Universidad de las Ciencias Informáticas

Facultad 3

CEGEL

“Desarrollo del módulo Casación del subsistema Administrativo del proyecto Sistema de Informatización de los Tribunales Populares Cubanos”

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

Autores:

Leosbel Poll Sotomayor

Miguel Esteban Gómez Martínez

Tutor:

Ing. Pedro Frank Cadenas del Llano

La Habana, Junio 2015

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 _____.

Leosbel Poll Sotomayor

_____

Firma del Autor

Miguel Esteban Gómez Martínez

_____

Firma del Autor

Ing. Pedro Frank Cadenas del Llano

_____

Firma del Tutor

AGRADECIMIENTOS

Miguel: Agradezco a toda mi familia que está presente en estos momentos a mis tías Gladys, Alicia y Lissette que dieron su granito de arena para que todo esto fuera posible, a mi papá, a mis amigos y a mi primo pedro. También le agradezco a mi tutor que jugó un papel muy importante en el desarrollo de este trabajo, a Yoraima, gracias por tu ayuda y por tener tanta paciencia con nosotros, a mi compañero de tesis que ha sido un dolor de cabeza en todo este tiempo pero sin duda alguna ha sido un placer haber hecho la tesis con él. Finalmente y no menos importante le agradezco a mi madre que me apoyó en todas mis decisiones y estuvo siempre presente cuando más la necesité y que sacrificó muchos años de su vida para que yo pudiera terminar mi carrera.

Leosbel: Quiero agradecer a mis padres y a mi hermano que aun estando casi siempre lejos de mí, todo el tiempo permanecieron presentes en mi cabeza. A mis hermanos del barrio Osmel y Melian, y a mi hermanita Surelis, que siempre creyeron 100% en mis decisiones y me apoyaron en lo necesario para hacerlas realidad. A mi compañero de tesis, que ese si me ha dado dolores de cabeza, pero solo porque lo quiero como a mi hermanito. A mi tutor, más que tutor ha sido un amigo para nosotros,

gracias por los sabios consejos de "busca bien, que por ahí esta el error". A los amigos de la Universidad quienes me dieron su apoyo siempre que lo necesité en especial al "jose". A mi hermanito René, quién fue compañero de cuarto, de GJM, de muchísimos momentos buenos y malos exceptuando los del dómino. A todas las personas que brindaron incondicionalmente su tiempo y esfuerzo para que este trabajo se hiciera realidad. Por último quiero agradecer a una personita que ha cambiado mi vida en estos años, que a pesar de mi carácter ha hecho de mí una mejor persona, les hablo de mi esposa Betty que junto a su familia lograron hacerme sentir parte de ella. A todos muchas gracias.

DEDICATORIA

Miguel: Este trabajo se lo dedico a mi familia y en especial a mi madre que es la que ha estado a mi lado en todos los momentos tantos buenos como malos.

Leosbel: Este trabajo va dedicado a mis padres y en especial a mí.

RESUMEN

Actualmente los Tribunales Populares Cubanos manejan una gran cantidad información, por lo que su almacenamiento es cada vez más complicado, ya que existen factores como la humedad que provoca el deterioro de dicha información. Por otra parte, todos los procesos que se llevan a cabo en las diferentes materias se realizan de forma manual, dependiendo totalmente de documentos escritos en papel, lo cual permite la duplicidad y la pérdida de dicha información. A raíz de todos estos problemas surge la necesidad de controlar estos documentos dando paso a la creación de este trabajo. Es por este motivo que el Tribunal Supremo Popular de Cuba llegó a un convenio con la Universidad de las Ciencias Informáticas creándose de esta forma el proyecto Sistema de Informatización de los Tribunales Populares Cubanos. Mediante la utilización de las principales tecnologías y tendencias de desarrollo de software que se emplean en el proyecto en su primera fase se plantea una solución para economizar esfuerzos mediante el desarrollo del paquete de análisis, “Interposición del recurso”. Se ofrecen todos los artefactos obtenidos durante el diseño e implementación de los componentes, los cuales son evaluados mediante un conjunto de pruebas y métricas, a través de las cuales se validará su diseño y funcionamiento.

ÍNDICE

INTRODUCCIÓN

En la actual sociedad, es indispensable que el tiempo de resolución y respuesta a los problemas que surgen sea el menor posible, así como la prestación eficiente y eficaz de servicios. El constante desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) ha dado al traste con este fenómeno, contribuyendo con el cumplimiento de las necesidades de la población antes mencionadas. Es por ello que numerosos países, incluida Cuba, ha involucrado las TIC en todas las ramas productivas y de servicios. El sector jurídico no ha estado exento de este desarrollo, y aunque nuestro país no cuenta con las últimas tecnologías, los Tribunales Populares Cubanos (TPC) también se han sumado a este universo de avance tecnológico, donde se están dando pasos para facilitar, agilizar y organizar los procesos judiciales a través de una justicia tecnológicamente avanzada.

Los TPC manejan un gran cúmulo de información, conformando así los expedientes de los casos que se tramitan. El almacenamiento y búsqueda de todos los documentos en muchas ocasiones se dificulta, ya que existe la posibilidad de que no esté disponible debido a que otra persona puede estar utilizándolo. Dichos documentos se generan de forma manual, propiciando que se cometan errores, alargando estos procesos debido a la necesidad de realizar correcciones, trayendo como consecuencia que para cuando se corrijan estos errores ya el proceso estaría fuera de término. Los expedientes que surgen en cada proceso son guardados en papel, el cual se deteriora con el paso del tiempo, provocando así la pérdida de la información sensible que contienen. De esta forma se evidencia la necesidad de controlar los documentos que se generan de forma tal que dichos problemas puedan ser erradicados.

Para dar solución a estos problemas, los TPC solicitaron al Centro de Gobierno Electrónico (CEGEL) de la Universidad de las Ciencias Informáticas (UCI) la creación de una herramienta que permita informatizar los procesos que se realizan en dichas instituciones, creándose de esta forma el proyecto: Sistema de Informatización de los Tribunales Populares Cubanos (SITPC). Los TPC están organizados en tres instancias: supremo, provincial y municipal, donde cada uno realiza procesos de diversas materias: Civil, Administrativo, Laboral, Económico y Penal, cada una de estas a su vez presenta diferentes procedimientos.

Con el objetivo de organizar y brindar una mejor solución, se decidió distribuir el desarrollo de todo el proyecto en varios subsistemas y módulos, constituyendo cada materia un subsistema y cada procedimiento un módulo a implementar.

El recurso de Casación es un medio de impugnación que tiene por objeto anular una sentencia judicial, que contiene una incorrecta interpretación o aplicación de la ley, o que ha sido dictada en un procedimiento que no ha cumplido las solemnidades legales. En el caso de la materia Administrativa después de dictada la sentencia en la instancia provincial, una parte postula la

revisión de los errores jurídicos atribuidos a la misma, reclamando la correcta aplicación de la ley sustantiva o la anulación de la sentencia, y una nueva decisión. Este proceso tiene lugar en la sala de lo Administrativo del TSP, por lo que no está exento de los problemas planteados anteriormente que aquejan a los TPC.

Por lo anteriormente expuesto se plantea como **problema a resolver**: ¿Cómo gestionar los procesos asociados al recurso de Casación de la materia Administrativa de manera que se garantice el control de la información generada?

Se define como **objeto de estudio**: Sistemas informáticos para la gestión de los procesos en los Tribunales Populares Cubanos.

Para darle solución al problema planteado se tiene como **objetivo general**: Desarrollar un sistema informático para la gestión de la información generada en los procesos de Casación de la materia Administrativa del Tribunal Provincial Popular que garantice el control de la misma.

La presente investigación se encuentra enmarcada en el **campo de acción**: Informatización del procedimiento Casación de la materia Administrativa del Tribunal Supremo Popular.

Conforme a lo antes descrito se define como **Idea a defender**: Si se desarrolla una herramienta informática que gestione los procesos asociados al recurso de Casación de la materia Administrativa, se garantizará el control de la información generada.

Para cumplir con el objetivo general se trazan los siguientes **objetivos específicos**:

- Elaborar el marco teórico de la investigación para justificar las principales tecnologías asociadas al desarrollo de software.
- Aplicar las técnicas de captura de requisitos para realizar la especificación de los mismos solicitados por el cliente.
- Realizar el análisis de los requisitos pactados con el cliente para diseñar el diagrama de caso de uso.
- Efectuar la implementación del módulo Casación del subsistema Administrativo para darle solución a los problemas planteados.
- Validar los resultados obtenidos mediante la ejecución de pruebas de *software* para verificar la calidad de los mismos.

Para dar cumplimiento al objetivo planteado y ofrecer una solución eficiente al problema se definen las siguientes **tareas de la investigación** a cumplir:

1. Identificación de los procesos que se tratan en el módulo Casación del subsistema Administrativo en los TPP para tener conocimiento de los mismos.

2. Caracterización del proceso de desarrollo de *software* y su metodología.
3. Caracterización de la plataforma de desarrollo y de las herramientas a utilizar para diseñar, implementar y probar el sistema.
4. Identificación, análisis y especificación de los requisitos de *software* del módulo Casación del subsistema Administrativo, con el objetivo de tener todas las funcionalidades requeridas y obtener una definición precisa, completa y verificable de cada uno de los requisitos.
5. Realización del diagrama de proceso de negocio.
6. Realización de casos de uso del diseño (diagramas de comportamiento).
7. Realización de los diagramas de clases del diseño.
8. Caracterización de los paradigmas de programación para utilizar buenas prácticas.
9. Caracterización y selección de los patrones de diseño factibles para la propuesta de solución.
10. Realización de los diagramas de secuencia.
11. Realización del modelo de datos.
12. Realización del diagrama de despliegue.
13. Implementación de los casos de uso.
14. Diseño de casos de prueba para validar las funcionalidades.

Para el estudio preliminar de la investigación se utilizaron los siguientes métodos científicos:

Métodos Teóricos:

- Analítico-sintético: se evidenció mediante el estudio y análisis realizado de la bibliografía. Permitió seleccionar correctamente los conceptos y definiciones con el objetivo de mejorar la comprensión del problema, procurar el cumplimiento de los objetivos y así lograr resultados satisfactorios en la investigación.
- Histórico-lógico: permitió estudiar la evolución mediante la informática de los procesos judiciales, las soluciones creadas para este fin y en qué medida se relacionan con la presente investigación.
- Modelación: se empleó en la creación de modelos que simbolizan abstracciones, con el objetivo de modelar la estructura y funcionamiento de la implementación del módulo Casación.

Métodos Empíricos:

- Tormenta de ideas: permitió la obtención acertada de los requisitos a implementar, así como el procedimiento a seguir para la creación de esta investigación.
- *Test:* se empleó mediante la realización de las pruebas al módulo Casación, con el objetivo de obtener el grado de calidad y fiabilidad de la solución propuesta.

Estructura del documento:

Capítulo I: Fundamentación teórica. Se realiza un estudio del estado del arte referente a los sistemas informáticos para la gestión judicial, además se exponen los conceptos fundamentales relacionados con el tema de la investigación, de igual forma se describen la metodología, marco de trabajo, herramientas, lenguajes de programación y patrones de diseño ya definidos por la arquitectura que se emplea en el desarrollo de la solución.

Capítulo II: Solución propuesta. En este capítulo se presenta la solución técnica de la investigación. Se evidencia el modelo de negocio con sus principales actores, estructurado por el diagrama de proceso de negocio .Se expone el modelo de diseño conformado por los diagramas de casos de uso, diagrama de clases y los diagramas de secuencia (diagramas de interacción), así como el modelo de implementación con el diagrama de despliegue y el de componente. Se muestra además el modelo de datos compuesto por el diagrama entidad-relación.

Capítulo III: Validación de resultados. Se evalúa el grado de fiabilidad y calidad de los resultados obtenidos con la creación de este trabajo, aplicando algunas de las métricas asociadas a la medición de la calidad del diseño y la implementación de *software*, que se emplean con este fin internacionalmente.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se exponen los conceptos fundamentales para la comprensión del procedimiento de Casación de la materia Administrativa de los TPC. Se analizan algunos de los sistemas internacionales existentes, que presentan características similares a la solución que se necesita obtener. También se fundamenta la selección de la metodología, marco de trabajo, herramientas, lenguajes de programación y patrones de diseño que serán empleados en el desarrollo del proyecto SITPC.

1.2 Conceptos generales

1.2.1 Sistema de Tribunales Populares Cubanos

El Sistema de Tribunales Populares Cubanos está constituido por el Tribunal Supremo Popular (TSP), los TPP y los Tribunales Municipales Populares (TMP). El TSP ejerce su jurisdicción en todo el territorio nacional y tiene su sede en la capital de la República. En cada provincia existe por lo menos un Tribunal Provincial. Los TPP ejercen su jurisdicción en el territorio de las correspondientes provincias y tienen sus sedes donde determine el Consejo de Gobierno del Tribunal Supremo Popular. Los TMP ejercen su jurisdicción en el territorio correspondiente a los municipios en que radiquen y, tienen su sede en la cabecera de estos, aunque puede situarse en lugar distinto; siempre dentro del territorio del municipio, por determinación del Consejo de Gobierno del Tribunal Supremo Popular, cuando las circunstancias así lo ameriten. Los TPC constituyen un sistema de órganos estatales, estructurado con independencia funcional de cualquier otro, y subordinado jerárquicamente a la asamblea nacional del poder popular y al consejo de estado. Se rigen por los principios consagrados en la constitución, que norman la organización y el funcionamiento de los órganos estatales (Ministerio de Justicia, 2008).

1.2.2 Materia Administrativa

La Jurisdicción de lo Contencioso-Administrativo es destinada al conocimiento y aplicación del derecho Administrativo, es decir, el referente al conjunto normativo destinado a la regulación de la actividad de la administración pública en su versión contenciosa o de control de la legalidad y de sometimiento de esta a los fines que la justifiquen. Así como para atender los recursos de los administrados contra resoluciones de la administración que consideran injustas (Constantinos Stamatoulos, 2014).

1.2.3 Recurso de Casación

El recurso de casación es un recurso extraordinario que tiene por objeto anular una sentencia judicial que contiene una incorrecta interpretación o aplicación de la ley o que ha sido dictada en un procedimiento que no ha cumplido las solemnidades legales. Su fallo le corresponde a la corte

nacional de justicia y, habitualmente al de mayor jerarquía, como el Tribunal Supremo. Sin embargo, en ocasiones también puede encargarse del recurso un órgano jurisdiccional jerárquicamente superior o en su caso uno específico (Constantinos Stamatoulos, 2014).

1.3 Sistemas informáticos jurídicos de gestión existentes a nivel internacional

1.3.1 Sistema jurídico INTEGRA

Software jurídico desarrollado por PROMAD (Programa Nacional de Modernización de la profesión Legal), la cual posee como principal objetivo modernizar la profesión legal brasileña, ofreciendo dos herramientas: el sistema jurídico INTEGRA y el sitio del abogado en internet. INTEGRA es una solución que ofrece organización a los procesos llevados a cabo en las distintas compañías y bufetes de abogados en Brasil. Su objetivo es procurar un mejor control en cuanto a las agendas de reuniones, encuentros con clientes y procesos judiciales. Cuenta con una comunidad de seguidores que ayuda a la retroalimentación del sistema, se puede explotar gratuitamente hasta dos años, su obtención pasado este período depende del momento de la contratación, variando su precio (PROMAD, 2013).

1.3.2 Aranzadi Infolex Nube

Solución española de gestión de despacho, perteneciente a la compañía *Thomson Reuters*, dicha solución surge de la incorporación a dicha empresa de Aranzadi en 1999 y de Jurisoft en el 2011. Este sistema brinda medios para la gestión integral de todas las áreas de negocio de abogados, procuradores y servicios jurídicos de empresas, así como asociaciones y administraciones públicas. Centraliza en una única aplicación el control de los expedientes jurídicos, expedientes administrativos, informes y expedientes consultivos, así como la visión del estado de los expedientes por parte de los involucrados en el proceso, su costo puede variar (Thomson Reuters, 2014).

1.3.3 Abogafín

Software jurídico para la gestión de expedientes en despachos de abogados, bufetes, departamentos jurídicos y procuradores. Forma parte de la gama de productos informáticos de Netfincas *software*, fundada en el 2002. Esta empresa española ha estado desde entonces en el negocio de la investigación y desarrollo de *software*, aportando soluciones informáticas para administradores de fincas, asesorías, despachos jurídicos y gestores inmobiliarios. Abogafín permite disponer del control total sobre los expedientes jurídicos de un despacho, realizando un correcto seguimiento de los mismos, tiene incorporado un sistema de documentación adjunta, además permite el control de clientes, al poder contactarlos de manera directa a través de la propia aplicación. Integra todos los libros oficiales para profesionales, libro de gastos e ingresos, provisiones y suplidos. (NetFincas Software, 2008).

1.4 Sistemas Jurídicos existentes en Cuba

1.4.1 SisProp

Este *software* 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.

El sistema funciona en un entorno de red local, en el cual una de las máquinas funciona como servidor de bases de datos, y los jueces y los secretarios de la sala penal acceden desde sus estaciones de trabajo al sistema. El cliente se conecta a la base de datos y le permite acceso a la información pertinente para cada uno de ellos.

Debido a las características de este *software* se decidió aplicar la tecnología cliente-servidor y utilizar como sistema de gestión de bases de datos a MS SQL Server 2000 y como entorno de programación Borland Delphi 6 y por medio de la tecnología ADO (*Access Data Object*) lograr la conexión desde las aplicaciones clientes a la base de datos (Morell, 2008).

