreter (Intérprete)
- Iterator (Iterador)
- Mediator (Mediador)
- Memento (Recuerdo)
- Observer (Observador)
- State (Estado)
- Strategy (Estrategia)
- Template Method (Método plantilla)
- Visitor (Visitante)

2.5.1.1 Factory Method (Método de fabricación)

El patrón de diseño Factory Method consiste en definir una interfaz para crear objetos de tipo genérico permitiendo a las subclases decidir qué tipo de objetos concreto crear. En esencia permite que una clase delegue en sus subclases la creación de objetos. Todos los objetos instanciados normalmente provienen de una misma clase padre, es su comportamiento el que los diferencia.

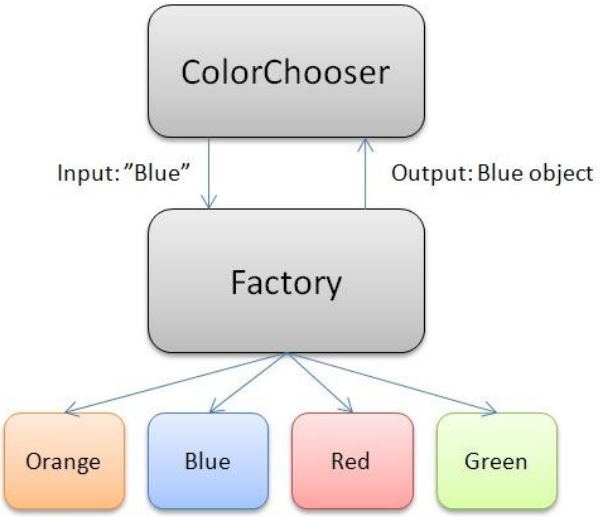

Figura 21. Ejemplo del funcionamiento del patrón de diseño Factory Method.

En este ejemplo, se le pasa un parámetro al método getInstance() de la clase Factory. Dependiendo del parámetro pasado se devuelve un objeto. En este caso, un objeto de la clase Blue es retornado.
En el desarrollo del SITPC este patrón es utilizado para crear y retornar instancias de los repositorios de las entidades y funciona de la siguiente manera: se realiza una petición al contenedor de servicios en la que se le pasa por parámetros el nombre de un servicio.

```
/**
 * Metodo que lista todos los usuarios
 * @return type
 */
public function listarUsuario($datos) {
    $usuario = $this->get('comun.servicios.Usuario')->filtro(
```

Figura 22. Ejemplo de petición al contenedor de servicios.

En este servicio está definida la clase repository, el factory_method y la entidad a la que se refiere el servicio.

```
#--------------U-------------
    comun.servicios.Usuario:
        class: Base\SeguridadBundle\Repository\UsuarioRepository
        factory_service: doctrine.orm.default_entity_manager
        factory_method: getRepository
        arguments: [Base\SeguridadBundle\Entity\Usuario]
```

Figura 23. Ejemplo de servicio de la clase service.yml

Internamente se hace una llamada a la clase EntityManager.php, pasándole por parámetro la entidad identificada en el servicio. La clase EntityManager.php es la encargada de implementar el patrón Factory Method, lo que se hace específicamente en el método getRepository(entityName).

```
/**
 * Gets the repository for an entity class.
 *
 * @param string $entityName The name of the entity.
 * @return EntityRepository The repository class.
 */
public function getRepository($entityName)
{
    $entityName = ltrim($entityName, '\\');

    if (isset($this->repositories[$entityName])) {
        return $this->repositories[$entityName];
    }

    $metadata = $this->getClassMetadata($entityName);
    $repositoryClassName = $metadata->customRepositoryClassName;

    if ($repositoryClassName === null) {
        $repositoryClassName = $this->config->getDefaultRepositoryClassName();
    }

    $repository = new $repositoryClassName($this, $metadata);

    $this->repositories[$entityName] = $repository;

    return $repository;
}
```

Figura 24. Método getRepository() de la clase EntityManager que implementa el patrón Factory.

2.5.1.2 Decorator (Envoltorio)

El principal objetivo de este patrón es añadir responsabilidades a objetos concretos de manera dinámica y transparente sin afectar a otros objetos. Este patrón brinda más flexibilidad que la herencia estática y evita que las clases más altas en la jerarquía estén demasiado cargadas de funcionalidad y sean complejas.

En la implementación de SITPC este patrón se evidencia en el uso de una plantilla global para las vistas que decorará las demás páginas de la aplicación.

Figura 25. Página decorada con la plantilla global.

2.5.2 Patrones GRASP

Los patrones GRASP[17] describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. El nombre se eligió para indicar la importancia de captar (grasping) estos principios, si se quiere diseñar eficazmente el software orientado a objetos. Dentro de los diferentes patrones GRASP los utilizados en la construcción de la solución fueron Creador, Experto, Alta cohesión, Bajo acoplamiento, y Controlador.

2.5.2.1 Creador

El Creador dirige la asignación de responsabilidades relacionadas con la creación de instancias y su principal objetivo es que el creador de la instancia sea capaz de conectar con la instancia producida en cualquier evento. Este patrón da solución a la siguiente interrogante: ¿Quién es responsable de crear una nueva instancia de alguna clase? Para darle solución a esta pregunta se mostraran diferentes casos de cuando es correcto asignarle a una clase B la responsabilidad de crear una instancia de la clase A:

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

En la implementación del SITPC este patrón se evidencia en las clases Gtr en el momento de crear nuevas instancias de las clases Entidades. Las Gtr al recibir de toda la información referente a las

[17] Del inglés general responsibility assignment software patterns, en español patrones generales de software para asignar responsabilidades

entidades son las responsables de crear las mismas. Otro ejemplo son las clases Controller, que son las responsables de la creación de los formularios que se mostrarán en las vistas.

```
//Es modificar se carga la entidada a modificar y se activa modificar
else {
    $entidad = $this->getGestor()->buscarSalaPorId($id);
    $provincia = $entidad->getDireccion()->getProvincia()->getId();
    $modificar = true;
    $ids=array();
    $salasmateria=$entidad->getSalaMateria();

    for ($i = 0; $i < count($salasmateria); $i++) {
        $ids[$i]=$salasmateria[$i]->getTipoInstanciaMateria()->getTipoMateria()->getId(
    }
      var_dump($ids);
      exit;
}
$form = $this->createForm(new SalaType(), $entidad);
```

Figura 26. Uso del patrón creador en la clase Controller GestionarSala.

