Universidad de las Ciencias Informáticas

Facultad 3

Centro de Gobierno Electrónico

Desarrollo del módulo Casación del subsistema Penal del proyecto de Informatización para la Gestión de los Tribunales Populares Cubanos

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

Autor (es):

Lilian Puzo Sifontes

René del Valle Rodríguez

Tutor (es):

Ing. Darián González Ochoa

Ing. Yeleny Almora Gálvez

Co-Tutor:

Ing. Daniel Alberto Ojeda Estrada

Cuidad de La Habana, junio de 2015

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores del trabajo “Desarrollo del módulo Casación del subsistema Penal del proyecto de Informatización para la Gestión de los Tribunales Populares Cubanos” y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio. Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

_____ _____

Autor: Lilian Puzo Sifontes Autor: René del Valle Rodríguez

_____ _____

Tutor: Ing. Yeleny Almora Gálvez Tutor: Ing. Darián González Ochoa

DEDICATORIA

De Lilian

A mis padres, por sus infinitos sacrificios, a mi hermana, mi familia, mi novio y mis amigos porque gracias a ellos estoy aquí.

De René

A toda mi familia, vecinos y amigos que de una forma u otra ayudaron a culminar mis estudios en esta universidad, en especial a mis padres, todo lo que soy hoy es gracias a ellos y para ellos.

RESUMEN

En los Tribunales Populares Cubanos se procesa un alto volumen de información, la misma se encuentra dispersa en múltiples libros de registro y archivos, por lo que la introducción de errores suele ser muy frecuente. Con el objetivo de dar solución a este problema la Facultad 3 de la Universidad de las Ciencias Informáticas cuenta con un proyecto, que tiene como propósito desarrollar un sistema que estandarice los procedimientos jurídicos. Uno de los procesos que se desea informatizar es el conocido como Recurso de Casación, perteneciente a la materia Penal.

En la presente investigación se realiza una descripción de las principales herramientas, el lenguaje de modelado y la tecnología para el desarrollo del módulo Casación, justificando la utilización de cada una de ellas. Se exponen además los artefactos generados durante la implementación del sistema. Al finalizar se muestran los resultados obtenidos al realizar pruebas a la solución desarrollada, haciendo uso de métricas, así como de pruebas de caja blanca y caja negra para comprobar la calidad del software. A partir de esta propuesta de solución se espera contribuir al logro de una mejor gestión de la información en la actividad jurisdiccional del país.

Palabras claves: materia Penal, Procedimiento de Casación, Tribunales Populares Cubanos.

Summary

In Cuban People's Courts a high volume of information is processed, it is scattered in many record books and files that is why introduction of errors is very frequent. In order to solve this problem 3rd Faculty of the Information Sciences University has a project with the objective to develop a system to standardize legal procedures. One of the processes to be computerized is known as Cassation resource, it pertains to criminal matters.

In the present investigation a description of the main tools, language modeling and technology for software development is done, justifying the use of each. The generated artifacts are also exposed during system implementation. At the end of the results obtained when testing the developed solution, using metrics as well as white box testing and black box to check the quality of software is. From this proposed solution and its follow-up, is expected to contribute to comprehensive computerization of judicial activity of the country.

Keywords: Cassation Procedure, Criminal matters, Cuban People's Courts.

ÍNDICE

Índice de figuras

INTRODUCCIÓN

Hoy en día las Tecnologías de la Información y las Comunicaciones (en lo adelante TIC) están inmersas en las actividades del quehacer humano, abarcando diversas áreas, cambiando la forma de pensar y de hacer el trabajo diario. Ante su indetenible desarrollo y su influencia en la dinámica de los procesos sociales en el mundo actual, donde el tiempo es un recurso cada vez más importante y escaso, se hace imprescindible el uso de las tecnologías para que los servicios a la población sean más ágiles, efectivos y estén a la altura de una sociedad que está siempre en movimiento.

En nuestro país mejorar y simplificar los procesos, ahorrar tiempo y dinero a través de los sistemas de información se ha hecho hoy una tarea primordial. Pues Cuba, con un proyecto de desarrollo que tiene como principios la igualdad social, la participación popular y la solidaridad, ha diseñado medidas y estrategias que permiten convertir los conocimientos y las tecnologías, en instrumentos a disposición de las profundas transformaciones, logrando más eficiencia en los procesos y por consiguiente, un aumento en la calidad de vida de los ciudadanos (Fernández 2013).