Los sistemas informáticos mencionados anteriormente son propietarios, por lo que el costo de su adquisición y licencia de uso son elevados, no son multiplataforma, incumpliendo con la política de migración a software libre de Cuba, las funcionalidades que poseen no son compatibles con los procesos que se llevan a cabo en los TPC, además de no contener el procedimiento de casación, por lo que no cumplen con los requisitos que necesita la solución. Por ello es necesario desarrollar un sistema nuevo que gestione los procesos del procedimiento casación que se llevan a cabo en la materia Administrativa de los TPC.

1.5 Metodología de desarrollo de *software*

Para el desarrollo de una aplicación con calidad es necesario utilizar una serie de procedimientos, técnicas, herramientas y soporte documental. Una metodología de desarrollo de *software* es un enfoque estructurado para la creación de *software* que incluye modelos de sistemas, notaciones, reglas, sugerencias de diseño y guías de procesos (Chiesa, 2004).

1.5.1 Capacidad de Madurez para la Integración del Modelado (CMMI)

Es un conjunto de herramientas que ayudan a una organización a mejorar sus procesos de desarrollo de productos y servicios, adquisiciones y mantenimiento; por lo que en la Universidad de las Ciencias Informáticas se establece el siguiente ciclo de vida básico para los proyectos del alcance del Programa de Mejoras: (Universidad de las Ciencias Informáticas, 2010)

- **Estudio Preliminar:** En esta fase se llevan a cabo las actividades relacionadas con la planeación del proyecto a un alto nivel. En esta fase se realiza un estudio inicial de la

organización cliente que permite obtener información fundamental acerca del alcance del proyecto y realizar estimaciones de tiempo, esfuerzo y costo.

- **Modelado del Negocio:** Es la fase destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea automatizar para tener garantías de que el software desarrollado va a cumplir su propósito. Para la descripción y modelado del negocio pueden ser utilizadas diferentes técnicas como el Modelado de Casos de Uso del Negocio y BPMN.

Artefacto que se genera:

✓ Modelo de procesos de negocio.

✓ Reglas del negocio.

- **Requisitos:** El esfuerzo principal en esta fase es desarrollar un modelo del sistema que se va a construir. Incluye un conjunto de casos de uso, servicios que describen todas las interacciones que tendrán los usuarios con el software, estos responden a los requisitos funcionales del sistema. Además la especificación de requisitos incluye requisitos no funcionales.

Artefacto que se genera:

✓ Especificación de requisitos de software.

- **Análisis y Diseño:** Durante esta fase, de considerarse necesario, a través de los modelos de análisis, los requisitos descritos pueden ser refinados y estructurados para conseguir una comprensión más precisa de los mismos y una descripción que sea fácil de mantener y ayude a estructurar el sistema. Además en ella es modelado el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Esto contribuye a una arquitectura sólida y estable que se convierte en un plano para la implementación. Los modelos desarrollados en esta etapa son más formales y específicos de una implementación.

Artefactos que se generan:

✓ Especificación de casos de uso del sistema.

✓ Diagramas de secuencia.

✓ Diagramas de clases del diseño.

✓ Diagrama de despliegue.

- **Implementación:** En la implementación a partir de los resultados del análisis y diseño se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ejecutables y similares.

Artefacto que se genera:

✓ Diagrama de componentes.

- **Pruebas Internas:** Durante esta fase el proyecto verifica el resultado de la implementación probando según sea necesario cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas.

Artefacto que se genera:

✓ Diseño de casos de prueba.

- **Pruebas de Liberación:** Pruebas diseñadas e implementada por el laboratorio del proyecto SITPC a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.
- **Despliegue:** Durante esta fase se procede a la entrega de la solución, así como a la instalación, configuración, prueba y puesta en marcha del software en el entorno real del cliente. Las pruebas de esta fase incluyen pruebas de aceptación y pruebas pilotos. También deben realizarse en este periodo la capacitación y acompañamiento a clientes para asegurar que adquieran los conocimientos necesarios en la manipulación del software.
- **Soporte:** Durante esta fase y por un tiempo limitado el proyecto ofrecerá un servicio para resolver conflictos y problemas de usabilidad y rendimiento del software entregado al cliente, suministrándole actualizaciones y parches a errores.

1.5.2 RUP: Proceso Unificado de Desarrollo

RUP es un proceso de ingeniería de software que proporciona un enfoque disciplinado para asignar tareas y responsabilidades dentro de un equipo de desarrollo. Su objetivo principal es asegurar la entrega de un producto de alta calidad que cubra los requerimientos de los usuarios finales dentro de los tiempos y el presupuesto (Botero, 2014).

.**Características esenciales**

Los autores de RUP destacan que el proceso de software propuesto por RUP tiene tres características esenciales:

- Dirigido por los Casos de Uso
- Centrado en la arquitectura
- Es iterativo e incremental.

Proceso dirigido por casos de uso

Utiliza los casos de uso para el desarrollo de las disciplinas con los artefactos, roles y actividades necesarias. Los casos de uso son la base para la implementación de las fases y disciplinas del RUP.

Proceso centrado en la arquitectura

Arquitectura de un sistema es la organización o estructura de sus partes más relevantes. Una arquitectura ejecutable es una implementación parcial del sistema, construida para demostrar algunas funciones y propiedades. RUP establece refinamientos sucesivos de una arquitectura ejecutable, construida como un prototipo evolutivo.

Proceso iterativo e incremental

La estrategia que se propone en RUP es tener un proceso iterativo e incremental en donde el trabajo se divide en partes más pequeñas o mini proyectos. Permitiendo que el equilibrio entre casos de uso y arquitectura se vaya logrando durante cada mini proyecto, así durante todo el proceso de desarrollo. Cada mini proyecto se puede ver como una iteración (un recorrido más o menos completo a lo largo de todos los flujos de trabajo fundamentales) del cual se obtiene un incremento que produce un crecimiento en el producto. El ciclo de vida iterativo se basa en la evolución de prototipos ejecutables que se muestran a los usuarios y clientes (Morano, 2010).

En el caso del SITPC, se decidió utilizar CMMI trabajando en conjunto con RUB utilizando del mismo sus 3 características esenciales como son: Dirigido por caso de uso, centrado en la arquitectura e Iterativo Incremental. Según las necesidades del proyecto en el ciclo de vida se pueden realizar iteraciones a partir de la fase de negocio, de esta forma de adecúan a las características del SITPC del centro CEGEL de la Facultad 3.

1.6 Lenguaje de Modelado

El lenguaje de modelado es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar parte o un diseño de *software*. Este debe poseer una semántica precisa que permita la interpretación unívoca de los modelos. El modelado es esencial en la construcción de *software* para comprender mejor lo que se crea, puntualizar el comportamiento deseado del sistema, descubrir oportunidades de simplificación y reutilización (Larman, 1999).

1.6.1 UML: Lenguaje de Modelado Unificado

UML es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema de *software* orientado a objetos, su empleo para la modelación de distintos tipos de sistemas: sistemas de *software*, sistemas de *hardware*, y organizaciones del mundo real, constituye una de sus principales ventajas. UML es una consolidación de muchas de las notaciones y

conceptos más usadas orientados a objetos, incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de *software* reutilizables (Larman, 1999).

1.6.2 BPMN: Notación para el Modelado de Procesos de Negocio

BPMN es una notación gráfica estandarizada para el modelado de procesos de negocio en un formato de flujo de trabajo. Su principal objetivo es proveer una notación estándar fácilmente legible y entendible por parte de todos los involucrados e interesados del negocio, entre los mismos se encuentran los analistas de negocio, quienes definen y redefinen los procesos, los desarrolladores responsables de implementar los procesos, los gerentes y administradores del negocio, quienes monitorean y gestionan los procesos. En síntesis BPMN tiene la finalidad de servir como lenguaje común para cerrar la brecha de comunicación que frecuentemente se presenta entre el diseño de los procesos de negocio y su implementación (Stephen A. White, 2009).

1.7 Herramienta CASE

La tecnología CASE supone la automatización del desarrollo del software, contribuyendo a mejorar la calidad y la productividad en el desarrollo de sistemas de información y se plantean los siguientes objetivos:

- Permitir la aplicación práctica de metodologías estructuradas, las cuales al ser realizadas con una herramienta se consigue agilizar el trabajo.
- Facilitar la realización de prototipos y el desarrollo conjunto de aplicaciones.
- Simplificar el mantenimiento de los programas.
- Mejorar y estandarizar la documentación.
- Aumentar la portabilidad de las aplicaciones.
- Facilitar la reutilización de componentes software.
- Permitir un desarrollo y un refinamiento visual de las aplicaciones, mediante la utilización de gráficos. (ERNESTO CRUZ ORELLANA, 2006)

1.7.1 Visual Paradigm

Es una herramienta CASE que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, implementación y pruebas. Ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite construir diagramas de diversos tipos, código inverso, generar código desde diagramas y generar documentación. La herramienta también proporciona abundantes tutoriales, demostraciones interactivas y proyectos (Torres, 2011).

Se decidió emplear Visual Paradigm 8.0 como herramienta CASE en el proyecto SITPC ya que la universidad posee licencia para su uso, además se puede utilizar tanto en Linux como en Windows, por ser una herramienta multiplataforma, elemento acorde con la política de migración a *software* libre trazada por el país.

1.8 Marcos de Trabajo de desarrollo

Un Marco de Trabajo es una estructura de soporte que incluye artefactos o módulos del *software*, partiendo de este, otros pueden ser desarrollados sin necesidad de comenzar desde cero. Los Marcos de Trabajo orientados a objeto están estructurados en bibliotecas de clases por lo cual evitan frecuentemente el trabajo arduo de la programación (Larman, 2005).

Para el desarrollo del presente trabajo se define como marco de trabajo Symfony en su versión 2.1, para el trabajo con el diseño de las vistas Bootstrap en su versión 2.1 y para el acceso a los datos Doctrine en su versión 2.0. A continuación se describe con más detalles cada una de estas tecnologías:

1.8.1 Symfony 2.1

Symfony 2.1 ha sido ideado para exprimir al límite todas las nuevas características de php 5.3 siendo por este motivo uno de los marcos de trabajo PHP (*Hypertext Preprocessor*) con mejor rendimiento. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en un proyecto. Proporciona varias herramientas, encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web (Eguiluz Pérez, 2012).

1.8.2 Bootstrap 2.1

Bootstrap es un marco de trabajo diseñado para simplificar el proceso de creación de diseños web. Para ello ofrece una serie de plantillas css y de ficheros JavaScript, los cuales permiten conseguir:

- Interfaces que funcionen de manera brillante en los navegadores actuales y correctos en los no tan actuales.
- Un diseño que pueda ser visualizado de forma correcta en distintos dispositivos y a distintas escalas y resoluciones.
- Una mejor integración con las librerías que suelen usar habitualmente, como por ejemplo jQuery.
- Un diseño sólido basado en herramientas actuales y potentes estándares como CSS3/HTML5 (Genbeta, 2012).

1.8.3 Doctrine 2.0

Doctrine 2 es un mapeador-objeto-relacional (ORM) para PHP 5.3 que proporciona persistencia transparente de objetos PHP. Se sitúa en la parte superior de una poderosa capa de abstracción de base de datos. La principal tarea de los mapeadores objeto-relacionales es la traducción transparente entre objetos (PHP) y las filas relacionales de la base de datos. Está dividido en dos capas principales, la DBAL (*Database Abstraction Layer*) y el ORM (Doctrine Project Team, 2011).

1.9 Otras tecnologías utilizadas

1.9.1 jQuery 1.8

Biblioteca para el lenguaje JavaScript que ofrece una infraestructura para crear fácilmente aplicaciones complejas del lado del cliente, sirve de ayuda en la creación de interfaces de usuario, efectos dinámicos y aplicaciones que hacen uso de Ajax. Este marco de trabajo tiene licencia para uso en cualquier tipo de plataforma, personal o comercial, es un producto serio, estable, bien documentado y con un gran equipo de desarrolladores a cargo de la mejora y actualización. Cuenta también con una gran comunidad de creadores de plugins o componentes, lo que permite encontrar soluciones ya creadas para implementar diferentes elementos (Eguíluz Pérez, 2008).

1.9.2 Subversion 1.5

- Es una herramienta de *software* libre para el control de versiones.
- Subversion es capaz de manejar los cambios hechos sobre ficheros y directorios a lo largo del tiempo.
- Permite recuperar versiones antiguas y ver el historial de cambios de un sistema de archivos.
- Es una herramienta distribuida capaz de trabajar sobre diversos protocolos (svn, ssh, http, https).
- No es una herramienta de gestión de la configuración de *software*, principalmente porque no está diseñada para manejar *software*, sino archivos arbitrarios (Sáchez, 2004).

1.9.3 Twig 1.0

Características:

- Es un motor de plantillas moderno basado en PHP5.
- Flexible, desarrollo de propios filtros
- Contiene herencia de plantillas
  - ✓ Include
  - ✓ Extens

Ventajas:

- Separa la capa de la Vista del Modelo/Controlador.
- No se mezcla PHP/HTML en las plantillas.

- Sintaxis simple, fácil de entender.
- Se integra con los principales IDEs.
- Se puede integrar en otros proyectos de forma independiente (Lemus, 2014).

1.10 Lenguaje de Programación

Un lenguaje de programación es un idioma artificial utilizado por los programadores para dar indicaciones precisas a una computadora, de cómo se debe comportar un *software* a nivel físico o lógico. Permite especificar sobre qué datos debe operar un ordenador, la manera de almacenarlos, transmitirlos y qué acciones llevar a cabo bajo una variedad de circunstancias.

1.10.1 HTML5

Es el lenguaje de marcado de hipertexto usado para la construcción de páginas web. Fue creado en 1986 por Tim Berners Lee; el cual tomó dos herramientas preexistentes: El concepto de Hipertexto el cual permite conectar dos elementos entre si y el SGML (Lenguaje Estándar de Marcación General) el cual sirve para colocar etiquetas o marcas en un texto que indique como debe verse. HTML no presenta ningún compilador, por lo tanto algún error de sintaxis que se presente, éste no lo detectará y se visualizará en la forma como éste lo entienda. El conjunto de etiquetas que se creen para la construcción de una página web, se deben guardar con la extensión *.htm* o *.html*. Estos documentos pueden ser mostrados por los visores o "browsers" de páginas Web en Internet, como Netscape Navigator, Mosaic, Firefox, Opera entre otros (Gutiérrez, 2012)

1.10.2 CSS3

CSS es un lenguaje de hojas de estilos creado para controlar la presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para la creación de páginas web complejas. La separación de los contenidos y su presentación presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados “documentos semánticos”). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes (Eguíluz Pérez, 2013).

1.10.3 JavaScript 2.0

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas[1]. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos

[1] Una página web dinámica es aquella que incorpora efectos como texto que aparecen y desaparecen, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.

con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java (Eguiluz Pérez, 2009).

1.10.4 PHP 5.3

Lenguaje orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos. Puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo. Es un lenguaje flexible, potente y de alto rendimiento conocido hasta el día de hoy. El parecido de este, con los lenguajes más comunes de programación estructurada; como C y Perl, permiten a la mayoría de los programadores crear aplicaciones complejas, además de involucrarse con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones (PHP Documentation Group , 2014).

Como lenguaje de programación se utilizó de lado cliente HTML 5 para el trabajo con las etiquetas, CSS 3 para el estilo de las páginas y para las validaciones JavaScript en su versión 2.0. De lado servidor se utilizó como lenguaje de programación web PHP en su versión 5.3.

1.11 IDE: Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado (IDE por sus siglas en inglés), es un entorno de programación empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI por sus siglas en inglés).Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación (Programación_Desarrollo, 2015).

1.11.1 IDE Netbeans 7.3

Entorno de desarrollo de código abierto integrado con PHP que posee un sistema que reconoce y carga clases, métodos y objetos, permitiendo de esta forma la creación de aplicaciones web. Propone un esqueleto para organizar el código fuente, es multiplataforma e integra lenguajes como HTML, JavaScript, CSS, y PHP. Brinda apoyo al marco de trabajo utilizado en el proyecto Symfony2, lo que permite dejar a un lado la consola de Symfony y centrarse en el desarrollo en el IDE, teniendo cargada todas las clases y ayuda. Se integra además con el control de versiones SubVersion, lo que es aprovechado por los desarrolladores para la realización de las tareas diarias con el repositorio del proyecto, lo cual reduce el tiempo de desarrollo y facilita el trabajo (Netbeans, 2015).

1.12 Sistema Gestor de Base de Datos

Un sistema de gestión de bases de datos (SGBD) es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos, además de proporcionar herramientas para añadir, borrar, modificar y analizar los datos. Estos sistemas

también proporcionan métodos para mantener la integridad y administrar el acceso de usuarios a los datos, también para recuperar la información si el sistema se corrompe. Como SGBD se decidió utilizar PostgreSQL para el proyecto SITPC.

1.12.1 PostgreSQL 9.3

PostgreSQL es un poderoso sistema gestor de base de datos relacional de código abierto. Cuenta con más de 15 años de desarrollo activo y una arquitectura probada que se ha ganado una sólida reputación de confiabilidad, integridad de datos y corrección. Funciona en todos los principales sistemas operativos, incluyendo Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), y Windows. Es compatible con ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), tiene soporte para claves foráneas, uniones, vistas, disparadores y procedimientos almacenados. También es compatible con el almacenamiento de objetos binarios, incluyendo imágenes, sonidos o vídeo. Tiene interfaces de lenguajes de programación como: C / C++, Java, Perl, Python, Ruby,entre otros y la documentación en varios idiomas (The PostgreSQL Global Development Group , 2015).