2.5.2.2 Experto

Este patrón resuelve la interrogante: ¿A qué clase es más conveniente asignarle una responsabilidad en un diseño orientado a objetos? El patrón Experto responde que la clase que debe llevar la responsabilidad es la experta en información, que no es más que aquella clase que contiene toda la información para cumplir con la responsabilidad. Si se le otorga a una sola clase (Experta) la responsabilidad, existe una dependencia nula con otras clases por lo que se garantiza un Bajo acoplamiento y un alto grado de encapsulamiento.

Al igual que el patrón Creador en el desarrollo del SITPC este patrón se observa en la creación de las vistas para mostrárselas al usuario, como las clases Controller, que son las expertas en información, son las encargadas de crear los formularios, y a través de los mismos se creará la vista que se le mostrará al usuario.

```
  $form = $this->createForm(new TribunalType(), $tribunal);

if ('POST' == $request->getMethod()) {

 $form->bindRequest($request);
 $tribunal=$form->getData();
if(!$tribunal->getDireccion()->getProvincia()){
     $provincia=null;
}
 if ($form->isValid()) {
     $this->getGestor()->adicionarModificarTribunal($tribunal);
     return new Response(1);
 } else {
     return $this->render('AgBundle:Paginas\gestionarTribunales:modificarTribunales.html.twig',
             array('formulario' => $form->createView(), 'bandera' => $bandera,
                 'id' => $idtribunal,'provincia'=>$provincia));
 }
```

Figura 27. Uso del patrón Experto en la clase GestionarTribunalesController.

2.5.2.3 Bajo acoplamiento

Cuando una clase depende de muchas otras decimos que existe un alto acoplamiento, esto trae como consecuencia que al realizar un cambio en las clases afines se realizarán cambios locales y que estas clases cuando estén aisladas serán más difíciles de entender, por lo que la mejor práctica es asignar responsabilidades manteniendo un bajo acoplamiento.

Este patrón se observa en el uso del patrón arquitectónico MVC del SITPC como las clases de la Vistas no tienen relación con las clases gestoras y las entidades, que representan el modelo, y estas últimas no presentan ninguna relación con los controladores.

2.5.2.4 Alta cohesión

La cohesión es la medida de cuan relacionadas y enfocadas están las responsabilidades de un clase. Una clase con una baja cohesión son difíciles de reutilizar, son delicadas debido a los constantes cambios y son difíciles de comprender por lo que la mejor opción es mantener la complejidad dentro de límites manejables a través de la alta cohesión.

En la implementación del SITPC este patrón se ve en la interrelación que existe entre las clases controladora, las clases gestora y los casos de uso. Cada caso de uso presenta una clase controladora y una gestora, la primera encargada de manejar la lógica de presentación y el flujo de los datos provenientes de la vista y la segunda encargada de manejar la lógica del negocio del caso de uso.

2.5.2.5 Controlador

El objetivo de este patrón es asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades como validaciones, seguridad, entre

otras. El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión.

Un error muy común es asignarle demasiada responsabilidad y alto nivel de acoplamiento con el resto de los componentes del sistema. Este patrón garantiza un mayor potencial de los componentes reutilizables, garantiza que los eventos del sistema sean manejados por la capa de los objetos del dominio y no por la de la interfaz. Este patrón se evidencia en la solución a través de las clases controladoras que se encuentran en la carpeta Controller perteneciente a cada Bundle de la solución, y en la estructura de Symfony2 lo podemos encontrar en el uso del controlador frontal que se encuentra en la carpeta web de cada proyecto de Symfony2.

2.6 Inyección de dependencia

La inyección de dependencias consiste en pasar a cada componente todo lo que necesita a través de sus constructores, métodos o campos. Este patrón a su vez se deriva de uno más genérico, el patrón de inversión del control. En symfony2 este patrón se evidencia en el trabajo con los servicios y el contenedor de servicios. Un servicio es un objeto PHP que se crea cuando se es necesario utilizar la función a la que hace referencia dicho servicio y un contenedor de servicios (o contenedor de inyección de dependencias) es un objeto PHP que gestiona la creación de las instancias de los servicios. Sin el contenedor de servicios fuera necesario crear cada servicio manualmente. El contenedor de servicio se encuentra accesible desde cualquier controlador de symfony2 y a través de estos controladores es posible acceder a cualquier servicio mediante el método get() o a través del objeto container.

```
class GestionarSalaController extends BaseController {

    private function getGestor() {
        if (!$this->container->has('ag.salagtr')) {
            throw new \LogicException('Este servicio no esta registrado en la aplicacion');
        }
        return $this->container->get('ag.salagtr');
    }

    public function cargarSalasAction($id) {
        $idtribunal = $id;
        $modelo = $this->get('ag.tm.sala');
```

Figura 28. Llamada a un servicio a través del método get() y el objeto container.

Las ventajas de utilizar un contenedor de servicios es que un servicio solo es creado cuando se necesita y de igual forma es eliminado una vez que se deja de usar, esto trae un ahorro de memoria y un aumento de velocidad de procesamiento

2.7 Conclusiones parciales

En este capítulo se abordó lo referente a los flujos Diseño e Implementación. Se describió la arquitectura y los patrones que son utilizados en el desarrollo del sistema y se entendió de una mejor forma el sistema realizado. Con la conclusión de este capítulo se obtuvo la implementación del módulo Administración y Gobierno del SIT. Se obtuvo el Diagrama de despliegue que muestra su organización.

Capítulo 3: Validación de los resultados

Introducción

En este capítulo se evalúa el grado de calidad y que tan fiable son los resultados obtenidos con el desarrollo de este trabajo, a través del uso de un conjunto de métricas utilizadas internacionalmente para este fin, específicamente las asociadas a las de la medición de la calidad del diseño e implementación de software como son los instrumentos de evaluación Tamaño Operacional de Clase, Relaciones entre Clases, Carencia de cohesión en los métodos y Árbol de profundidad de herencia. Se realizarán además pruebas de Caja Negra con el objetivo de encontrar y corregir la mayor cantidad de errores del sistema en el menor tiempo posible.

3.1 Métricas de software.

Aquella aplicación continua de técnicas basadas en la medida de los procesos de desarrollo del software, para producir una información de gestión significativa al mismo tiempo que se mejoran aquellos procesos y sus productos, se denominan métrica de software. “Un Método y una escala cuantitativos que pueden ser usados para determinar el valor que toma cierta característica en un producto de software concreto” (20). Las métricas empleadas están diseñadas para evaluar los siguientes atributos de calidad:

- ✓ Responsabilidad, consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- ✓ Complejidad de implementación, grado de dificultad que tiene implementado un diseño de clases.
- ✓ Reutilización, grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- ✓ Acoplamiento, grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.
- ✓ Complejidad del mantenimiento, grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software.
- ✓ Cantidad de pruebas, número o grado de esfuerzo para realizar las pruebas de calidad del producto (componente, módulo, clase, conjunto de clases, etc.) diseñado. (5)

3.1.1 Métricas aplicadas al diseño del software.

3.1.1.1 Tamaño Operacional de Clase (TOC)

Consiste en medir el tamaño general de una clase tomando el valor de la cantidad de operaciones. Si el resultado obtenido indica valores grandes, significa que la clase posee un alto grado de responsabilidad, debido a esto se reducirá la reutilización, se hará mucho más difícil la implementación y la realización de pruebas de dicha clase. Por tanto mientras menor sea el valor para el TOC se hará mucho más fácil la reutilización de dicha clase dentro del sistema.

Atributos	TOC
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Tabla 1. Atributos de calidad y modo en que afectan al TOC.

3.1.1.2 Relaciones entre Clases (RC)

Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

Atributos	RC
Acoplamiento	Un aumento del RC implica un aumento del acoplamiento de la clase
Complejidad de mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 2. Atributos de calidad y modo en que afectan al instrumento RC.

3.1.1.3 Carencia de cohesión en los métodos (CCM)

El CCM no es más que la cantidad de métodos que acceden a un mismo atributo dentro de la clase. Cuando el CCM es alto los métodos deben acoplarse a otro por medio de los atributos lo que incrementa la complejidad del diseño de clases. En general, los valores de CCM altos implican que la clase debe ser rediseñada descomponiéndola en dos o más clases distintas.

3.1.1.4 Árbol de profundidad de herencia (APH)

El APH se define como la longitud máxima de los nodos a la raíz del árbol. Mientras mayor sea el APH las clases de los más bajos niveles heredarán muchos métodos. Esto dificulta la tarea de predecir el comportamiento de una clase además de introducir una dificultad mayor en el diseño.

3.1.2 Resultados de la aplicación de las métricas de diseño

3.1.2.1 Resultado de la aplicación de la métrica Tamaño Operacional de Clase (TOC)

La métrica TOC fue aplicada a una muestra de 30 clases escogidas aleatoriamente del módulo Administración y Gobierno, la Figura 29 muestra el porciento de la cantidad de procedimientos presentes en las clases agrupados en intervalos definidos, resultando 15 clases entre 1-5 procedimientos, 10 clases entre 6-10, 3 clases entre 11-15 y 2 clases entre 21-25 procedimientos.

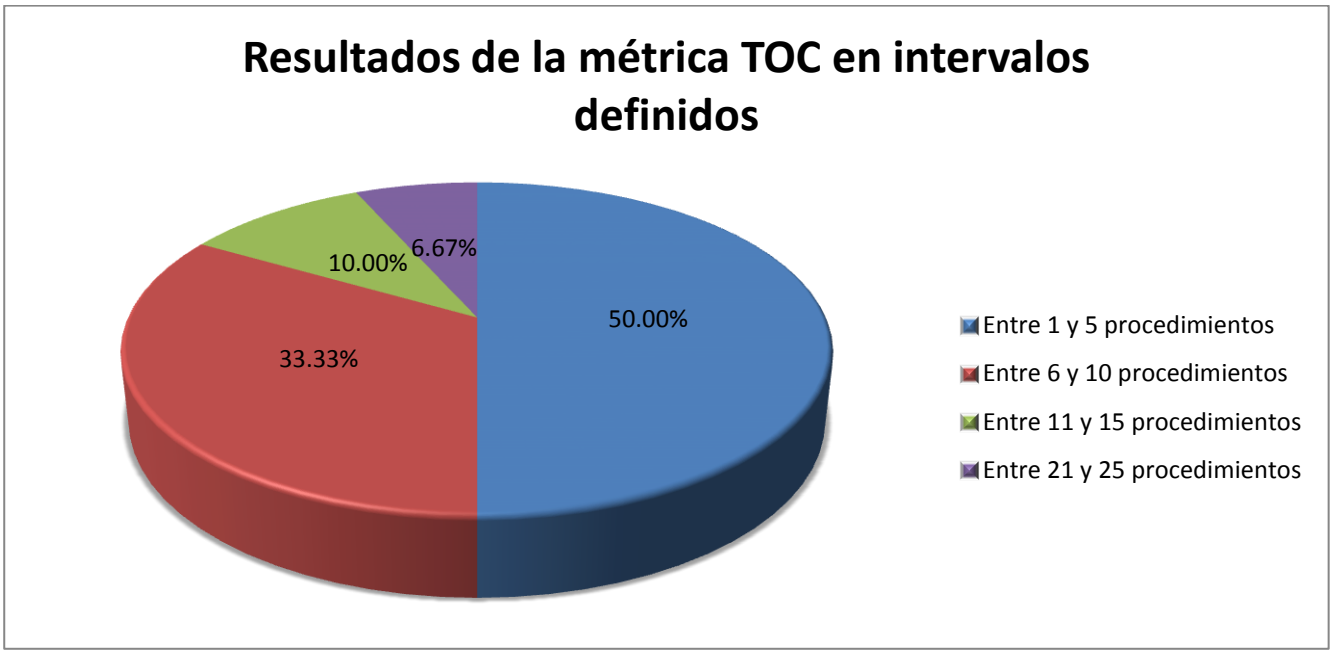

Figura 29. Representación en porciento de los resultados obtenidos, agrupados en intervalos definidos tras aplicar la métrica TOC.

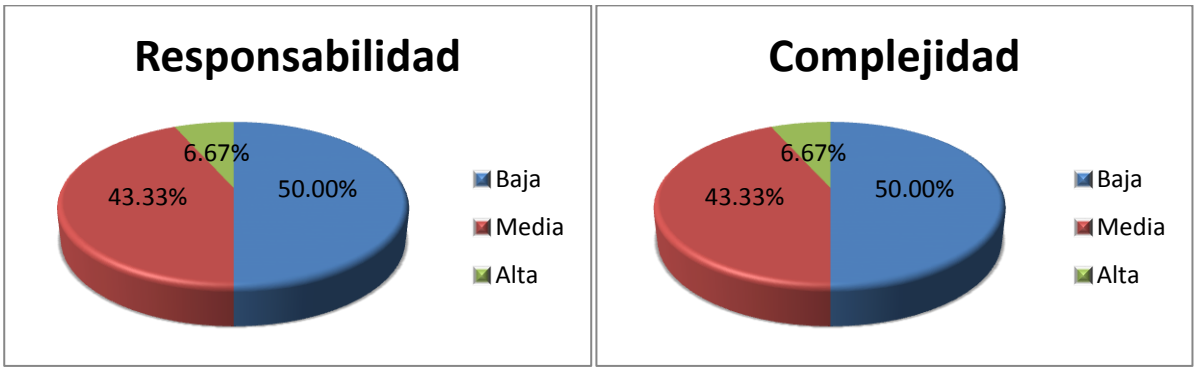

Figura 30. Representación de los resultados de la evaluación de la métrica TOC en los atributos: responsabilidad y complejidad.