1.13 Servidor WEB

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 dichas 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.

1.13.1 Apache 2.2

Apache es uno de los servidores de aplicaciones web existentes a nivel mundial más utilizados. Algunas de sus principales características son:

- Corre en diferentes Sistemas Operativos (Windows, Linux).
- Es una tecnología gratuita de código abierto.
- Es un servidor altamente configurable de diseño modular.
- Permite personalizar la respuesta ante los posibles errores que puedan originarse en el servidor.
- Tiene una alta configurabilidad en la creación y gestión de logs (The Apache Software Foundation, 2015).

Como herramienta CASE se utilizó Visual Paradigm en su versión 8.0, como servidor de aplicaciones Apache en su versión 2.2 y para el desarrollo de la aplicación se utilizó el IDE Netbeans en su versión 7.3

1.14 Arquitectura de *software*

Es una pieza central del desarrollo de sistemas modernos. El objetivo de la arquitectura consiste en desarrollar sistemas grandes de forma eficiente, estructurada y con capacidad de reúso. La

arquitectura forma parte del proceso de diseño de *software* el cual también forma parte del proceso de desarrollo de *software* que comprende, requerimientos, diseño, implementación, prueba y mantenimiento (Mora, 2011).

1.14.1 Arquitectura por capas

El estilo arquitectural en capas se basa en una distribución jerárquica de los roles y las responsabilidades para proporcionar una división efectiva de los problemas a resolver. Los roles indican el tipo y la forma de la interacción con otras capas y las responsabilidades la funcionalidad que implementan. Las capas se ocupan de la división lógica de componentes y funcionalidad y no tienen en cuenta la localización física de componentes en diferentes servidores o en diferentes lugares (Henríquez, 2010).

1.14.2 Arquitectura Modelo Vista Controlador

El patrón MVC fue diseñado para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Sus características principales son que el Modelo, las Vistas y los Controladores se tratan como entidades separadas; esto hace que cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas. Este modelo de arquitectura presenta varias ventajas:

- Hay una clara separación entre los componentes de un programa; lo cual permite implementarlos por separado.
- Hay un Interfaz de Programación de Aplicaciones (API siglas en inglés) muy bien definido; cualquiera que use el API, podrá reemplazar el Modelo, la Vista o el Controlador, sin aparente dificultad.
- La conexión entre el Modelo y sus Vistas es dinámica; se produce en tiempo de ejecución, no en tiempo de compilación (Gutiérrez, 2012).

Para la solución del procedimiento de Casación en cuestión se asumió por la dirección del Proyecto la aplicación de la Arquitectura en capas que está basada en el patrón arquitectónico MVC. Este estilo aporta un diseño basado en niveles de abstracción crecientes, que facilita a los implementadores partir un problema complejo en una secuencia de pasos incrementales. Además admite fácilmente optimizaciones y refinamientos. Sin ignorar que aporta una importante reutilización de los componentes generados.

1.15 Patrones de Diseño

1.15.1 Patrones de caso de uso

Estos patrones se utilizaron como plantillas que describieron como deberían ser estructurados y organizados los casos de uso. Son patrones que capturan mejores prácticas para modelar casos

de uso. Para comprender mejor esta afirmación a continuación se describen los patrones que fueron utilizados:

- **Extensión:** Consiste en dos casos de uso y una relación extendida entre ellos. Puede ser instalado en sí mismo, así como extendido en el caso de uso base. El referente puede ser concreto o abstracto. Este patrón se aplica cuando un flujo puede extender el flujo de otro caso de uso así como ser realizado en sí mismo.
- **Inclusión:** Se incluye una relación del caso de uso base al caso de uso de inclusión. El último puede ser instalado en sí mismo. El caso de uso base puede ser concreto o abstracto.
- **Múltiples actores:** A un caso de uso ingresan más de dos actores y estos tienen un rol común (EcuRed_Patrones_de Casos_de_Uso).

1.15.2 Patrones de clases de diseño

Los patrones no se proponen descubrir ni expresar nuevos principios de la ingeniería del *software*. Todo lo contrario, intentan codificar el conocimiento, las expresiones y los principios ya existentes, cuanto más trillados y generalizados mejor. Dentro de los existentes, en la presente investigación se utilizarán:

- **Inyección de dependencia:**

El componente Inyección de dependencias, te permite estandarizar y centralizar la forma en que se construyen los objetos en la aplicación. En otras palabras consiste en pasar (*inyectar*) a las clases todos los objetos que necesitan (*dependencias*) ya creados y configurados (Eguiluz, 2011).

Patrones GRASP:

- **Experto:** En este patrón específicamente la asignación de la responsabilidad debe recaer sobre la clase que conoce toda la información necesaria para cumplir con la misma, como la creación de un objeto o la implementación de un método (Espino, 2011).
- **Creador:** Guía la asignación de responsabilidades relacionadas con la creación de objetos. Su intención básica es encontrar un creador que necesite conectarse al objeto formado en alguna situación (Larman, 1999).
- **Controlador:** Es un objeto que no pertenece a la interfaz de usuario, responsable de recibir o manejar un evento del sistema. Un Controlador define el método para la operación del sistema. (Larman, 1999).
- **Bajo acoplamiento:** Patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño. Permite asignar una responsabilidad de modo que su colocación no amplíe tanto el acoplamiento que produzca los resultados adversos de un alto acoplamiento. Tolera además el

diseño de clases independientes, reduciendo el impacto de los cambios y favoreciendo la reutilización (Larman, 1999).

- **Alta cohesión:** El patrón Alta Cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.

Una clase con baja cohesión realiza un trabajo excesivo y no conviene este tipo de clases pues presentan los siguientes problemas:

- Son difíciles de comprender.
- Son difíciles de reutilizar.
- Son difíciles de conservar.
- Son delicadas: las afectan constantemente los cambios (Laman, 1999).

Patrones GoF:

- **Patrón decorador:** Extiende la funcionalidad de un objeto dinámicamente de tal modo que es transparente a sus clientes, utilizando una instancia de una subclase de la clase original que delega las operaciones al objeto original. Provee una alternativa muy flexible para agregar funcionalidad a una clase (Aleaga, 2013).

1.15.3 Patrones de base de dato

Para el diseño del modelo de datos se utilizan un conjunto de patrones muy útiles para organizar estructuralmente la base de datos de la aplicación donde se almacena la información. A continuación se muestran los utilizados para esta investigación:

- **Llaves subrogadas:** Con el empleo de este patrón se genera una llave primaria única para cada entidad en vez de usar un atributo identificador en el contexto dado.
- **Árbol Simple:** Este patrón normalmente utilizado cuando el árbol es la representación de una estructura de datos. Los elementos a almacenar son del mismo tipo, es decir pueden ser almacenados en la misma entidad. No pueden existir ciclos, es decir, un hijo no puede ser su propio padre (EcuRed_Patrones_de_BD).

1.16 Conclusiones parciales

En este capítulo se presentaron los conocimientos teóricos fundamentales para la solución del problema planteado y se analizaron los principios que rigen el funcionamiento del procedimiento de Casación, por lo que se puede concluir que:

- A partir del estudio del estado del arte se comprendió que 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, por lo que es necesario implementar un sistema que responda a las necesidades de los mismos.

- Se realizó el análisis de las tecnologías informáticas previamente definidas por el proyecto SITPC permitiendo la selección de aquellas que mejor se ajustan al sistema que se desea implementar.

CAPÍTULO II PROPUESTA DE SOLUCIÓN

2.1 Introducción

En el presente capítulo se describe la solución técnica de la investigación, donde se analiza y se especifica la arquitectura sobre la cual se desarrollará el sistema, las diferentes capas que la componen y cada uno de sus componentes. A partir de los requisitos identificados en el recurso de Casación se presenta el modelo de negocio para una mejor comprensión del mismo. Por otra parte se ilustrará el modelo de diseño conformado por los diagramas de clases y de secuencia, el modelo de implementación con los diagramas de componentes y de despliegue. Finalmente se muestra el modelo de datos con el diagrama entidad-relación y para la implementación del sistema se describen los estándares de codificación.

2.2 Modelado del negocio

En el modelo de negocio se definen una serie de procedimientos de varios aspectos específicos del procedimiento de Casación, de forma tal que se pueda entender mejor su funcionalidad, su lógica del negocio y como está estructurado. A continuación se muestra la descripción de los actores vinculados al negocio.

2.2.1 Actores del negocio

Actor	Descripción
Abogado	Actor generalizador de la Secretaria encargado de registrar los escritos de interposición del recurso, sosteniendo el recurso, de personería de la parte no recurrente y solicitando ejecución de sentencia.
Secretaria	Actor generalizador de la Secretaria TPP y Secretaria TSP encargado de dar entrada al expediente.
Secretaria TPP	Actor encargado de notificar emplazamiento y registrar resulta de emplazamiento.
Secretaria TSP	Actor encargado de entregar papel de fianza y registrar resulta de pago de fianza.
Juez ponente TPP	Actor encargado de disponer sobre el escrito de interposición del recurso de casación, ordenar elevar las actuaciones al TSP.
Juez ponente TSP	Actor encargado de disponer fianza y ordenar conformar piezas separadas y ordenar la ejecución de sentencia en el TPP.

Tabla 1 Actores del negocio

2.2.2 Proceso del negocio

En la siguiente tabla se describe cómo va a estar estructurado el diagrama de proceso de negocio de Registrar Escrito de Interposición de Recurso.

Objetivo	Interponer el recurso de casación en el TPP para que se admita y eleven las actuaciones para su tramitación en el TSP.
Evento(s) que lo genera(n)	Presentación del escrito de interposición del recurso ante el TPP.
Pre condiciones	Que se haya dictado sentencia o auto definitivo en el TPP.
Marco legal	Ley de Procedimiento Civil, Administrativo, Laboral y Económico.
Reglas de negocio	Ver documento Reglas del negocio en el Expediente del proyecto.
Responsable	Secretaria, Presidente de sala, Juez ponente del TPP, Parte.
Clientes internos	TSP, TPP.
Clientes externos	Parte recurrente.
Entradas	Escrito de interposición del recurso.
Flujo básico	Presentar escrito de interposición del recurso. La parte recurrente presenta en el TPP el escrito de interposición del recurso en los 5 días luego de notificada la sentencia y aporta el convenio jurídico. Registrar escrito de interposición del recurso. La secretaria del TPP recibe el escrito, lo registra y lo pasa al juez ponente para que disponga sobre él. Disponer sobre el escrito de interposición del recurso. El juez ponente del TPP recibe el escrito de interposición del recurso y lo revisa para verificar que cumpla con las formalidades requeridas, si no cumple ordena subsanarlo (ver flujo alterno 3.a); si fue entregado en el término lo admite sino lo rechaza (ver flujo alterno 3.b). Admitir escrito. El juez ponente del TPP crea la providencia de admisión y ordena emplazar por 10 días a las partes. Notificar emplazamiento. La secretaria del TPP es la encargada de notificar el emplazamiento a todas las partes que intervienen en el

	proceso. Registrar resulta. La secretaria del TPP registra la resulta del emplazamiento. Ordenar elevar las actuaciones. Luego de haber sido emplazadas todas las partes el juez ponente del TPP crea la providencia ordenando elevar las actuaciones al TSP. Enviar expediente al TSP. La secretaria del TPP dispone de 5 días para enviar el expediente. Se crea el oficio de elevación donde se consigna el expediente a enviar con sus respectivos expedientes gubernativos. Recibir escritos de personería de las partes. Las partes se personan en el TSP en el término de emplazamiento y la secretaria registra los escritos en el libro de presentación de escrito (LPE) y los archiva temporalmente hasta que se reciba el expediente. La parte no recurrente puede solicitar la ejecución del fallo de instancia (ver flujo alterno 9.a). Recibir expediente del TPP. La secretaria del TSP firma el oficio de elevación creado en el TPP y recibe el expediente enviado. Conformar rollo. La secretaria del TSP crea el rollo donde incluye los escritos recibidos y el expediente. Radicar rollo. La secretaria del TSP radica el rollo con el número consecutivo que le corresponde y lo entrega al presidente de la sala. Turnar. El presidente de la sala de lo Civil y Administrativo del TSP es el encargado de designar al juez ponente para la tramitación judicial mediante la providencia de turnado. Luego la pasa a la secretaria para que esta la firme y la una al rollo. Entregar rollo al juez ponente. La secretaria del TSP es la encargada de entregarle el rollo al mencionado juez.

Tabla 2 Descripción del proceso Interposición de recurso

2.2.3 Diagrama de proceso de negocio

En la siguiente figura se mostrará el flujo de actividades que se llevan a cabo en el proceso de Interposición de recurso:

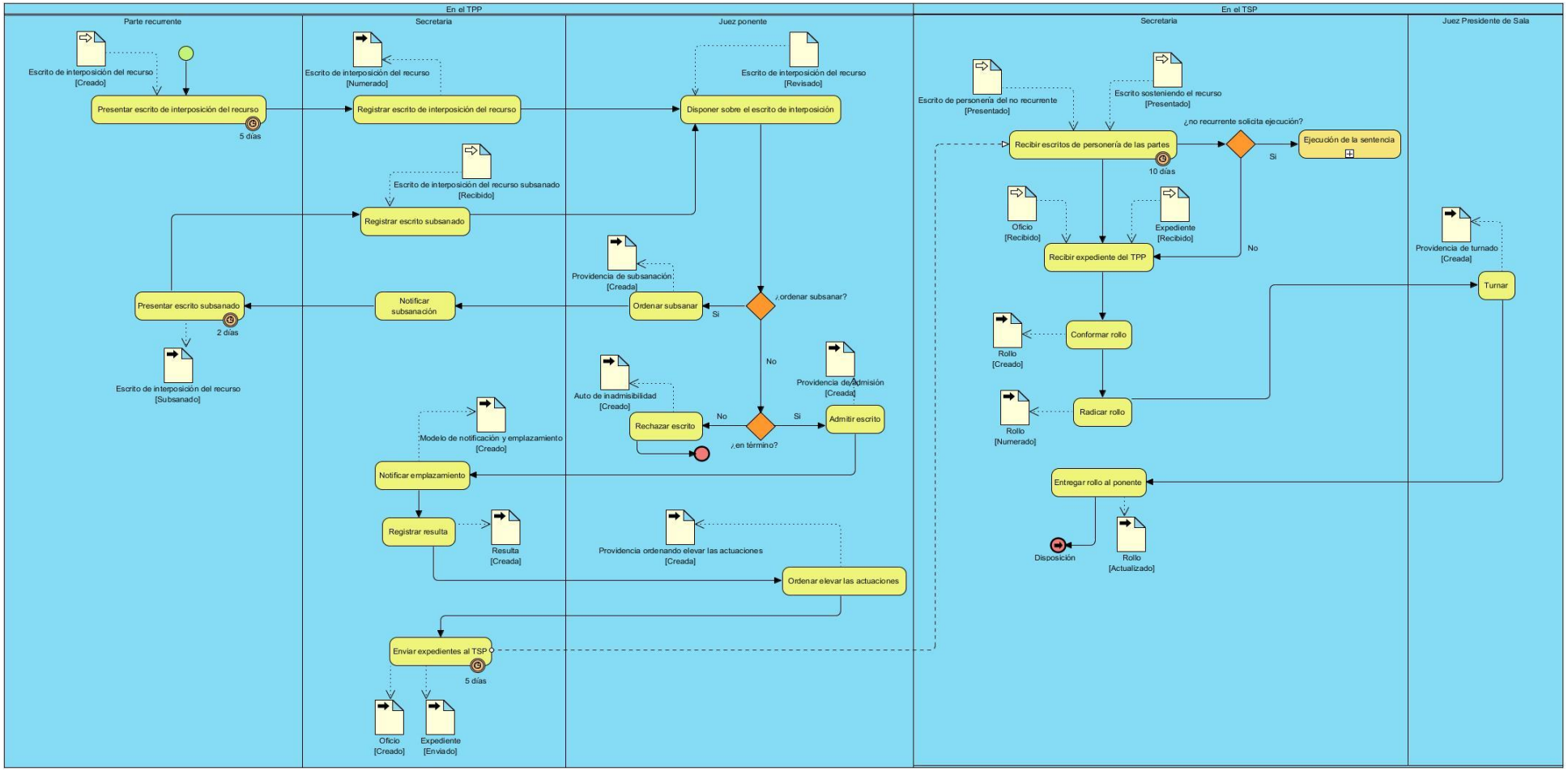


Figura 1 Diagrama de proceso de negocio

2.3 Propuesta del sistema

Al realizar varios encuentros de trabajo con el cliente se obtuvieron como resultado las funcionalidades que debe cumplir el sistema, como por ejemplo los requisitos funcionales y los no funcionales de la aplicación a desarrollar, así como los casos de uso. A continuación se mostrará con más detalles cada uno de estos aspectos.

2.3.1 Técnicas de captura de requisitos

Entrevistas

En las entrevistas, se realizaron preguntas a las partes interesadas sobre el sistema a desarrollar. Las entrevistas se hicieron abiertas ya que de esta forma no existe un programa definido y se fueron creando preguntas en dependencia de las respuestas aportadas por el cliente. Se examinó una serie de cuestiones con los clientes del sistema y, por tanto, se desarrolló una mejor comprensión de sus necesidades.

Arqueología de documentos

Con la aplicación de esta técnica se tratan de determinar posibles requisitos sobre la base de inspeccionar la documentación utilizada por la empresa. Esta herramienta sirve más que nada como complemento de las demás, y nos ayuda a obtener información que de otra manera sería sumamente difícil conseguir (Dávila, 2001).