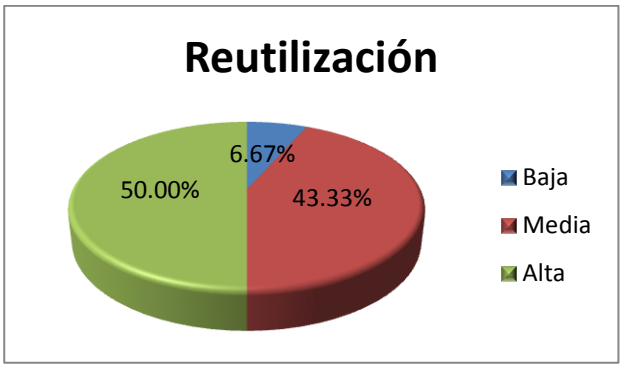

Figura 31. Representación de los resultados de la evaluación de la métrica TOC en el atributo reutilización.

Después de hacer un análisis de los resultados obtenidos se concluye que la mayoría de las clases incluidas poseen de 1 a 5 procedimientos en su composición representando el 50% de la muestra. Debido a esto, el resultado influye positivamente en los demás atributos de calidad puesto que ayuda a que el 50% de las clases posea una responsabilidad baja, es decir que no estén demasiado sobrecargadas, además puede observarse que el 50% del total de clases analizadas tiene una complejidad también baja y que se pueda realizar una explotación alta del atributo reutilización, estos resultados se ilustran en las figuras 30 y 31.

3.1.2.2 Resultado de la aplicación de la métrica Relación entre Clases (RC)

La métrica de diseño RC fue aplicado a una muestra de 11 clases del módulo Administración y Gobierno seleccionadas aleatoriamente, la Figura 32 ilustra la cantidad de dependencias y el porciento que representan de la muestra seleccionada. Las figuras 33 y 34 muestran los resultados que arrojó la aplicación de la métrica en los atributos de calidad acoplamiento, complejidad de mantenimiento, reutilización y cantidad de pruebas.

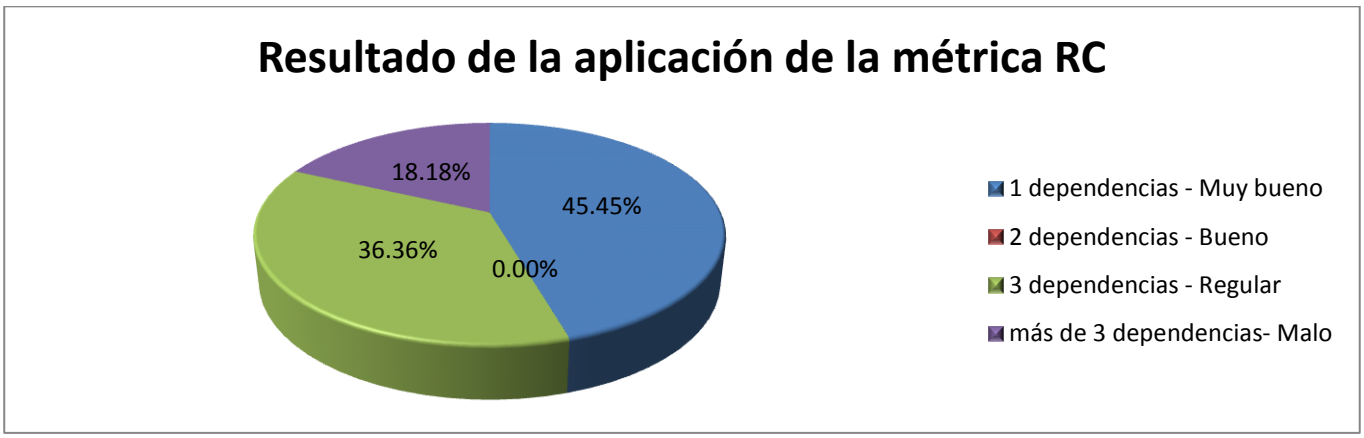

Figura 32. Representación en porciento de la aplicación de la métrica RC en intervalos definidos.

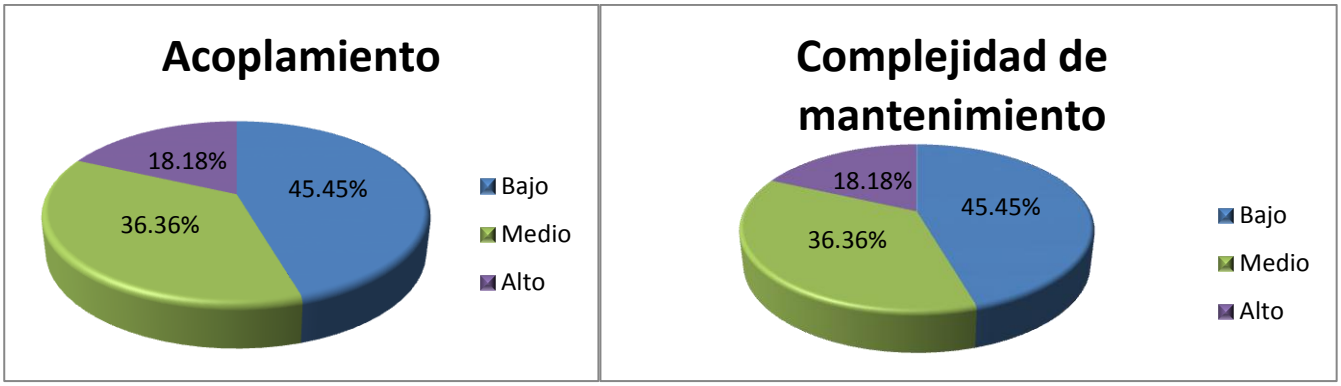

Figura 33. Representación de los resultados de la evaluación de la métrica RC en los atributos: acoplamiento y complejidad de mantenimiento.

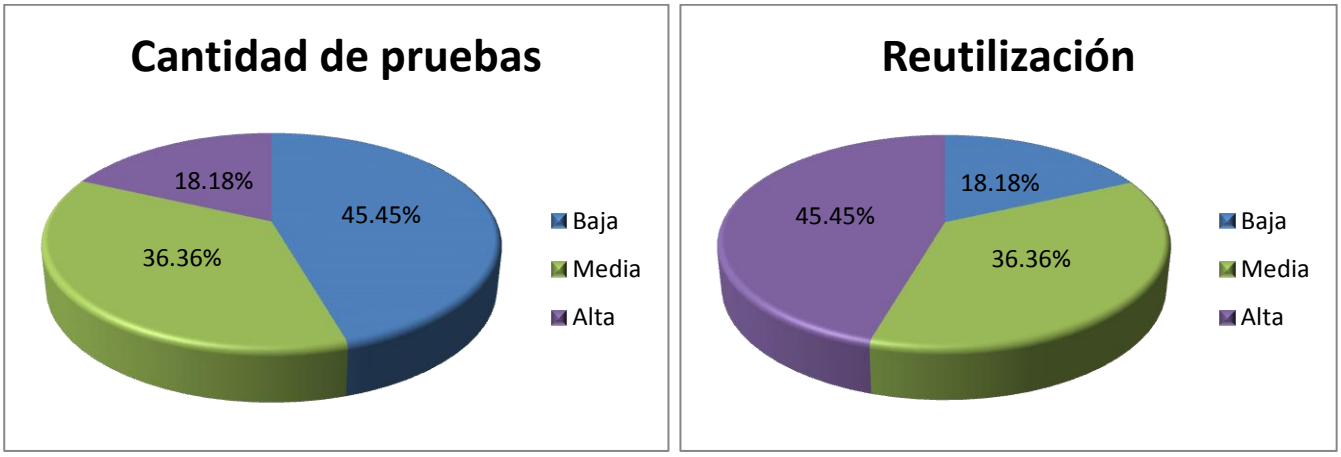

Figura 34. Representación de los resultados de la evaluación de la métrica RC en los atributos: reutilización y cantidad de pruebas.

Después de hacer un análisis de los resultados obtenidos a través de la métrica de software RC aplicada se obtiene que el 46% de las clases incluidas poseen un bajo acoplamiento lo cual ayuda significativamente en el sentido de que al ser afectada una clase este cambio repercutirá de manera mínima en las demás. El porciento de reutilización de clases (46%) es alto lo cual fomenta y potencia el uso de las mismas en otras clases, objetos, etc., en caso de ser necesario, además la complejidad de mantenimiento es un 46% baja facilitando que no resulte nada difícil el mantenimiento de las mismas (por ejemplo la optimización de métodos) y por último el atributo relacionado con la cantidad de pruebas es 46% bajo lo cual señala que a las clases se les puede realizar un bajo número de pruebas de poca complejidad y de bajos costes.

3.1.2.3 Resultado de la aplicación de la métrica Carencia de Cohesión en los Métodos (CCM)

Esta métrica fue aplicada a una de las clases más significativas del negocio:

Clase “Tribunal”	
Atributos	**Identificador**
id	a
activo	b
abreviatura	c
nombre	d
numeroTribunal	e
annoFiscalActual	f
direccion	g
regla	h
tipoInstancia	i
horarioHabil	j
tribunalPadre	k
tribunalActual	l
sala	m
tramite	n

Tabla 3 Atributos de la clase Tribunal.

Clase “Tribunal”	
Métodos	**Atributos**
getId	a
getActivo	b
setActivo	b
getAbreviatura	c
setAbreviatura	c
getNombre	d
setNombre	d
getNumeroTribunal	e

setNumeroTribunal	e
getAnnoFiscalActual	f
setAnnoFiscalActual	f
getDireccion	g
setDireccion	g
addRegla	h
removeRegla	h
getRegla	h
getTipoInstancia	i
setTipoInstancia	i
getHorarioHabil	j
setHorarioHabil	j
getTribunalPadre	k
setTribunalPadre	k
addTribunalActual	l
removeTribunalActual	l
getTribunalActual	l
addSala	m
removeSala	m
getSala	m
addTramite	n
removeTramite	n
getTramite	n

Tabla 4. Métodos de la clase Tribunal.

Este procedimiento se le aplicó a 9 clases escogidas aleatoriamente del módulo Administración y Gobierno:

Clases	Nivel CCM
Tribunal	3
Sala	3

AbogadoOficio	2
GestionarTribunalesController	0
GestionarSalaController	0
AbogadoOficioController	0
GestionarTribunalesGtr	0
SalaGtr	0
AbogadoOficioGtr	0

Tabla 5. Niveles CCM de las clases de la muestra después de aplicada la métrica.

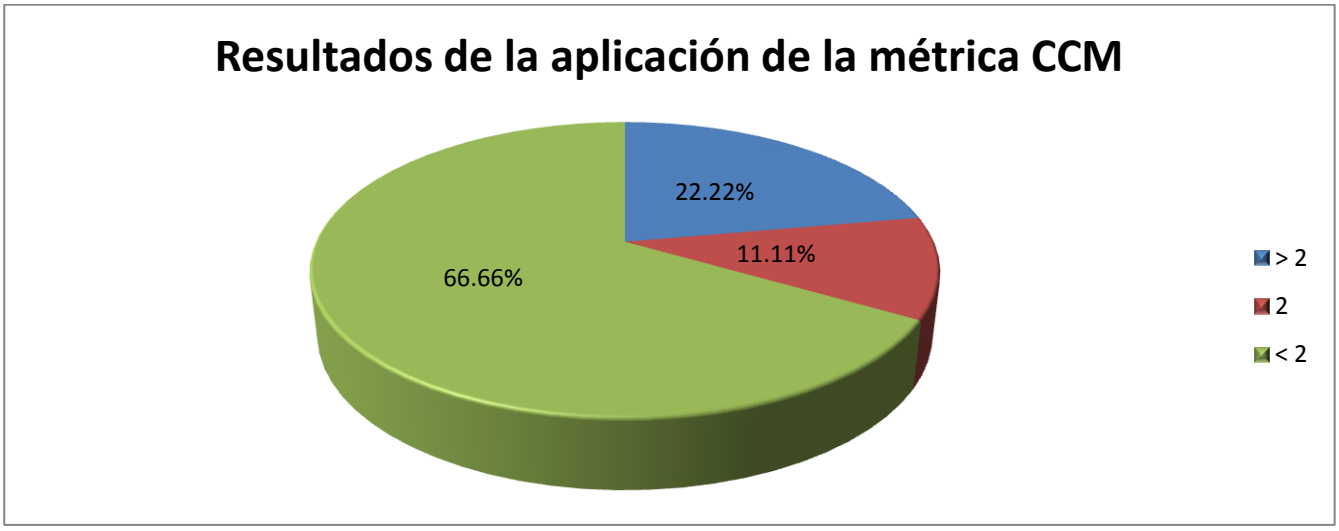

Figura 35. Representación en porciento de la aplicación de la métrica CCM.