Tormenta de ideas

Esta técnica se utilizó efectuando reuniones en grupo donde se generaron ideas en un ambiente libre de críticas o juicios. Ayudó a generar una gran variedad de vistas del problema y a formularlo de diferentes formas, sobre todo al comienzo del proceso de captura, cuando los requisitos son todavía muy difusos.

De esta forma utilizando cada una de las técnicas mencionadas se tuvo como resultado los siguientes requisitos funcionales:

2.3.2 Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares.

Al haber definido el ámbito y la naturaleza del problema a resolver, se aplicaron las técnicas de captura de los requisitos como tormentas de ideas, arqueología de documentos y entrevistas, los cuáles dieron como resultado los requisitos funcionales del sistema. De ellos se muestra algunos relacionados con el escrito de interposición de recurso:

RF01 Adicionar motivos de interposición del recurso de Casación en el TPP.

RF02 Eliminar motivos de interposición del recurso de Casación en el TPP.

RF04 Listar partes del proceso.

RF06 Adicionar técnico auxiliar.

Para una información más ampliada sobre dichos requisitos ver el documento CEGEL-SITPC-Especificacion_de_requisitos_de_software.

2.4 Diagrama de caso de uso

Mediante el uso de los patrones de inclusión, exclusión y actores múltiples se obtuvo el diagrama que se muestra a continuación con un total de 16 CU con sus respectivos actores:

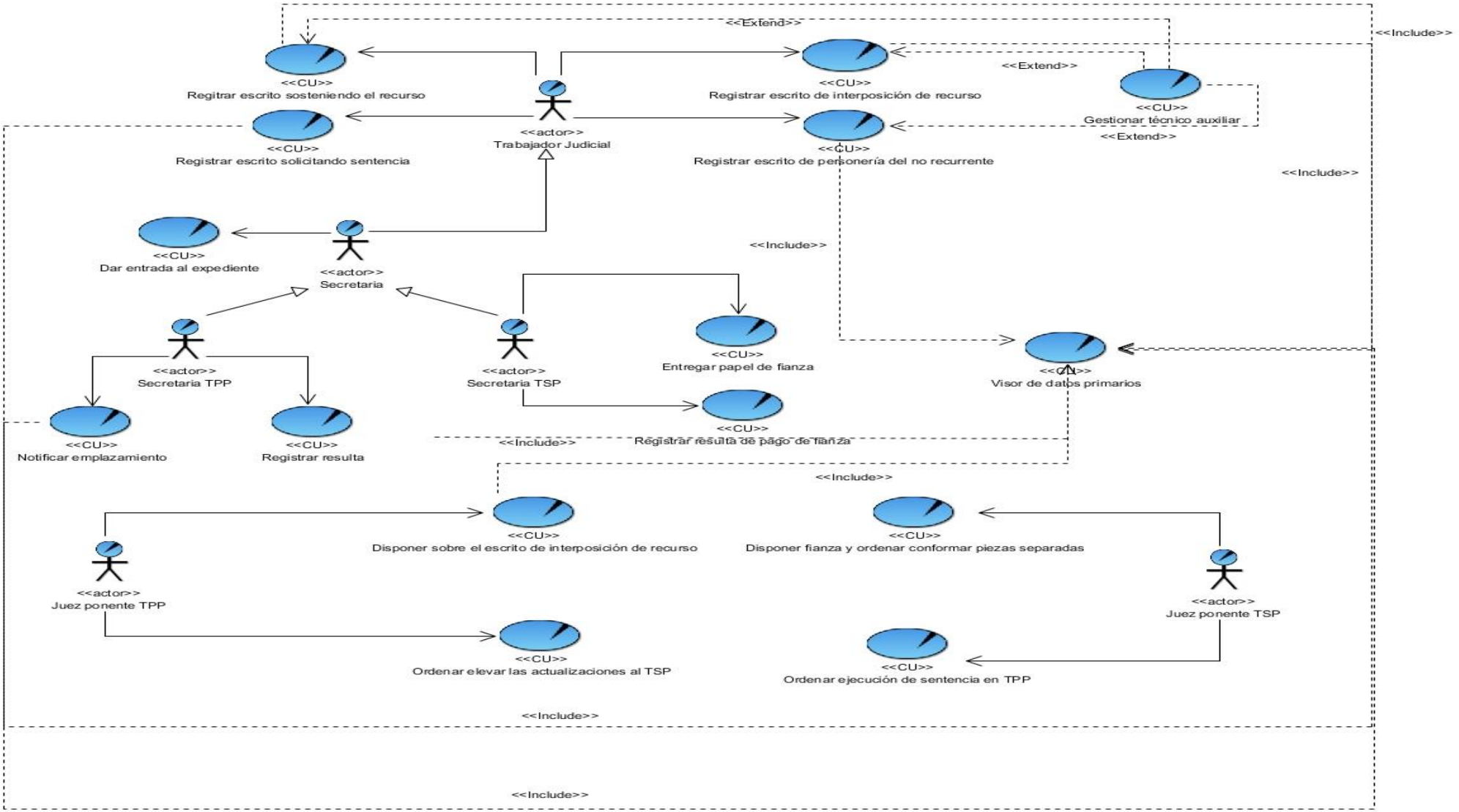


Figura 2 Diagrama de casos de uso

2.4.1 Especificación del caso de uso Registrar escrito de interposición de recurso

Objetivo	Registrar escrito
Actores	Secretaria, Abogado, Fiscal.
Resumen	El CU se inicia cuando se presenta el escrito de interposición del recurso. La parte recurrente presenta en el TPP el escrito de interposición del recurso en los 5 días luego de notificada la sentencia y aporta el convenio jurídico.
Complejidad	Alta.
Prioridad	Crítica.
Precondiciones	Que se haya dictado sentencia o auto definitivo en el TPP.
Postcondiciones	• Queda presentado el escrito de interposición del recurso y los documentos que lo acompañan. • Se actualiza el LPE.
Flujo de eventos	
Flujo básico Registrar recurso de Casación subsanado.	

	Actor	Sistema
1.	Accede en el menú izquierdo a la opción “Escrito”.	
2.		Solicita los números de LPE y el año.
3.	Introduce los datos necesarios.	
4.		Muestra los datos primarios del LPE (invoca al caso de uso Visualizar datos primarios) y muestra las siguientes opciones: • Presentado por. • Tipo de escrito. Documentos que acompañan.
5.	Introduce los datos y el tipo de escrito De interposición del recurso de casación e introduce los motivos por los cuales va a registrar el escrito.	
6.		En el caso que sea la secretaria el sistema muestra 2 opciones:

		• Registrar • Cancelar En el caso de que sea el abogado el sistema muestra 2 opciones> • Vista previa • Cancelar
7.	Seleccionar la opción “cancelar”	
8.		Tanto para la secretaria como para el abogado el sistema cancela toda la operación y muestra la pantalla de inicio.
9.	Selecciona la opción “Registrar”	
10.		Valida que los datos introducidos son correctos, guarda los cambios y muestra la pantalla de Inicio.
11.	Selecciona la opción “Vista previa”	
12.		Valida que los datos introducidos son correctos y guarda los cambios. Muestra el visor con las opciones: • Pasar a definitivo. • Guardar. • Cancelar.
13.	Selecciona la opción “Cancelar”	
14.		Cancela la operación y muestra nuevamente la página de Registrar escrito.
15.	Selecciona la opción “Guardar”	
16.		En el caso de que el abogado desee modificar algo en el visor el sistema lo guarda y muestra la pantalla de inicio.
17.	Selecciona la opción “Pasar a definitivo”	

18.		Muestra la pantalla de Inicio. Termina el CU.

Tabla 3 Especificación del caso de uso Registrar escrito

2.5 Requisitos no funcionales

Los requisitos no funcionales o atributos de calidad son los que especifican criterios que pueden emplearse para juzgar la operación de un sistema en lugar de sus comportamientos específicos.

El equipo de arquitectura del proyecto definió los requisitos no funcionales, algunos de ellos son los siguientes:

- **Usabilidad:** Mostrar los mensajes, títulos y demás textos que aparezcan en la interfaz del sistema en idioma español.
- **Fiabilidad:** Manejo de excepciones por el sistema. Las excepciones que se produzcan, podrán ser tratadas mediante una llamada a un gestor de excepciones que informará al usuario del error a través de la interfaz gráfica y se registrará en un fichero de logs.
- **Eficiencia:** Rendimiento del sistema. El sistema deberá permitir como mínimo 40 transacciones simultáneas.
- **Seguridad:** Definir una jerarquía de usuarios para el manejo centralizado de los mismos en el sistema.
- **Portabilidad:** El sistema debe ser una aplicación web.
- **Reusabilidad:** Garantizar que los formatos de los archivos de salida del sistema sean compatibles con los programas más comunes.

Para auxiliarse de esta información ver documento: “CEGEL-SITPC Especificación de requisitos no funcionales” del proyecto SITPC.

2.6 Arquitectura del Sistema

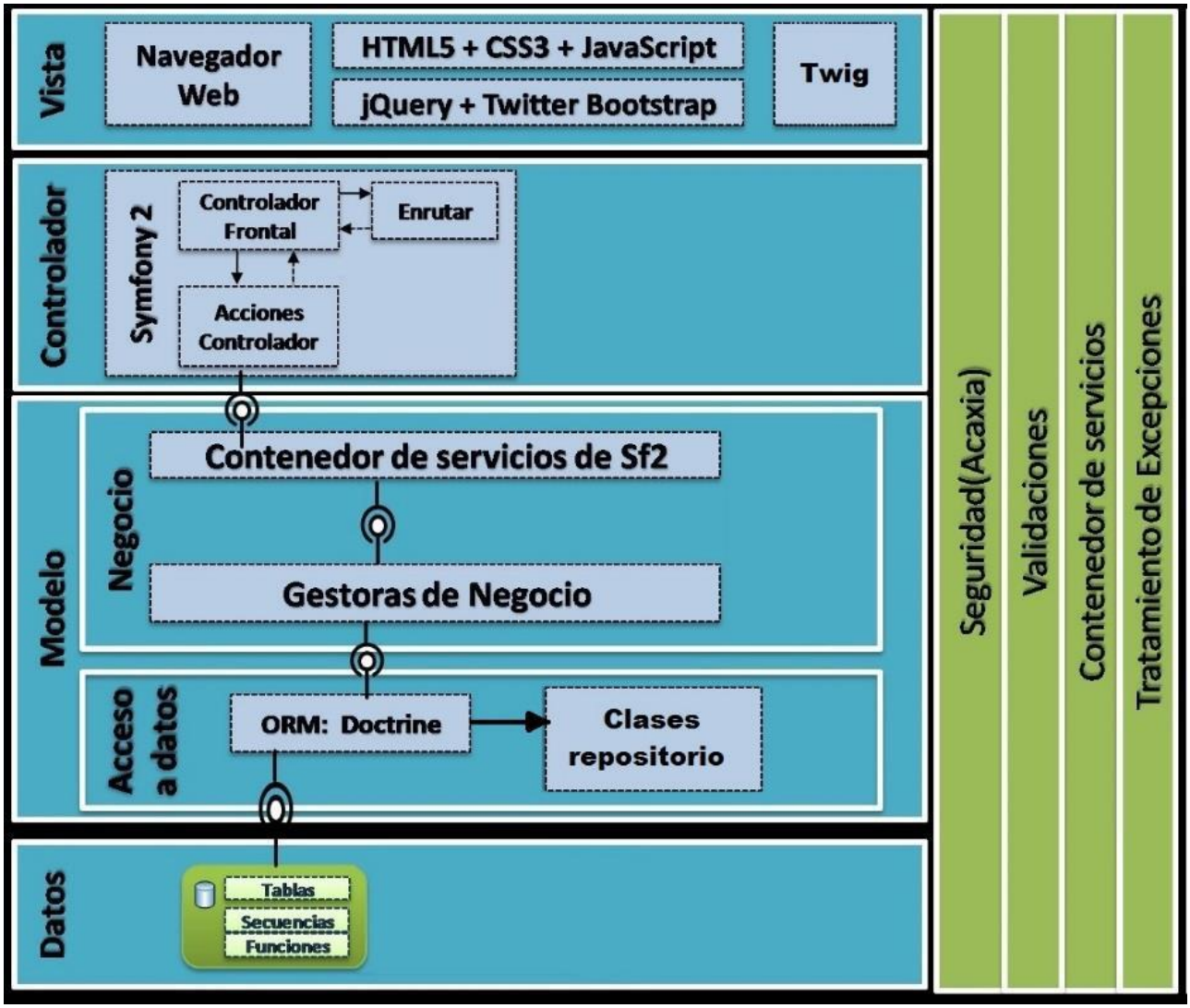


Figura 3 Arquitectura del sistema

El SITPC definió el estilo arquitectónico MVC y la arquitectura por capa ya que de esta forma cada una de ellas solo puede hacer referencia a componentes en capas inmediatamente inferiores, además reduce las dependencias ya que cada nivel no es consciente de ningún detalle o interfaz de los superiores. Cada capa juega un papel muy importante dentro de la arquitectura, como por ejemplo la vista o como también se le conoce la capa de presentación que es donde se diseña la interfaz por la cual el usuario va a interactuar con el sistema. Por otra parte está el controlador que es el responsable de recibir los eventos de entrada desde la vista utilizando el enrutador como intermediario. En el modelo está la lógica del negocio la cual utiliza los gestores para acceder a los datos e interactúa con la capa de presentación para recibir solicitudes y presentar los resultados y también interactúa con la capa de datos solicitándole al manejador de los datos que ejecute una operación de adicionar, modificar, eliminar o simplemente una consulta a los datos.

En la siguiente imagen está representado el patrón arquitectónico MVC mediante una serie de carpetas que se generan cuando se crea un proyecto de Symfony:

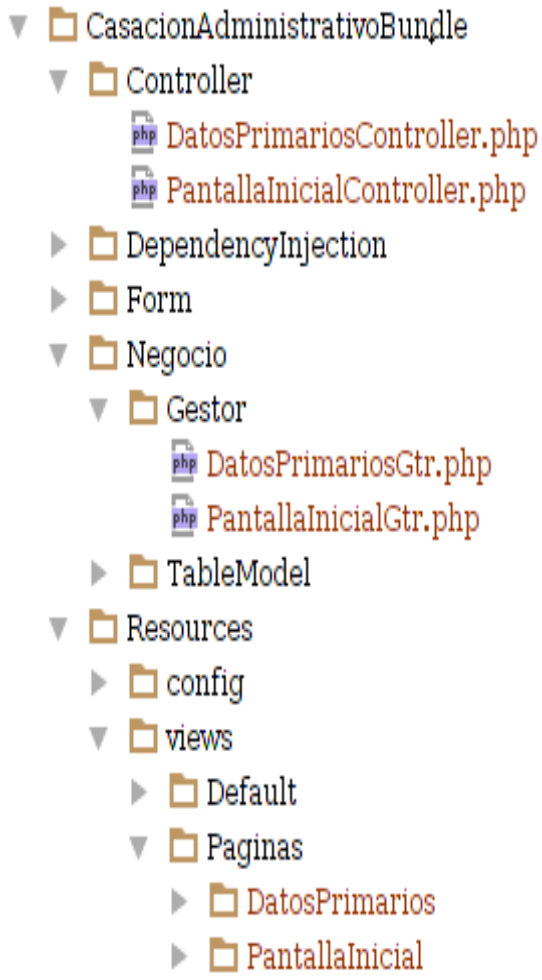


Figura 4 Carpetas generadas por Symfony

Componentes verticales:

- **Seguridad:** El sistema Acaxia es el que se encarga de la seguridad en la aplicación, es el que garantiza la autenticación, autorización, auditoría, administración de perfil y administración de conexiones. La autenticación garantiza la fortaleza de la contraseña estableciendo parámetros a cumplir por la misma, como longitud (más de 8 caracteres), vencimiento de la contraseña cada 90 días, tenencia de mayúscula, números y caracteres especiales. La autorización, permite gestionar los permisos para cada uno de los usuarios que accedan a la aplicación, teniendo en cuenta el principio de mínimo privilegio. Para realizar la auditoría se hace uso del control de trazas, permitiendo saber que usuario se autenticó en el sistema, el día y el tiempo que estuvo conectado, así como las acciones realizadas por el mismo. La administración de perfil permite al usuario la personalización de su perfil en el sistema, posibilitándole modificar datos personales, así como el cambio de su contraseña cada cierto tiempo.

- **Validaciones:** Las validaciones se realizan tanto del lado del cliente como del lado del servidor, para asegurar la entrada de valores correctos a la aplicación, garantizando un correcto funcionamiento de la misma.
- **Tratamiento de excepciones:** El tratamiento de excepciones lo brinda Symfony2 de forma interna, se encuentra de forma transversal en toda la aplicación pues en cualquiera de las capas se puede lanzar una excepción. En la arquitectura propia del SITPC lo que se realizó fue adaptar la plantilla de error por defecto a mostrar.

2.7 Modelo de Diseño

El modelo de diseño sirve como abstracción del modelo de implementación y su código fuente, siendo utilizado como entrada fundamental de las actividades de implementación, está integrado por los diagramas de clase del diseño y por los diagramas de interacción.

2.7.1 Diagrama de clases de diseño

El diagrama de clases se realizó para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de agregación.