Analizando los resultados obtenidos se concluye que el 67% de la muestra presenta un nivel 0 de CCM. Este resultado representa un nivel óptimo de acuerdo a lo planteado por la métrica. Esto disminuye la complejidad del diseño de clases y por tanto favorece a una alta cohesión.

3.1.2.4 Resultado de la aplicación de la métrica Árbol de Profundidad de Herencia (APH)

Esta métrica fue aplicada a una muestra de 11 clases:

Clases	Nivel APH
Tribunal	0
Sala	0
Turnado	0
Persona	0

TipoBanco	0
BaseController	0
GestionarTribunalesController	1
GestionarSalaController	1
BaseGtr	0
GestionarTribunalesGtr	1
SalaGtr	1

Tabla 6. Niveles APH de las clases de la muestra después de aplicada la métrica.

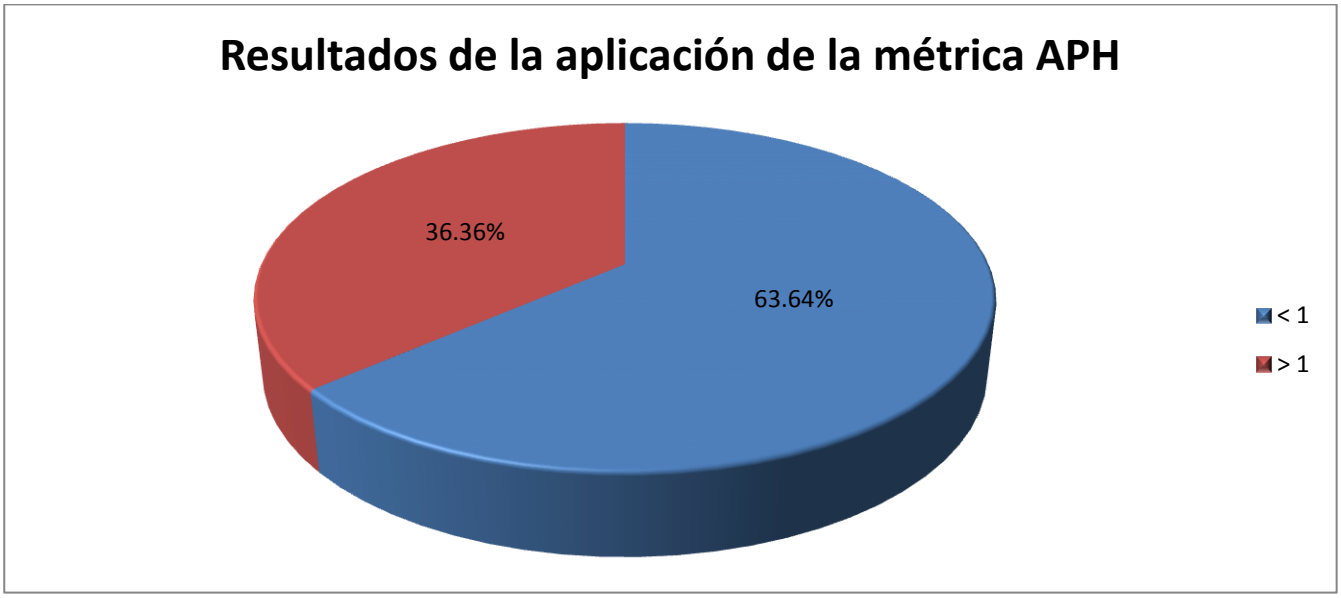

Figura 36. Representación en porciento de la aplicación de la métrica APH.

Analizando los resultados obtenidos se concluye que el 64% de la muestra presenta un nivel 0 de APH y el resto de las clases nivel 1. Este resultado representa un nivel óptimo de acuerdo a lo planteado por la métrica, evidenciando una baja complejidad en el diseño y asegurando que se prediga fácilmente el comportamiento de las clases.

3.2 Pruebas

Las pruebas de software (testing en inglés) son los procesos que permiten verificar y revelar la calidad de un producto software antes de su puesta en marcha. Básicamente, es una fase en el desarrollo de software que consiste en probar las aplicaciones construidas. En este sentido, las pruebas pueden

considerarse como un proceso que intenta proporcionar confianza en el software y cuyo objetivo fundamental es demostrar al desarrollador y al cliente que el software satisface sus requisitos. (21)

3.2.1 Pruebas Funcionales

Para la evaluación de la implementación del módulo Administración y Gobierno se decidieron realizar pruebas funcionales. El objetivo de este tipo de pruebas es detectar errores en la implementación de los requerimientos, el método seleccionado para realizar estas pruebas fue el método de caja negra.

3.2.1.1 Pruebas de Caja Negra

Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Ellas se concentran en los requisitos funcionales del sistema y los ejercitan. Estas pruebas permiten encontrar:

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

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están:

- ✓ **Técnica de la Partición de Equivalencia**: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- ✓ **Técnica del Análisis de Valores Límites**: esta Técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- ✓ **Técnica de Grafos de Causa-Efecto**: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Dentro del método de Caja Negra la técnica de la Partición de Equivalencia es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software y descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. Esta técnica fue la seleccionada para aplicarle al módulo Administración y Gobierno.

3.2.1.1.1 Técnica de la Partición de Equivalencia.

Una partición equivalente es una técnica de prueba de Caja Negra que divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba.

3.2.1.1.2 Casos de Prueba

Son un conjunto de condiciones o variables que el analista utilizará para determinar si el requisito de una aplicación es parcial o completamente satisfactorio.

Escenario	Descripción	Carné de identidad	Primer nombre	Segundo nombre	Primer apellido	Segundo apellido	Respuesta del sistema	Flujo central
EC 1.1 Adicionar abogado de oficio.	*El sistema permitirá adicionar un abogado de oficio, seleccionando un abogado existente para nombrarlo abogado de oficio.*	*V* *84080421634*	*V* *Luis*	*V* *NA*	*V* *Pérez*	*V* *Pupo*	*Adiciona el abogado seleccionado como abogado de oficio del tribunal.*	*1. Selecciona la opción Procedimientos en el menú superior. 2. Selecciona la sala o sección (En caso de que tenga asociada más de una sala o sección) 3. Selecciona la opción: Actividades del "cargo" (Ej.: Presidente de tribunal) 4.Selecciona la opción Gestionar abogado de oficio dentro de Gestionar usuario en el menú lateral izquierdo. 5. Selecciona la opción "Adicionar".*
EC 1.2 Buscar abogado.	*El sistema permitirá buscar a un Abogado a partir de los criterios introducidos.*	*NA* *Ver CU Buscar abogado*	*NA*	*NA*	*NA*	*NA*	*El sistema realizó la búsqueda satisfactoriamente según los criterios de búsquedas introducidos.*	
EC 1.3 La operación es cancelada.	*El sistema permitirá cancelar la operación.*	*NA*	*NA*	*NA*	*N A*	*NA*	*Cancela la operación y vuelve a la interfaz previa.*	*1.Selecciona la opción "Cancelar" 2. Cancela la operación y vuelve a la interfaz previa.*

Tabla 7. Caso de prueba asociado a la sección Adicionar Abogado de Oficio.