En la siguiente figura se muestra el diseño de las clases del caso de uso principal Registrar escrito de interposición de recurso:

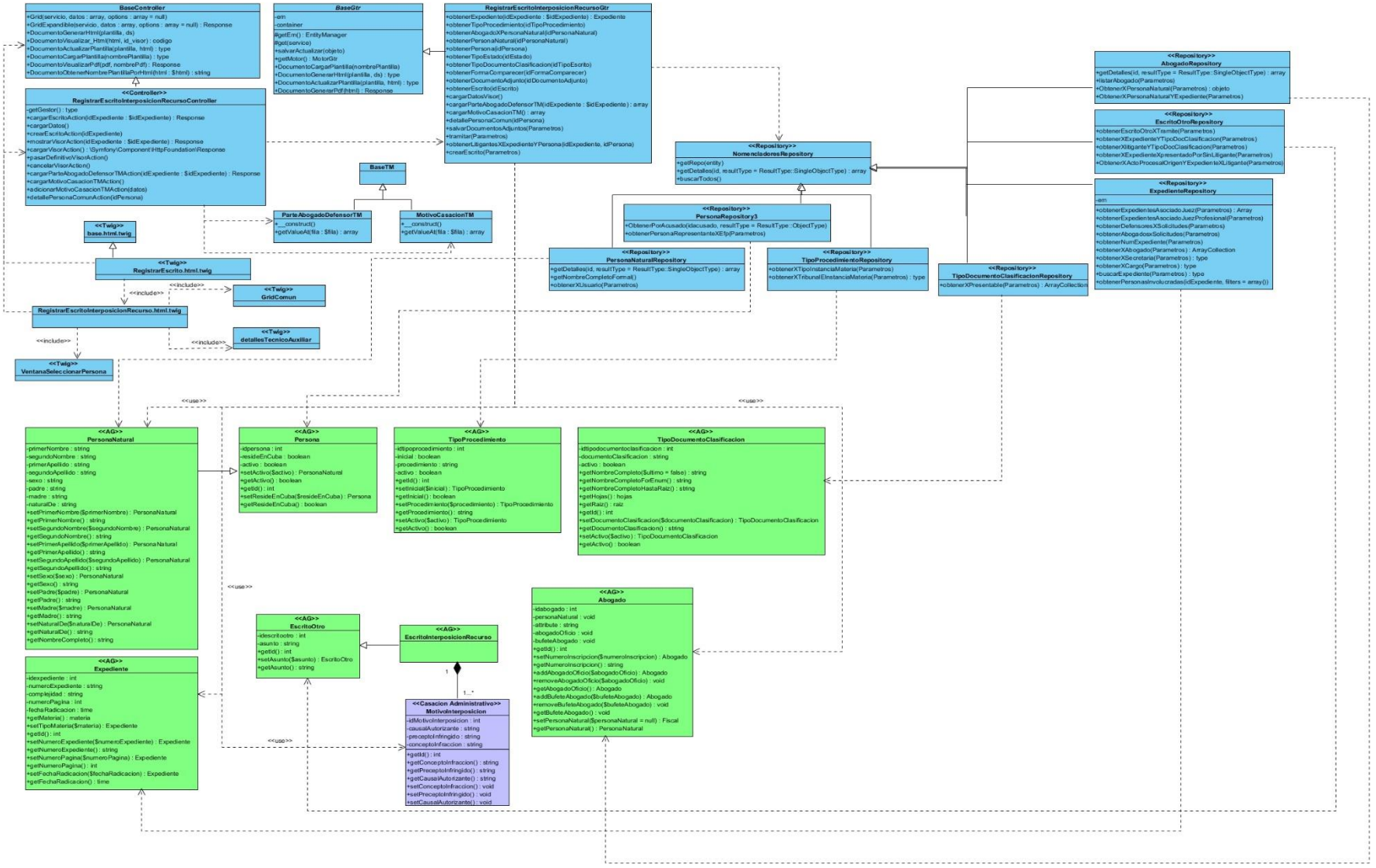


Figura 5 Diagrama de clases de diseño

2.7.2 Diagrama de secuencia

El diagrama de secuencia muestra una interacción ordenada según la secuencia temporal de eventos. En particular, muestra los objetos participantes en la interacción y los mensajes que intercambian ordenados según su secuencia en el tiempo.

A continuación se muestra el diagrama de secuencia donde se describe cómo se comporta de forma general el código de implementación del caso de uso Registrar escrito de interposición de recurso:

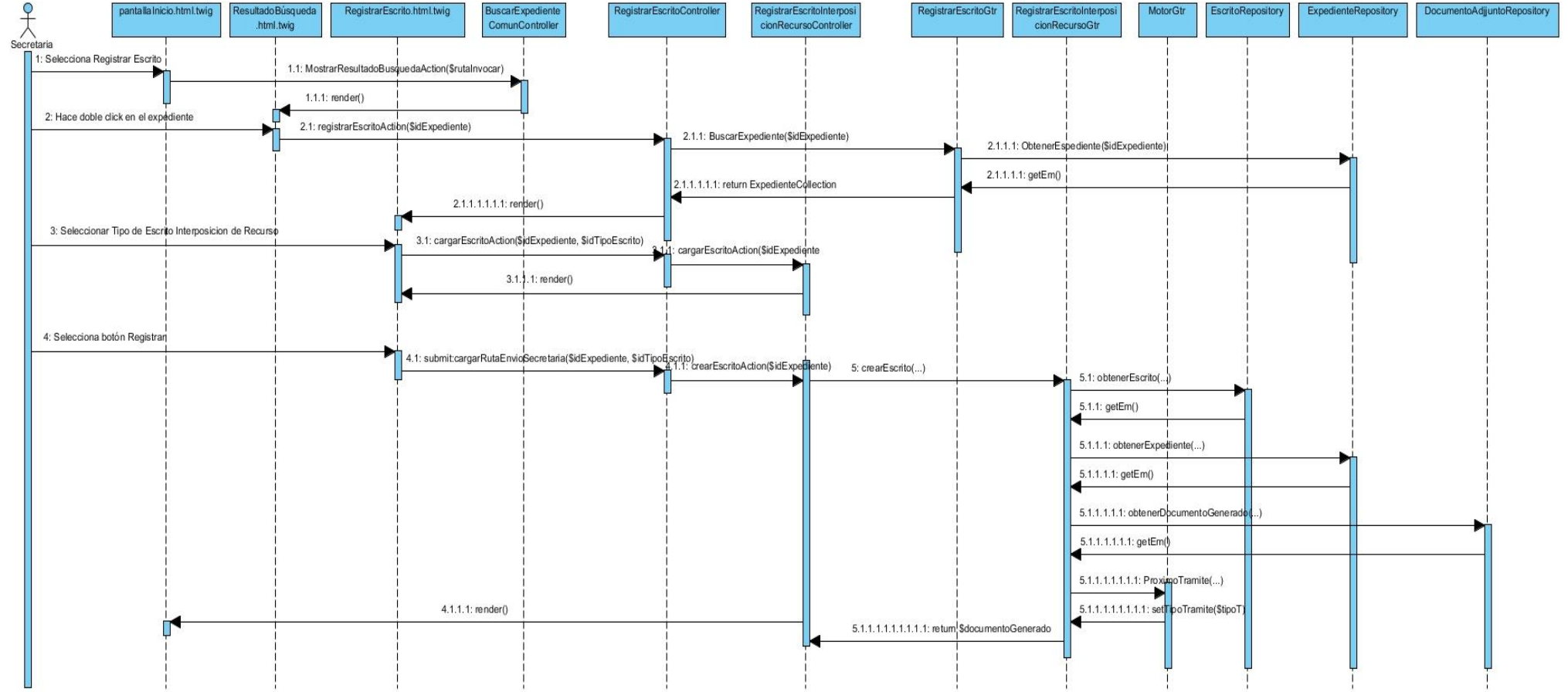


Figura 6 Diagrama de secuencia del caso de uso Registrar escrito de interposición de recurso

2.8 Modelo de Implementación

El modelo de implementación muestra las dependencias entre las partes de código del sistema y la estructura del sistema en ejecución, describe cómo los elementos del diseño y las clases se implementan en términos de componentes o código fuente.

2.8.1 Diagrama de componente

En este diagrama se muestra el conjunto de componentes y sus relaciones de manera gráfica mediante el empleo de nodos y arcos.

A continuación se ilustrará una serie de componentes por los cuáles está compuesto el caso de uso Registrar escrito de interposición de recurso:

Capa Vista: Contiene el paquete de las vistas que se muestran al usuario, estas se ubican en CasacionAdministrativoBundle\Resources\views\Paginas. También se encuentran los estilos CSS de Bootstrap utilizados para el diseño de las páginas y para las validaciones están los componentes de JavaScript de Bootstrap para las validaciones.

Capa Controladora: Agrupa el controlador frontal y el paquete de las clases controladoras del *bundle* Casación de Administrativo donde se encuentran las acciones del caso de uso principal Registrar escrito de interposición de recurso.

Capa Modelo: Agrupa las clases gestoras en el paquete ubicado en CasacionAdministrativoBundle\Negocio\Gestor, estas establecen la comunicación con las clases del paquete Acceso a datos, mediante el cual accede a los datos almacenados en la base de datos.

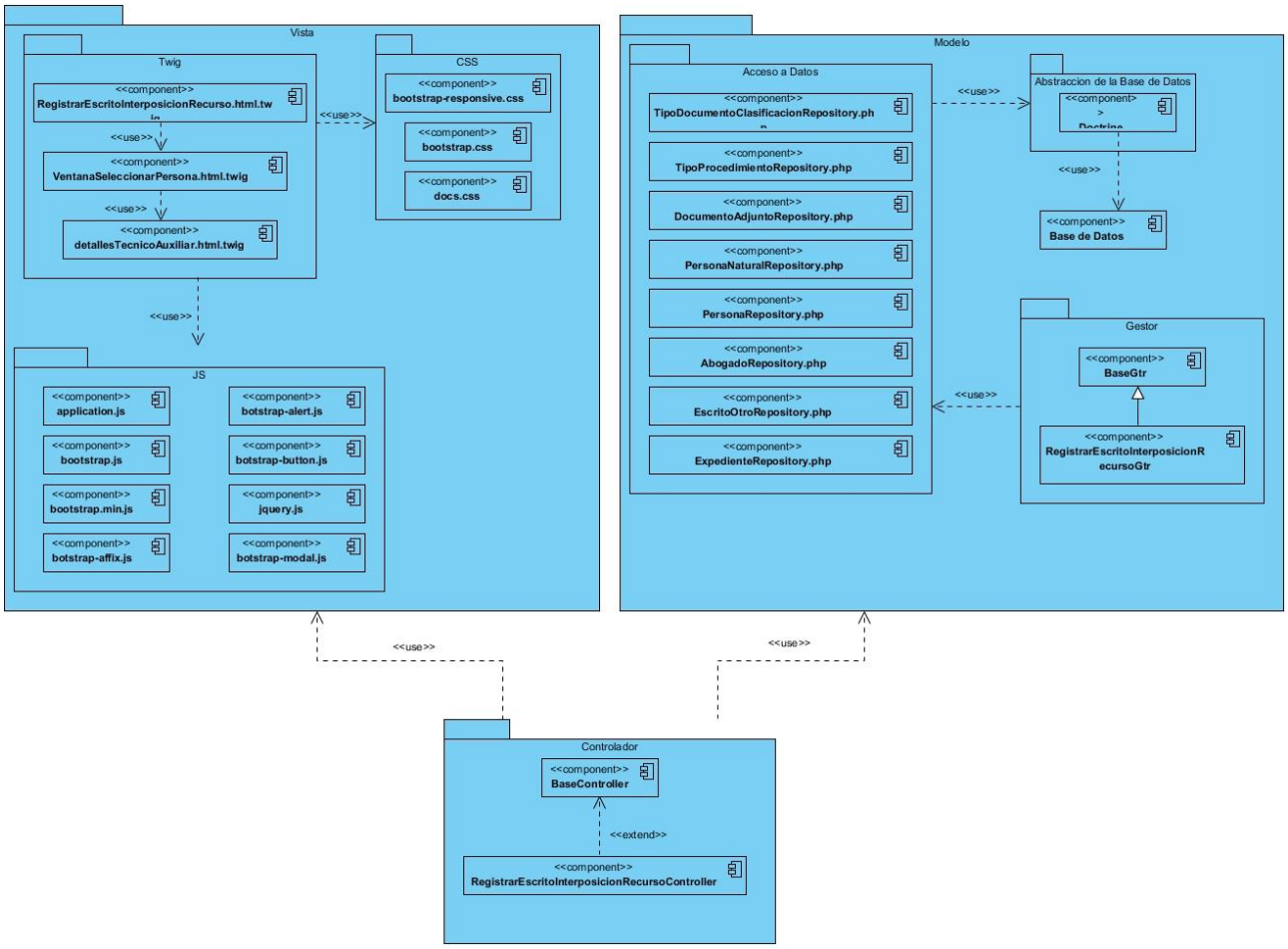


Figura 7 Diagrama de componente

2.8.2 Diagrama de despliegue

Este tipo de diagrama del Lenguaje Unificado de Modelado se realizó con el objetivo de mostrar las relaciones físicas de los distintos nodos que componen un sistema y la distribución de los componentes sobre dichos nodos. Se describió la topología del sistema y la estructura de los elementos de *hardware* y el *software* que ejecuta cada uno de ellos.

Mediante un análisis exhaustivo se definió como va a estar estructurado el sistema en los TPP, a continuación se muestra el resultado mediante una figura**:**

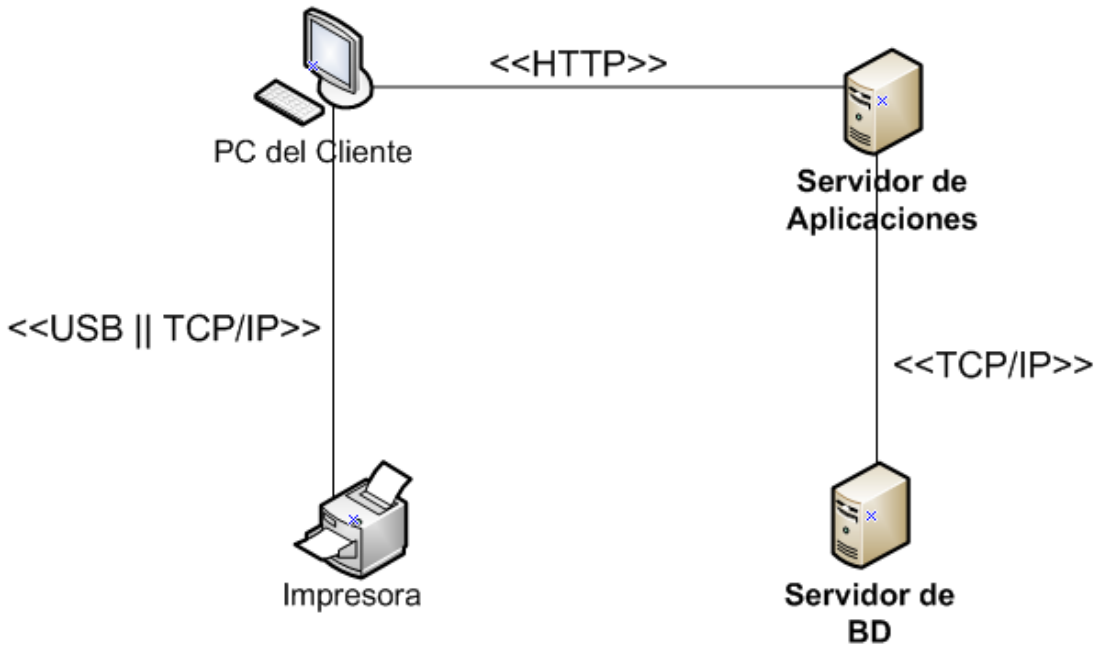


Figura 8 Diagrama de despliegue

2.9 Modelo de datos

Para el diseño del diagrama que se muestra a continuación se utilizaron los patrones llaves subrogadas y árbol simple, obteniéndose como resultado un total de 62 entidades de las cuales solo 9 almacenan información del caso de uso principal Registrar escrito de interposición de recurso. La presente base de datos se encuentra normalizada en primera forma normal ya que cada tupla contiene exactamente un valor de las tablas de la base de datos, o sea, no existen campos multivaluados, compuestos ni sus combinaciones. También se normalizaron en 2FN, pues se encuentran en 1FN y todos los atributos que no son claves en las tablas, dependen totalmente de la clave primaria que le corresponde. Finalmente se llevó a la 3FN, por encontrarse en 2FN y no tener dependencias transitivas en los atributos no primos de las entidades.