3.2.1.2 Resultados de las Pruebas de Caja Negra

El equipo de calidad del centro CEGEL realizó una serie de pruebas de caja negra al módulo Administración y Gobierno con el objetivo de validar su implementación. Se llevaron a cabo 3 iteraciones en las se detectaron un conjunto de No Conformidades (NC) que fueron resueltas de forma rápida y eficiente. En la 1era iteración se detectaron un total de 70 NC, en la 2da iteración 47 NC y en la 3era iteración no se encontraron NC. Estos resultados se muestran en la figura 37 y en ellos se evidencia la disminución del número de No Conformidades en cada iteración de cada caso de uso hasta llegar a un total de cero NC en la 3era iteración.

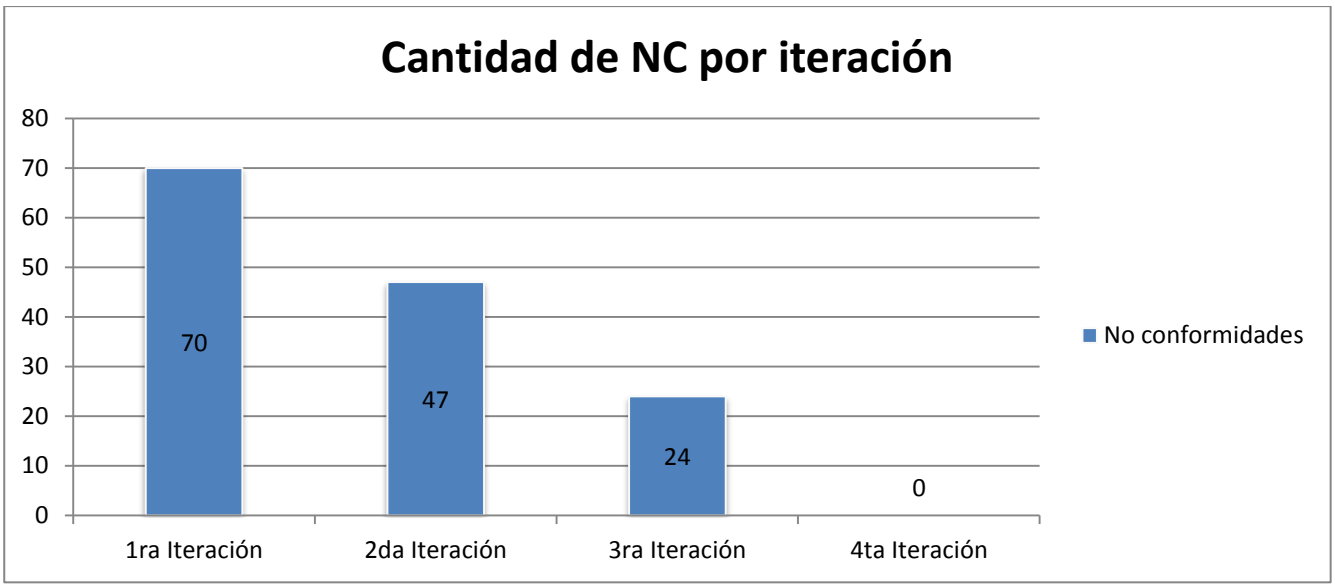

Figura 37. Representación del número de no conformidades por iteración.

3.3 Conclusiones Parciales

La calidad del módulo implementado y del diseño modelado fue la premisa fundamental en el desarrollo de este capítulo. En ese sentido se obtuvieron los casos de prueba que ayudan a comprobar el correcto funcionamiento del módulo desarrollado aplicando finalmente pruebas de caja negra. En general, los resultados obtenidos fueron favorables, desde el punto de vista funcional todos los requisitos realizan las funcionalidades requeridas y responden a las necesidades del cliente. Además se comprobó el grado mínimo de complejidad en el diseño de las clases y un bajo nivel de acoplamiento entre estas, logrando así un alto por ciento de reutilización.

CONCLUSIONES GENERALES

Con la culminación del presente trabajo de diploma se desarrolló el módulo Administración y Gobierno del Sistema de Informatización para la Gestión de los Tribunales Populares Cubanos (SITPC), contribuyendo a la agilización de los procesos judiciales para mejorar las actividades como el vencimiento de términos, gestión de tribunales, salas y materias, así como una serie de procesos de configuración necesarios para los restantes módulos del SITPC. Los resultados alcanzados permiten concluir:

- ✓ Se elaboró el marco teórico de la investigación a partir del estado del arte; el mismo evidenció la carencia de una solución informática capaz de responder a las necesidades y requerimientos de los TPC. Por otra parte, se analizaron diferentes tecnologías, lenguajes y herramientas indispensables para llevar a cabo el desarrollo de la investigación.
- ✓ Se realizó el diseño y la implementación del módulo Administración y Gobierno, como parte del SITPC, con el objetivo de erradicar los problemas existentes en los TPC para así mejorar el funcionamiento del proceso judicial en Cuba.
- ✓ Se aplicaron métricas al diseño realizado y pruebas al software implementado, las cuales demostraron el cumplimiento de diferentes atributos de calidad, lo que demuestra que el módulo Administración y Gobierno cumple satisfactoriamente con los requisitos definidos por los clientes.

RECOMENDACIONES

Después de haber cumplido los objetivos de la investigación y teniendo en cuenta las experiencias obtenidas de la misma, se recomienda:

- ✓ Realizar mejoras a las funcionalidades del módulo Administración y Gobierno utilizando las últimas versiones disponibles de las herramientas empleadas.
- ✓ Ejecutar pruebas piloto como interés de validación en la práctica y de aceptación al Módulo Administración y Gobierno, del cual forman parte indisoluble los componentes desarrollados en esta investigación.