2.9.1 Diagrama entidad-relación

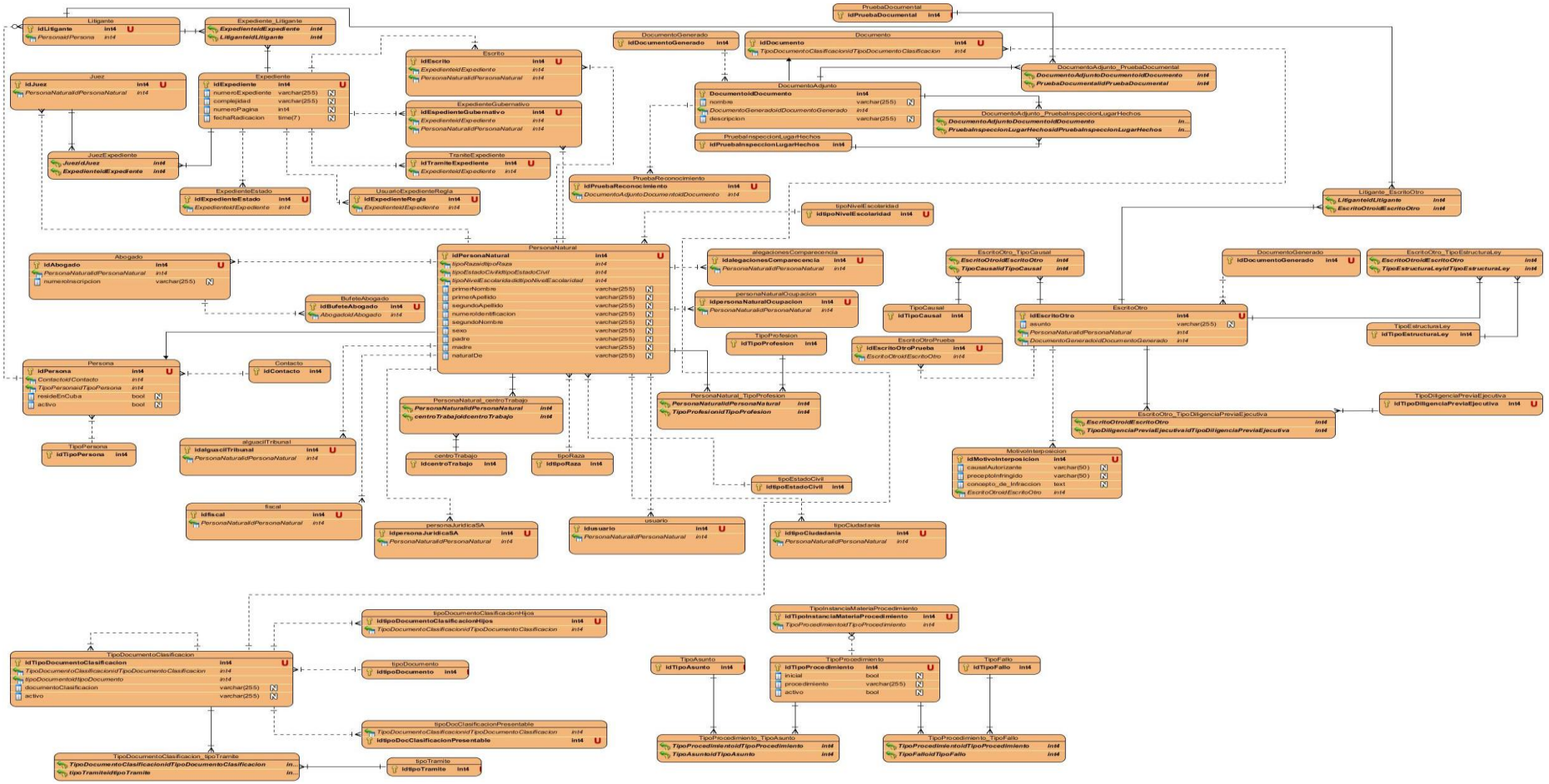


Figura 9 Diagrama entidad-relación

2.10 Estándares de Codificación

El estándar de codificación completo comprende todos los aspectos de la generación de código por lo tanto su uso fue de gran ayuda ya que permitió emplear correctamente los estándares de código a la hora de empezar a programar. En el caso particular del SITPC el estándar de codificación fue definido en sus inicios por la dirección del proyecto, creándose de esta forma el documento CEGEL_SITPC_Estándares de Codificación v1.0, por el cuál a continuación se mostrará el formato de los siguientes puntos:

- **Identación.**

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

✓ 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, entre otras cosas. Es de cada equipo decidir si se desea o no agregar más datos.

/**

*

* @Descripción de PantallaInicialGtr

* @modificado: 10 marzo del 2015

* @autor: Leosbel Poll

*

*/

- **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.

```php
/**
 * Este método me devuelve el gestor asociado al CU registrar escrito de interposicion del recurso de casación.
 * @return type
 * @throws \LogicException
 */
private function getGestor() {
    if (!$this->container->has('administrativo.cas.gtr.escritoInterposicionRecurso')) {
        throw new \LogicException('Este servicio no esta registrado en la aplicacion');
    }

    return $this->container->get('administrativo.cas.gtr.escritoInterposicionRecurso');
}
```

Figura 10 Ejemplo de un comentario

- **Clases**

Las clases serán colocadas en un archivo .php aparte, donde sólo se colocará el código de la misma. El nombre del archivo será el mismo del de la clase. De cierta forma ellas siguen las mismas reglas de las funciones, por tanto, debe colocarse un comentario antes de su declaración explicando su utilidad. Sus nombres deben iniciarse con letra mayúscula. Si su nombre tiene más de una palabra, la primera letra de cada nueva palabra debe capitalizarse. No se permiten las letras capitalizadas sucesivas; por ejemplo, "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.

- **Nombres de 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.

- **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. Ejemplo:

```
listarUsuariosAction() {
        instrucciones
}
```

2.11 Conclusiones Parciales

En el presente capítulo se mostraron todos los elementos relacionados con la arquitectura definida, los patrones de diseño y los componentes que se utilizaron en la implementación, para la organización del código se definieron los estándares de codificación del proyecto SITPC y además se generaron todos los artefactos en cada modelo de la arquitectura, dando como resultado:

- Al elaborar el modelo de diseño posibilitó tener una representación técnica del sistema que se implementó.
- Con la elaborar del modelo de implementación permitió generar los elementos necesarios para comprender mejor el proceso de implementación, la estrategia concebida para el despliegue en la instancia provincial de los tribunales.

CAPÍTULO III Análisis de los Resultados

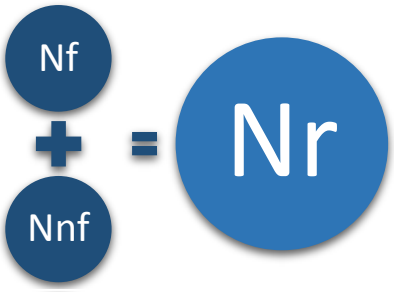
3.1 Introducción

En este capítulo se evalúa la calidad y la fiabilidad de los resultados obtenidos con el desarrollo de este trabajo, utilizando las métricas para validar la calidad del diseño y la implementación del *software*. Este proceso es de vital importancia ya que cuando los desarrolladores programan distintos tipos de funcionalidades no son conscientes de que en algún momento los resultados esperados pueden variar durante la ejecución del código. Para evitar que esto suceda se realizan un conjunto de pruebas y validaciones a los diferentes componentes garantizando el correcto funcionamiento y totalidad de las funcionalidades de los mismos.

3.2 Validaciones

3.2.1 Validación de la especificación de los Requisitos

La validación de los requisitos examina la especificación para asegurar que todos los requisitos de software se han establecidos de manera precisa; que se han detectado las inconsistencias, omisiones, errores y estos han sido corregidos. Este proceso se lleva a cabo realizando una operación donde se suman la cantidad de requisitos funcionales (Nf) más los requisitos no funcionales (Nnf), obteniéndose el número de requisitos existentes en el sistema (Nr).

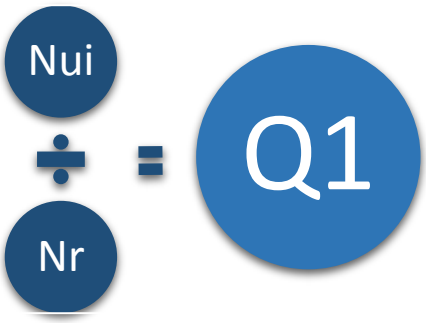


CTR=RF + RNF

CTR=60 + 45

CTR=105

Para determinar la **especificidad** (ausencia de ambigüedad) de los requisitos se sugiere una métrica basada en la consistencia de la interpretación de los revisores para cada requisito:



Q1=Nui/ Nr

Q1=105 / 105

Q1=1

Donde Nui es el número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas. Cuando más cerca de 1 este el valor de Q1 menor será la ambigüedad de la especificación, y como el resultado es 1 el nivel de ambigüedad es cero por lo que no se encontraron errores en el contenido o en la interpretación, como tampoco requisitos irreales.

3.2.2 Validación de requisitos mediante prototipos

Esta métrica de validación se utiliza para realizar una simulación del sistema que se quiere desarrollar. Empleando está métrica para la actividad de los requisitos, se comprueba la corrección y completitud de la especificación de requisitos.

Para validarlos se utilizaron prototipos no funcionales, siguiendo las siguientes tareas:

- **Seleccionar quién evaluará el prototipo:** Se seleccionaron adecuadamente los usuarios que participaron en la evaluación, estos fueron: la analista principal del módulo, una analista con experiencia en el rol y dos jueces de la materia Administrativa.
- **Desarrollar escenarios de validación:** Se identificaron una serie de escenarios o tareas, los cuales fueron llevados a cabo por los usuarios utilizando los prototipos.
- **Ejecutar los escenarios:** Se ejecutaron los escenarios previstos teniendo en cuenta la diferencia entre el producto final y el prototipo. Para ello los analistas suplieron la funcionalidad no presente en el prototipo, proporcionando al usuario la información que necesitaba para ejecutar los escenarios.
- **Documentar los problemas:** Se confecciona la lista de problemas encontrados. Después se procede a corregir y retroalimentar al prototipo para que refleje los cambios.

La validación por prototipos no funcionales de esta investigación se hizo posible con la generación de páginas html. Estas fueron presentadas al cliente en dos iteraciones, identificándose 5 no conformidades en la primera, las cuales fueron:

- Adicionar motivos de interposición
- Plantilla incorrecta en el visor correspondiente al abogado
- Nombre incorrecto en un prototipo de la vista correspondiente al juez ponente.
- Inconsistencia entre los datos del visor correspondiente al juez ponente y los motivos de subsanación.
- Seleccionar tipo de escrito del prototipo de la vista correspondiente a la secretaria.

Al haber finalizado este proceso se obtuvieron resultados satisfactorios para los 15 prototipos no funcionales de interfaz de usuario propuestos en la 2da iteración. En la siguiente figura se muestra con más detalles estos resultados:

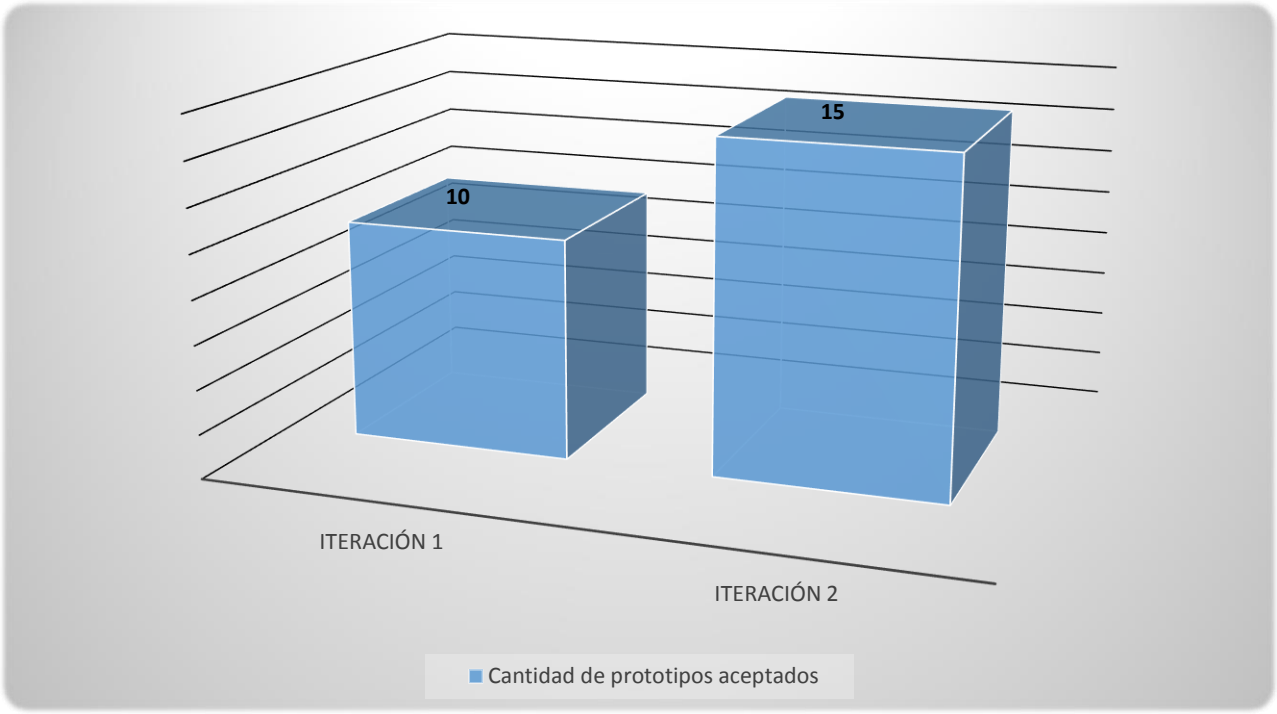


Figura 11 Resultado de la validación mediante prototipos

3.2.3 Validación del Diseño

Tamaño Operacional de las Clases (TOC)

El método TOC mide el tamaño general de una clase tomando el valor de la cantidad de operaciones y el número de atributos que están encapsulados dentro de dicha clase. A continuación se muestran las siguientes tablas con los valores altos y bajos de TOC para los atributos de calidad definidos para esta métrica:

Atributos de Calidad	Valores Altos del TOC
Reutilización	Disminuye la reutilización de las clases
Complejidad de implementación	Aumenta la complejidad de las pruebas del sistema
Responsabilidad	Aumenta la responsabilidad

Tabla 4 Atributos de Calidad para los valores altos de la métrica de TOC

Atributos de Calidad	Valores Bajos del TOC
Reutilización	Aumenta la reutilización de las clases
Complejidad de implementación	Disminuye la complejidad de las pruebas del sistema
Responsabilidad	Disminuye la responsabilidad

Tabla 5 Atributos de Calidad para los valores bajos de la métrica de TOC

En la siguiente tabla se muestra como se calcula cada una de los atributos de calidad aplicando la validación del diseño TOC:

Atributo de Calidad	Categoría	Criterio
Responsabilidad	Baja	**TOC** < =Promedio (Prom).
	Media	**TOC** Entre Prom. y 2* Prom.
	Alta	**TOC** >2*Prom.
Complejidad de implementación	Baja	**TOC** < =Prom.
	Media	**TOC** Entre Prom. y 2* Prom.
	Alta	**TOC** >2*Prom.
Reutilización	Bajo	**TOC** >2*Prom.
	Media	**TOC** Entre Prom. y 2* Prom.
	Alta	**TOC** < =Prom.

Tabla 6 Calculo para la métrica TOC

En la tabla que se mostrará a continuación se aplica la métrica TOC, donde se escogieron 8 clases con mayor complejidad de implementación en sus operaciones:

Clase	Operaciones	Responsabilidad	Complejidad de implementación	Reutilización
BaseController	23	Media	Media	Media
RegistrarEscritoInterposicionRecursoController	13	Baja	Baja	Alta
DisponerSobreEscritoInterposicionRecursoController	8	Baja	Baja	Alta
BaseGtr	22	Media	Media	Media
RegistrarEscritoInterposicionRecursoGtr	18	Media	Media	Media
DisponerSobreEscritoInterposicionRecursoGtr	4	Baja	Baja	Alta
BaseTableModel	29	Alta	Alta	Baja
PantallaInicialGtr	8	Baja	Baja	Alta

Tabla 7 Resultado de la aplicación de la métrica TOC

Al aplicar el método a estas 8 clases se obtuvo un total de 125 operaciones calculándose un promedio de 15,6 operaciones. Después de haber estudiado los resultados se obtuvieron para cada atributo de calidad una responsabilidad de un 12.5% de alta, 37.5% de media y un 50% de baja. Para la Complejidad de pruebas se tiene un 12.5% de alta, 37.5% de media y un 50% de baja. La Reutilización tiene un 50% de alta, 37.5% de media, 12.5% de baja. Analizando los resultados se puede concluir que más del 12% de las clases poseen un bajo nivel de complejidad por lo que se puede realizar una alta explotación de la reutilización. En la siguiente figura se muestran estos resultados.

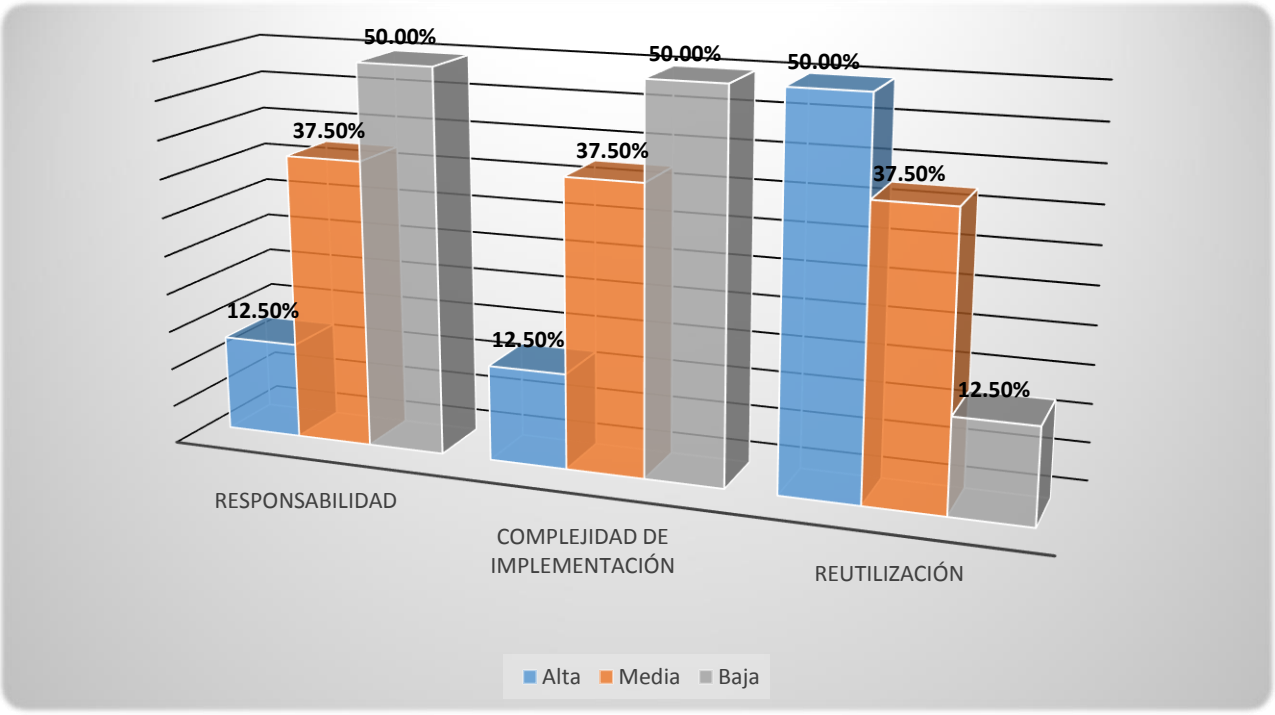


Figura 12 Análisis Obtenido del TOC

Relación entre Clases (RC)

Esta métrica se utilizó para conocer el número de relaciones uso de una clase con otra y evalúa los siguientes atributos de calidad:

Atributos de calidad	**Valores altos del RC**
Acoplamiento	Aumento el acoplamiento de la clase
Complejidad de mantenimiento	Aumenta la complejidad del mantenimiento de la clase.
Reutilización	Disminuye el grado de reutilización de la clase
Cantidad de pruebas	Aumenta la Cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 8 Atributos de Calidad para los valores altos de la métrica de RC

Atributos de calidad	**Valores cajos del RC**
Acoplamiento	Disminuye el acoplamiento de la clase
Complejidad de mantenimiento	Disminuye la complejidad del mantenimiento de la clase.