BIBLIOGRAFÍA REFERENCIADA

1. **Téllez, Julio.** *Derecho Informático.* México : McGraw-Hill, 1996.
2. **Fayol, Henry.** *Administración industrial y general.* Buenos Aires : s.n., 1980.
3. **Peña, Carlos A.** *Informática Jurídica y Derecho Informático.* Palermo : s.n.
4. **Hernández Pérez, José Alfredo.** Geothesis. [En línea] Agosto de 2008. [Citado el: 25 de Noviembre de 2012.] http://www.geothesis.com/index.php?option=com_content&view=article&id=479:-mo-de-administracin-un-sistema-informco&catid=21:artulos&Itemid=100.
5. **Sánchez Delgado, Daylenis y Grenot Jardines, Juan Carlos.** *Desarrollo del subsistema Administración y Gobierno para el proyecto Informatización de los Tribunales Populares Cubanos.* La Habana : s.n., 2011.
6. **Catalago de Software**. *Catalago de Software.* [En línea] 14 de Junio de 2012. [Citado el: 6 de Diciembre de 2012.] http://www.catalogodesoftware.com/producto-gestion-de-contratacion-publica-y-privada-197.
7. **Juiz, O.** INFORME DEL SISTEMA SISECO. [En línea] 2009.
8. **Creative Common.** [En línea] 30 de Diciembre de 2006. [Citado el: 7 de Diciembre de 2012.] http://www.um.es/docencia/barzana/IAGP/Iagp2.html#BM4.
9. **Universidad Nacional Tecnológica del Cono Sur de Lima.** Tecnología de la Información. Procesos del ciclo de vida del software. [En línea] 28 de Julio de 2006. [Citado el: 7 de Diciembre de 2012.] http://es.scribd.com/doc/62686746/10/Metodologias-Pesadas.
10. **Amaro Calderon, Sarah Damaris y Valverde Rebaza, Jorge Carlos.** *Metodologías Ágiles.* Trujillo-Perú : s.n., 2007.
11. **Santiago Z, María de Lourdes.** Rational.com. *Desarrollando aplicaciones informáticas con el Proceso de desarrollo Unificado (RUP).* [En línea] [Citado el: 12 de Noviembre de 2012.] http://www.rational.com.ar/herramientas/rup.html.
12. **Sambayón Group.** *Introducción a RUP.* 2008.
13. **PostgreSQL.** *Manual del usuario de PostgreSQL.* 2008.
14. **Ardissone, Juan.** Maestro del Web. [En línea] [Citado el: 10 de Enero de 2013.] http://www.maestrosdelweb.com/editorial/curso-symfony2-introduccion-instalacion/.
15. **Sebastián, J.** *Modelo Vista Controlador- Definición y Características.* 2010.
16. **TIC.** Oposiciones TIC. [En línea] 8 de Junio de 2011. [Citado el: 7 de Enero de 2013.] http://oposicionestic.blogspot.com/2011/06/arquitectura-cliente-servidor.html.
17. **Perdomo, Jose Gregorio.** Arquitectura Multicapas. *Sistemas y Datos.* [En línea] 12 de Febrero de 2008. [Citado el: 7 de Enero de 2013.] http://sistemasydatos.blogspot.com/2008/02/arquitectura-multicapas.html.
18. **Sommerville, Ian.** *Ingenieria de Software. Séptima edición.* Madrid : Pearson Education S.A, 2005.
19. **Pressman, Roger S.** *Ingeniería del Software. Un enfoque práctico.* s.l. : Mac Graw Hill, 2002.
20. **Vega Lebrún, Carlos, Rivera Prieto, Laura Susana y García Santillán, Arturo.** *Mejores Prácticas para el establecimiento y aseguramiento de la calidad de software* . ISBN.
21. **Carreras Profesionales CIBERTEC.** *Pruebas de Software.* 2009.

BIBLIOGRAFÍA CONSULTADA

1. **González, R.A.H.L.y.S.C.,** El proceso de investigación científica.(2011): Editorial Universitaria del Ministerio de Educación Superior. 110.
2. **Sampieri, R.H.,** Metodologia De La Investigacion. (1991).
3. **Appleton, B**. (02/14/2000) "Patterns and Software: Essential Concepts and Terminology."
4. **Productiva,** D. d. C. d. l. I. (2012). MODELO DE SISTEMA V2.0. SITPC. Módulo Administración y Gobierno: 354.
5. **Jurisoft, E. "Infolex - Gestion de Despachos**, tomado dehttp://www.infolex.es/ie/index.aspx.
6. **Software, B**. "GEDEX Software para la Gestión de Expedientes Jurídicos para Despachos de **Abogados y Profesionales**, tomado dehttp://www.brindys.com/gedex/casmenu.html.
7. **SRL, E. S. J. (2011).** "Lex-Doctor, GESTIÓN JURÍDICA."tomado dehttp://www.lex-doctor.com/productos_estudiosjuridicos_info.php.
8. **Morell, M. D. E. C. and M. J. R. G**. Guadarramas (2008). Sistema para la Tramitación de Procesos Penales. Facultad Matemática-Física-Computación. Villa Clara,Cuba, Universidad Central „Marta Abreu“ de Las Villas.
9. **Lira, K. G., Y. D. Rivero**, et al. (2010). "ESTRATEGIA DE MODELADO DE PROCESOS DE NEGOCIO PARA EL DESARROLLO DE SISTEMAS DE INFORMÁTICA JURÍDICA.".
10. **"Ingenieros de Software. Patrones de diseño."**, tomado de http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.
11. **"Documentación-Manual de PHP."** tomado de http://www.hospedajeydominios.com/mambo/documentacion-manual_php.html.
12. **La biblia Servidor Apache 2**.
13. **"NetBeans IDE. Sitio Oficial."**, tomado de http://netbeans.org/.
14. **"¿Qué es Doctrine ORM?"**, tomado de http://www.tecnoretales.com/programacion/que-es-doctrine-orm/.
15. **Gamma, E., R. Helm**, et al. Design Patterns.
16. **Vizcaíno, A., F. Ó. García**, et al. (2008). Trabajando con Visual Paradigm for UML. Univ. Cantabria – Fac. de Ciencias.

GLOSARIO

SITPC: Sistema de Informatización para la Gestión de los Tribunales Populares Cubanos.

TPC: Tribunales Populares Cubanos.

DCD: Diagrama de clases del diseño.

CU: Caso de uso.

DS: Diagrama de secuencia.

DC: Diagrama de componente.

CP: Casos de prueba.

NC: No conformidades, problemas detectados en un artefacto según insatisfacción con el resultado final de un Elemento de Configuración, lo pactado con anterioridad con el cliente, o no cumplimiento de un requisito.

CEGEL: Centro de Gobierno Electrónico.