Reutilización	Aumenta el grado de reutilización de la clase
Cantidad de pruebas	Disminuye la Cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 9 Atributos de Calidad para los valores bajos de la métrica de RC

La tabla que se muestra a continuación posee los valores utilizados para evaluar los atributos de calidad mencionados anteriormente:

Atributo de calidad	**Categoría**	**Criterio**
Acoplamiento	Ninguno	RC=0
	Bajo	RC=1
	Medio	RC=2
	Alto	RC>2
Complejidad de mantenimiento	Bajo	RC<= Promedio
	Media	Promedio<=RC<=2*Promedio
	Alta	RC>2*Promedio
Reutilización	Bajo	RC>2*Promedio
	Media	Promedio< RC <=2*Promedio
	Alta	RC<=Promedio
Cantidad de pruebas	Bajo	RC <= Promedio
	Media	Promedio<= RC <2*Promedio
	Alta	RC >= 2*Promedio

Tabla 10 Criterios de evaluación para la métrica RC

A continuación se muestra un ejemplo del resultado de la aplicación de esta métrica al diseño de la solución haciendo uso de las clases con mayores relaciones uso:

Clase	RC	Acoplamiento	Complejidad de mantenimiento	Reutilización	Cantidad de pruebas
BaseController	1	Baja	Baja	Alta	Baja
RegistrarEscritoInterposicionRecursoController	5	Alta	Alta	Baja	Alta
BaseGtr	1	Baja	Baja	Alta	Baja
RegistrarEscritoInterposicionRecursoGtr	7	Alta	Alta	Baja	Alta
PersonaRepository	2	Media	Baja	Alta	Baja
BaseTableModel	1	Baja	Baja	Alta	Baja
MotivoCasacionTM	1	Baja	Baja	Alta	Baja
ParteAbogadoDefensorTM	1	Baja	Baja	Alta	Baja

Tabla 11 Resultado de la aplicación de la métrica RC

Al aplicar la métrica a estas 8 clases se obtuvo un total de 19 dependencias calculándose un promedio de 2,4 relaciones. Después de evaluar los resultados obtenidos de cada uno de los atributos de calidad que mide esta métrica se puede llegar a la conclusión de que el 62,5% de las clases tienen un bajo acoplamiento, el 12,5% es de nivel medio y el 25% es de nivel alto. Por otra parte el 75% de las clases poseen una baja complejidad de mantenimiento, 0% de nivel medio y el 25% de nivel alto, también se puede decir que el 25% de las clases poseen una reutilización baja, tienen un 0% de nivel medio y de nivel alto tiene un 75%. Por último tenemos que existe un 75% de las clases con una baja cantidad de pruebas, una media del 0% y un 25% de alta. Como se puede observar en la figura el acoplamiento posee un nivel bajo por lo que existe poca dependencia entre las clases trayendo como consecuencia una alta probabilidad de reutilización. También se puede apreciar que existe un bajo nivel de complejidad de mantenimiento por lo que a la hora de optimizar métodos y demás operaciones no es necesario realizar una alta cantidad de pruebas. En la siguiente figura se evidencian los resultados:

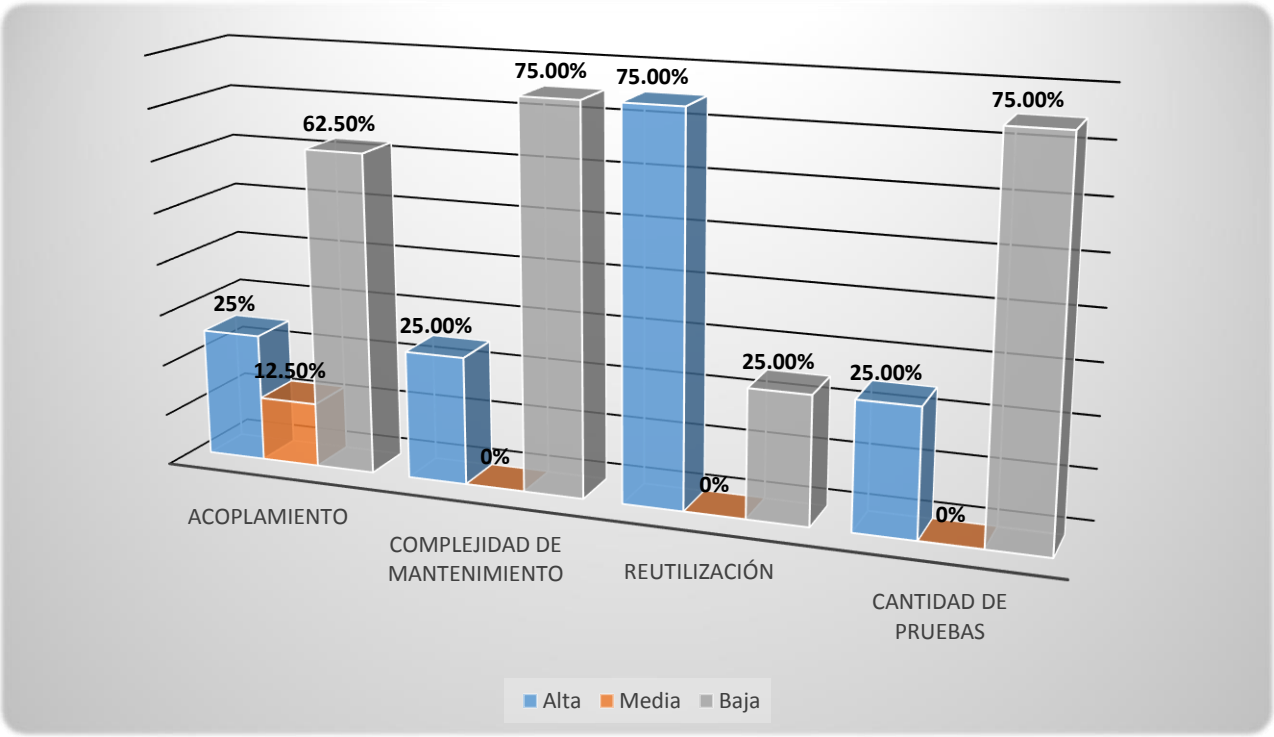


Figura 13 Análisis Obtenido de la RC

3.2.4 Validación utilizando pruebas

Pruebas de caja blanca

Las pruebas de caja blanca son aquellas pruebas estructurales que se centran en los detalles procedimentales del *software*, por lo que su diseño está fuertemente ligado al código fuente. Para poder validar el código se escogieron distintos valores de entrada para examinar cada uno de los posibles flujos de ejecución del programa y cerciorarse de que se devuelven los valores de salida adecuados. En la siguiente imagen se muestra una función de la cual se va a aplicar el método camino básico:

```php
public function cargarParteAbogadoDefensorTM($idExpediente) {
    $expediente = $this->obtenerExpediente($idExpediente); /1
    $result = array(); /1
    $litigantes = $expediente->getLitigante(); /1
    foreach ($litigantes as $litigante) { /2
        if ($litigante->getTipoFormaComparecer()->getId() ==
         EFormaComparecer::Demandado || $litigante->getTipoFormaComparecer()->getId() ==
          EFormaComparecer::Demandado_De_La_Administracion
          || $litigante->getTipoFormaComparecer()->getId() ==
           EFormaComparecer::Demandante) { /3
            $fila ['id'] = $litigante->getPersona()->getId(); /4
            $fila ['parte'] = $litigante->getPersona()->getNombreCompleto(); /4
            $abogadoDefensor = $this->getEm()->getRepository('ComunBundle:AG\Representante')
            ->obtenerRepresentantesxLitigante($litigante->getId(), EFormaComparecer::Abogado); /4
            if (count($abogadoDefensor) > 0) { /5
                $fila ['abogadoDefensor'] = $abogadoDefensor[0]->getNombrePropio(); /6
            } else {
                $fila ['abogadoDefensor'] = ''; /7
            }
            $result[] = $fila; /8
        }
    }
    return $result; /9
}
```

Figura 14 Método cargarParteAbogadoDefensorTM ($idExpediente)

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

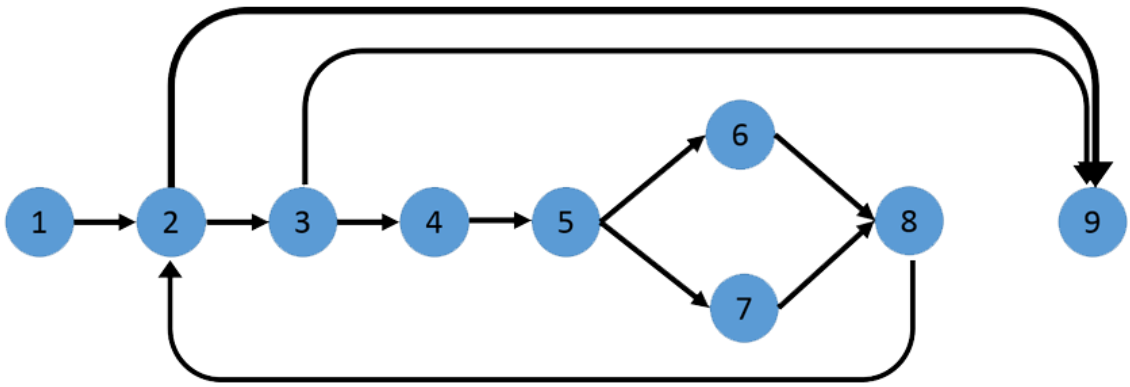
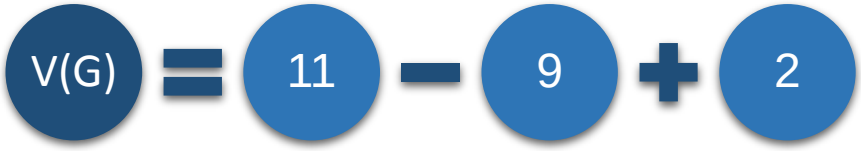
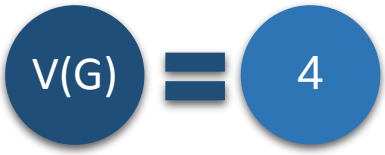


Figura 15 Flujo correspondiente al método cargarParteAbogadoDefensorTM ($idExpediente)

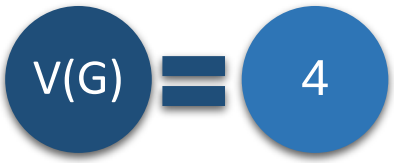
Para llevar a cabo la prueba de caja blanca en el segundo paso se tiene que obtener la complejidad ciclomática del flujo, lo cual proporciona una idea de la complejidad lógica del programa que se prueba.

La complejidad ciclomática V(G) de un grafo de flujo se define como:
V(G) = cantidad de aristas(A) – cantidad de nodos(N) + 2

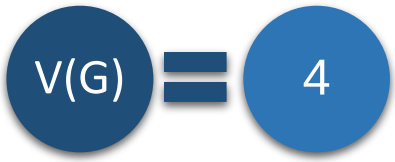




V(G)= cantidad de nodos predicados(P) + 1



V(G)= número de regiones del grado de flujo(R)



La complejidad ciclomática del grafo es 4, coincidiendo de esta forma con el conjunto básico de caminos independientes. El próximo paso es determinar los caminos básicos, cuya especificación es la siguiente:

Camino básico #1: 1-2-9

Camino básico #2: 1-2-3-9

Camino básico #3: 1-2-3-4-5-6-8-2-9

Camino básico #4: 1-2-3-4-5-7-8-2-9

Para el siguiente paso se debe preparar los casos de prueba que sigan la ejecución de cada camino.

Camino básico #1:

Descripción: El expediente no presenta litigantes.

Entrada: La variable *$idExpediente*.

Resultados esperados: El *grid* no carga datos. El resultado obtenido fue lo esperado.

Camino básico #2:

Descripción: El expediente tiene litigantes, pero ninguno de ellos es demandante, demandado o demandado de la administración.

Entrada: La variable *$idExpediente*.

Resultados esperados: El *grid* no carga datos. El resultado obtenido fue lo esperado.

Camino básico #3:

Descripción: El litigante en cuestión tiene abogado defensor.

Entrada: La variable *$idExpediente*.

Resultados esperados: El *grid* carga el litigante y su abogado defensor. El resultado obtenido fue lo esperado.

Camino básico #4:

Descripción: El litigante en cuestión no tiene abogado defensor.

Entrada: La variable *$idExpediente*.

Resultados esperados: El *grid* carga solo el litigante. El resultado obtenido fue lo esperado.

Pruebas de caja negra

Para validar la solución propuesta se realizaron pruebas funcionales utilizando pruebas de caja negra y se empleó la técnica de flujo lógico central, en el cual los casos de pruebas se especifican como un conjunto de operaciones que deben realizarse consecutivamente sobre el software bajo prueba.

Luego de aplicar los métodos de prueba a las funcionalidades implementadas, se obtuvieron resultados satisfactorios desde el punto de vista interno y funcional, atendiendo al correcto comportamiento del mismo ante diferentes situaciones. Las no conformidades detectadas fueron debidamente atendidas en aras de lograr el correcto funcionamiento de la solución desarrollada. En la gráfica se muestra la cantidad de no conformidades detectadas en cada iteración de pruebas realizadas:

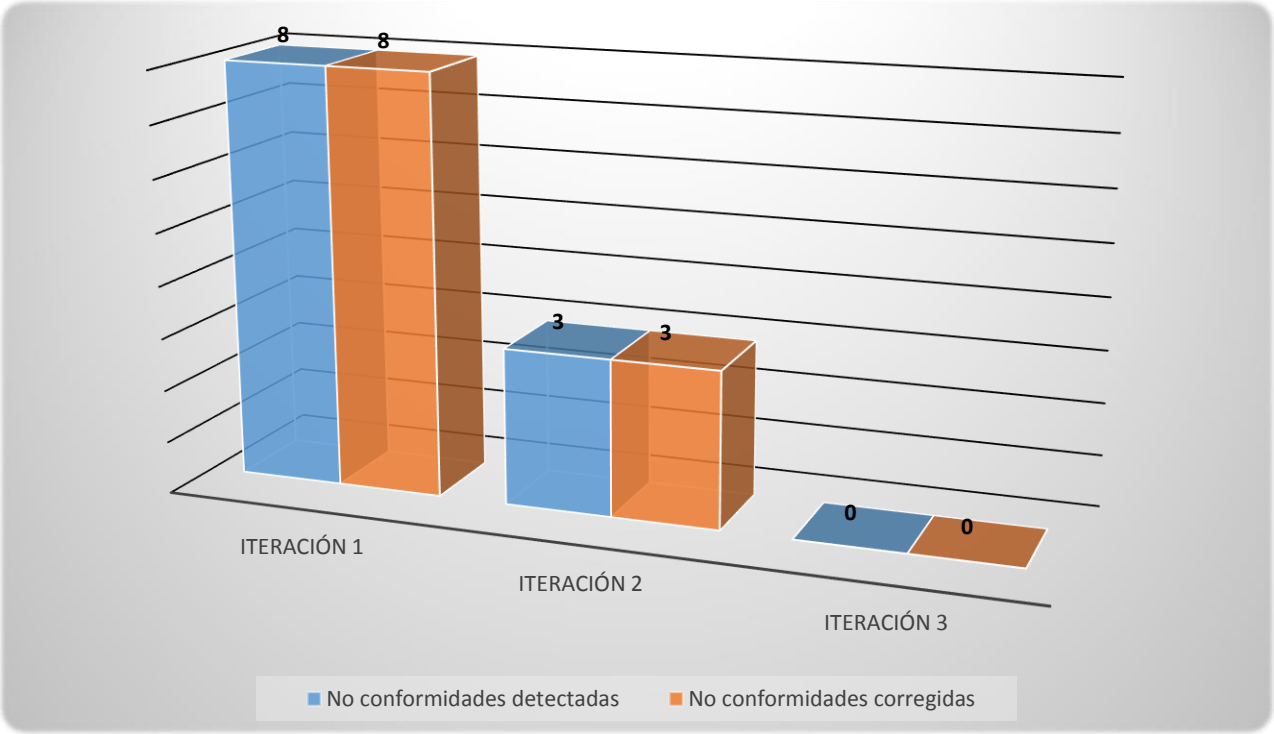


Figura 16 Cantidad de no conformidades detectadas por iteración

3.2.5 Validación de la investigación

En este acápite se procede validar la investigación mediante una encuesta donde se evalúa el uso del módulo Casación del subsistema Administrativo del proyecto SITPC. La misma se inicia con la selección del conjunto de expertos a los cuáles se les va hacer la encuesta, por consecuente se presentan qué atributos deben tener los mismos para evaluar la solución propuesta con respecto a la forma en que se desarrolla actualmente el procedimiento. Para finalizar se muestran los resultados con el uso de la aplicación.

Selección del conjunto de expertos

La encuesta se realizó a un conjunto de expertos ver (anexo 1) del centro CEGEL de la facultad 3 dirigido a un conjunto de desarrolladores y analistas, entre otros. Los encuestados reúnen un grupo de atributos que les sirve para conocer cómo funciona el sistema actualmente y que cambios tendrá cuando se despliegue la aplicación.

Atributos de los expertos

En la selección de los expertos mencionados anteriormente, se tuvo en cuenta las siguientes características:

- Conocimiento global de la solución propuesta.
- Presentar años de experiencia entre 3 y 7.
- Participación directa con los clientes en:

- ✓ Reuniones de aceptación.
- ✓ Talleres de conciliación.
- ✓ Eventos internacionales asociados a la justicia y el derecho.
- ✓ Pruebas piloto.
- ✓ Reuniones sobre el impacto del uso del sistema.

- Desempeñar en el proyecto uno de los roles principales.

Al haber definido el conjunto de expertos teniendo en cuenta los atributos de los mismos, se da comienzo a la ejecución de la encuesta (ver anexo 2) y por consecuente el análisis de los resultados obtenidos como se muestra a continuación.

Resultados de la encuesta

Los resultados de esta encuesta (ver anexo 3) arrojaron que la mayoría de los expertos coincidieron que con el uso de la aplicación los expedientes están organizados en una base de datos, por lo que aumenta la disponibilidad facilitando el acceso simultáneo de varias personas. Por otra parte se llegó a un consenso de que el tiempo de respuesta de la búsqueda dichos expedientes oscila entre 10 minutos y 1 hora, por lo que agiliza el proceso de ejecución de una sentencia. Todo esto en conjunto al alto grado de satisfacción por parte de los clientes ya que los mismos reciben notificaciones diarias sobre el vencimiento de los términos. Se apreció también, que el empleo de la solución facilitará basándose en lo que dicta la ley para tramitar el proceso, que se generen de igual forma los documentos que corresponden en cada momento, a pesar de la provincia en que se trabaje, conservando siempre dentro del propio documento las regiones que pueden ser o no editables por los usuarios.

3.3 Conclusiones Parciales

En el presente capítulo se evaluó el cumplimiento de las funcionalidades del sistema de acuerdo al análisis de las métricas, validándose el diseño y verificando que la implementación se haya hecho correctamente, trayendo como resultado que:

- Con la realización del diseño de la solución propuesta se presentaron buenos niveles de calidad y fiabilidad, donde las métricas aplicadas indicaron como principales resultados un predominio de la reutilización, del bajo acoplamiento y de la baja complejidad en la implementación y el mantenimiento.
- Al realizar los métodos de prueba unidad se comprobó que todas las funcionalidades especificadas fueron implementadas y se corrigieron las no conformidades detectadas a lo largo de las iteraciones definidas para la aplicación de estas pruebas.

CONCLUSIONES GENERALES

Con la realización del trabajo de diploma se desarrolló el paquete de análisis interposición de recurso del módulo casación del subsistema administrativo del proyecto SITPC, trayendo como resultado una gran mejoría en los procesos ejecutados en la materia Administrativa de los TPC. Los resultados alcanzados mediante la terminación de la aplicación permiten concluir que:

- En la elaboración del marco teórico se realizó el estudio del arte lo que demostró que los sistemas existentes no responden a las necesidades del procedimiento de Casación de los Tribunales Populares en su instancia provincial. Además se justificó la selección de la metodología, herramientas, tecnologías y patrones utilizados en la solución.
- Mediante las entrevistas a los especialistas del negocio y la tormenta de ideas se realizó el levantamiento de requisitos, resultando un total de 105 requisitos identificados, entre funcionales y no funcionales, los cuales se agruparon en CU y se describieron para su posterior diseño e implementación.
- En la etapa de análisis y diseño se realizaron los diagramas de clases del diseño a partir de los casos de uso, además del diseño de la base de datos, en los cuales se utilizaron patrones para su solución, obteniéndose finalmente la cantidad de clases y métodos a implementar, así como la relación entre clases.
- Para las validaciones se realizaron un conjunto de métricas y pruebas a la solución propuesta con el objetivo de comprobar la calidad de la misma, de esta forma se verificó que los requisitos identificados hayan sido implementados y que el sistema esté en correspondencia con lo que se diseñó.

RECOMENDACIONES

- Continuar con la implementación de los casos de uso del paquete de análisis de Interposición de recurso de Casación de la materia Administrativa en los TPP después del caso de uso Elevar actuaciones al TSP.
- Actualizar las librerías y marcos de trabajo de la aplicación SIT a las últimas versiones disponibles, con el fin de lograr mejoras sustanciales en el rendimiento del sistema.

BIBLIOGRAFÍA

Referencias

Alejandre Nuevo, Ricardo Alberto. 2013. *Desarrollo del módulo Administración y Gobierno.* La Habana : s.n., 2013. Visual Paradigm 8.0.

Chiesa, Florencia. 2004. *METODOLOGÍA PARA SELECCIÓN DE SISTEMAS ERP.* 2004. Metodología.

Constantinos Stamatoulos. 2014. Enciclopedia jurídica. [En línea] 2014. [Citado el: 3 de Febrero de 2015.] http://www.enciclopedia-juridica.biz14.com.

Definicion.de. 2014. Definicion.de. [En línea] 2014. [Citado el: 3 de marzo de 2015.] http://definicion.de/html/.

Doctrine Project Team. 2011. *Doctrine 2 ORM Documentation.* 2011.

Domingo Moquillaza Henríquez, Santiago. 2010. *Programación en N capas.* San Marcos,Peru : s.n., 2010.

Eckel, Bruce. 2011. *Pensar en C++.* 2011.

EcuRed. [En línea] [Citado el: 5 de marzo de 2015.] http://www.ecured.cu/index.php/Arquitectura_de_tres_niveles.

—. [En línea] [Citado el: 5 de marzo de 2015.] http://www.ecured.cu/index.php/Patr%C3%B3n_Modelo_Vista_Controlador.

—. [En línea] [Citado el: 9 de marzo de 2015.] http://www.ecured.cu/index.php/Arquitectura_de_software.

—. [En línea] [Citado el: 11 de marzo de 2015.] http://www.ecured.cu/index.php/Diagrama_de_Clase.

—. 2012. [En línea] marzo de 2012. [Citado el: 11 de marzo de 2015.] http://www.ecured.cu/index.php/Diagrama_de_despliegue.

—. [En línea] [Citado el: 20 de marzo de 2015.] http://www.ecured.cu/index.php/Diagrama_Entidad_Relación.

—. [En línea] [Citado el: 23 de marzo de 2015.] http://www.ecured.cu/index.php/Patrones_de_dise%C3%B1o_de_bases_de_datos.

Ecured. Ecured. [En línea] [Citado el: 4 de Febrero de 2015.] Herramientas CASE. http://www.ecured.cu/index.php/Herramienta_CASE.

—. 2012. Ecured. [En línea] 2012. [Citado el: 3 de marzo de 2015.] http://www.ecured.cu/index.php/Axure.

—. ecured.cu. [En línea] [Citado el: 3 de marzo de 2015.] http://www.ecured.cu/index.php/Twig.

Editorial Aranzadi. 2013. legaltoday.com. [En línea] 2013. [Citado el: 6 de Febrero de 2015.] http://www.legaltoday.com/actualidad/noticias/thomson-reuters-aranzadi-lanza-aranzadi-infolex-nube-el-software-definitivo-para-gestionar-un-despacho-de-forma-virtual.

Eduardo Torres Moreno, Miguel. 2009. *Técnicas de Levantamiento de Requerimientos con Innovación.* Colombia : s.n., 2009.

Eguiluz Pérez, Javier. 2012. *Desarrollo_agil_Symfony2.1.* 2012. Symfony 2.1.

Eguíluz Pérez, Javier. 2008. *Introducción a AJAX.* 2008.

—. 2013. *Introducción a CSS.* 2013.

Eguiluz Pérez, Javier. 2009. *Introducción a JavaScript.* 2009. JavaScript.

Fowler, Martin. 2002. *Patterns of Enterprise Application Architecture.* 2002.

Genbeta. 2012. http://www.genbetadev.com/desarrollo-web. [En línea] 23 de febrero de 2012. [Citado el: 3 de marzo de 2015.] http://www.genbetadev.com/desarrollo-web/disenando-tu-nuevo-proyecto-web-con-bootstrap-2-0.

Google. docs.google.com. [En línea] [Citado el: 13 de marzo de 2015.] https://docs.google.com/document/d/1rbxDFM0zsbFDNRZeM2FoXfRDbYSiSt6tCdbYPA0qdzs/edit?hl=en_US&pli=1.

Larman, Craig. 1999. *UML y Patrones.* 1999. UML Lenguaje unificado de modelado.

—. 2005. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 2005. Vol. 2.

—. 1999. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 1999. Vol. 1, UML Lenguaje unificado de modelado.

Leyva Hernández, Suany. 2013. *Sistema informático para la conformación automática de tribunales.* La Habana : s.n., 2013. RUP Rational Unufied Process.

Ministerio de Justicia. 2008. Gaceta oficial de la República de Cuba. [En línea] 2008. [Citado el: 3 de Febrero de 2015.] Tribunales Populares Cubanos. http://www.gacetaoficial.cu/html/tribunalespopulares.html.

Morell, Daniel E. Castro. 2008. *Sistema para la Tramitación de procesos penales.* 2008.

Netbeans. 2015. Netbeans. [En línea] 2015. [Citado el: 5 de Febrero de 2015.] http://netbeans.org/features/index.html.

NetFincas Software. 2008. Software Jurídico. [En línea] 2008. [Citado el: 6 de Febrero de 2015.] www.softwareabogados.net/software+juridico.html.

PHP Documentation Group . 2014. *Manual de PHP.* 2014.

PROMAD. 2013. PROMAD. [En línea] 2013. [Citado el: 6 de Febrero de 2015.] http://www.promad.com.br/sistema-juridico.

—. 2013. PROMAD. [En línea] 2013. [Citado el: 6 de Febrero de 2015.] http://www.promad.com.br/sistema-juridico/politica-de-preco.

—. 2013. PROMAD. [En línea] 2013. [Citado el: 6 de Febrero de 2015.] http://www.promad.com.br/sobre-o-promad/duvidas-comuns.

Sáchez, Pablo. 2004. *Subversion como herramienta para el control del.* Santander, España : s.n., 2004.

Stephen A. White, PhD Derek Miers. 2009. *BPMN, Guía de referencia y modelado.* 2009. BPMN.

The Apache Software Foundation. 2015. Apache HTTP Server. [En línea] 2015. [Citado el: 5 de Febrero de 2015.] http://http://httpd.apache.org/.

The PostgreSQL Global Development Group . 2015. PostgreSQL. [En línea] 2015. [Citado el: 5 de Febrero de 2015.] http://www.postgresql.org/.

Thomson Reuters. 2014. Aranzadi Infolex. [En línea] 2014. [Citado el: 6 de Febrero de 2015.] www.aranzadi.es/software.

—. 2014. Thomson Reuters Aranzadi. [En línea] 2014. [Citado el: 6 de Febrero de 2015.] https://www.infolexnube.es.

Universidad Politécnica de Valencia. 2003. *Rational Unified Process (RUP).* Espeña, Valencia : s.n., 2003.

Xavier Ferré Grau, María Isabel Sánchez Segura. 2008. *Desarrollo Orientado a Objetos con UML.* 2008.

evo, Ricardo Alberto. 2013. *Desarrollo del módulo Administración y Gobierno.* La Habana : s.n., 2013. Visual Paradigm 8.0.

Chiesa, Florencia. 2004. *METODOLOGÍA PARA SELECCIÓN DE SISTEMAS ERP.* 2004. Metodología.

Constantinos Stamatoulos. 2014. Enciclopedia jurídica. [En línea] 2014. [Citado el: 3 de Febrero de 2015.] http://www.enciclopedia-juridica.biz14.com.

Definicion.de. 2014. Definicion.de. [En línea] 2014. [Citado el: 3 de marzo de 2015.] http://definicion.de/html/.

Doctrine Project Team. 2011. *Doctrine 2 ORM Documentation.* 2011.

Eckel, Bruce. 2011. *Pensar en C++.* 2011.

EcuRed. [En línea] [Citado el: 5 de marzo de 2015.] http://www.ecured.cu/index.php/Arquitectura_de_tres_niveles.

—. [En línea] [Citado el: 5 de marzo de 2015.] http://www.ecured.cu/index.php/Patr%C3%B3n_Modelo_Vista_Controlador.

—. [En línea] [Citado el: 9 de marzo de 2015.] http://www.ecured.cu/index.php/Arquitectura_de_software.

Ecured. Ecured. [En línea] [Citado el: 4 de Febrero de 2015.] Herramientas CASE. http://www.ecured.cu/index.php/Herramienta_CASE.

—. 2012. Ecured. [En línea] 2012. [Citado el: 3 de marzo de 2015.] http://www.ecured.cu/index.php/Axure.

—. ecured.cu. [En línea] [Citado el: 3 de marzo de 2015.] http://www.ecured.cu/index.php/Twig.

Editorial Aranzadi. 2013. legaltoday.com. [En línea] 2013. [Citado el: 6 de Febrero de 2015.] http://www.legaltoday.com/actualidad/noticias/thomson-reuters-aranzadi-lanza-aranzadi-infolex-nube-el-software-definitivo-para-gestionar-un-despacho-de-forma-virtual.

Eguiluz Pérez, Javier. 2012. *Desarrollo_agil_Symfony2.1.* 2012. Symfony 2.1.

Eguíluz Pérez, Javier. 2008. *Introducción a AJAX.* 2008.

—. 2013. *Introducción a CSS.* 2013.

Eguiluz Pérez, Javier. 2009. *Introducción a JavaScript.* 2009. JavaScript.

Fowler, Martin. 2002. *Patterns of Enterprise Application Architecture.* 2002.

Genbeta. 2012. http://www.genbetadev.com/desarrollo-web. [En línea] 23 de febrero de 2012. [Citado el: 3 de marzo de 2015.] http://www.genbetadev.com/desarrollo-web/disenando-tu-nuevo-proyecto-web-con-bootstrap-2-0.

Larman, Craig. 1999. *UML y Patrones.* 1999. UML Lenguaje unificado de modelado.

—. 2005. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 2005. Vol. 2.

—. 1999. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 1999. Vol. 1, UML Lenguaje unificado de modelado.

Leyva Hernández, Suany. 2013. *Sistema informático para la conformación automática de tribunales.* La Habana : s.n., 2013. RUP Rational Unufied Process.

Ministerio de Justicia. 2008. Gaceta oficial de la República de Cuba. [En línea] 2008. [Citado el: 3 de Febrero de 2015.] Tribunales Populares Cubanos. http://www.gacetaoficial.cu/html/tribunalespopulares.html.

Netbeans. 2015. Netbeans. [En línea] 2015. [Citado el: 5 de Febrero de 2015.] http://netbeans.org/features/index.html.

NetFincas Software. 2008. Software Jurídico. [En línea] 2008. [Citado el: 6 de Febrero de 2015.] www.softwareabogados.net/software+juridico.html.

PHP Documentation Group . 2014. *Manual de PHP.* 2014.

PROMAD. 2013. PROMAD. [En línea] 2013. [Citado el: 6 de Febrero de 2015.] http://www.promad.com.br/sistema-juridico.

—. 2013. PROMAD. [En línea] 2013. [Citado el: 6 de Febrero de 2015.] http://www.promad.com.br/sistema-juridico/politica-de-preco.

—. 2013. PROMAD. [En línea] 2013. [Citado el: 6 de Febrero de 2015.] http://www.promad.com.br/sobre-o-promad/duvidas-comuns.

Sáchez, Pablo. 2004. *Subversion como herramienta para el control del.* Santander, España : s.n., 2004.

Stephen A. White, PhD Derek Miers. 2009. *BPMN, Guía de referencia y modelado.* 2009. BPMN.

The Apache Software Foundation. 2015. Apache HTTP Server. [En línea] 2015. [Citado el: 5 de Febrero de 2015.] http://http://httpd.apache.org/.

The PostgreSQL Global Development Group . 2015. PostgreSQL. [En línea] 2015. [Citado el: 5 de Febrero de 2015.] http://www.postgresql.org/.

Thomson Reuters. 2014. Aranzadi Infolex. [En línea] 2014. [Citado el: 6 de Febrero de 2015.] www.aranzadi.es/software.

—. 2014. Thomson Reuters Aranzadi. [En línea] 2014. [Citado el: 6 de Febrero de 2015.] https://www.infolexnube.es.