La Universidad de las Ciencias Informáticas (UCI), nacida bajo el calor de la Batalla de Ideas, fue creada con el objetivo de impulsar el desarrollo económico y la informatización de nuestro país. Fidel, como estratega fundacional, recomendó que la universidad fuese concebida como un centro de nuevo tipo, de alcance nacional, de características atípicas y tareas concretas en el proyecto de informatización de la sociedad cubana, con énfasis en la producción de Software. Reflejo de ello es la amplia gama de proyectos productivos que en ella se desarrollan, tanto para clientes nacionales como extranjeros.

En el campo del Derecho el avance de la tecnología informática no permanece al margen, el surgimiento de la Informática Jurídica es prueba de ello. Esta ciencia, surgida en los años 20, consiste en la aplicación de los ordenadores electrónicos orientada a la reducción de problemas jurídicos. Se hace referencia ahora con más énfasis a la informática jurídica de gestión y dentro de ella a la informática jurídica de gestión documental. Esta última facilita el procesamiento automático de documentos jurídicos (RIESTRA 2011).

A nivel internacional existen sistemas en el ámbito judicial que digitalizan algunos de los procesos de un tribunal, pero ellos no satisfacen las necesidades de los usuarios ni cumplen con la seguridad que este tipo de aplicaciones debe tener. Además, son creados por empresas privadas y no dan respuesta a las necesidades de los tribunales cubanos. Por otra parte, los módulos que los integran son completamente diferentes en cuanto a término, contenido, órganos que permiten comunicar y la manera de hacerlo, por lo que no se pueden reutilizar en nuestro país.

El sector jurídico cubano, aunque es uno de los más reservados en cuanto a la introducción de las tecnologías en sus procesos, no se ha quedado exento de estas trasformaciones. Hoy se han identificado varias áreas dentro del sector donde resulta imprescindible su informatización, con el fin de agilizar los procesos que se realizan, así como garantizar la celeridad de los mismos. Además, mediante la seguridad tecnológica se puede garantizar seguridad jurídica en los actos que se realizan de conformidad con las disposiciones legales que los regulan. También permite la estandarización de la forma de trabajo en cada instancia, ejercer el control y la debida supervisión que permita brindar servicios eficaces al ciudadano.

Entre las principales entidades jurídicas que se desean informatizar se encuentran el Ministerio de Justicia, las notarías, los registros civiles, los bufetes colectivos, la Fiscalía General de la República y el Sistema de Tribunales Populares Cubanos, siendo este último la esencia de este trabajo.

Una de las proyecciones estratégicas del Sistema de Tribunales Populares Cubanos es continuar desarrollando su infraestructura tecnológica y asegurar su gestión, administración y explotación en los principios de la Seguridad Informática. Para el logro de este propósito se concibió el acuerdo de colaboración con el Centro de Gobierno Electrónico (CEGEL) de la Universidad de las Ciencias Informáticas para trabajar en proyectos mutuamente beneficiosos en el ámbito de la formación, producción e investigación científica. Surge así el proyecto de Informatización para la Gestión de los Tribunales Populares Cubanos (SITPC), el cual contempla la informatización de los procedimientos de las materias: Administrativa, Civil, Económica, Laboral y Penal, de todos los tribunales del país.

Entre los procedimientos se encuentra el Recurso Casación de la materia Penal, que se ejecuta en el Tribunal Supremo Popular, en el cual está enmarcado el presente trabajo. Este consiste en el medio de impugnación por el cual, por motivos de derecho específicamente previstos en la ley, una parte postula la revisión de los errores jurídicos atribuidos a la sentencia de mérito que la perjudica, reclamando la correcta aplicación de la ley sustantiva, o la anulación de la sentencia, y una nueva decisión (Rivero 2008).

Es un recurso de gran importancia porque materializa un acto de voluntad del litigante, por el que solicita la revisión de la sentencia, amparándose en un error de derecho al juzgar o en un error o vicio procesal que desnaturaliza la validez de la sentencia emitida. Permite el control de la actividad judicial porque uno de los fines de la casación penal reside en el aseguramiento de una protección jurídica realista, pudiendo ser presentadas a la revisión del tribunal sólo aquellas partes de la decisión de los jueces de mérito que son independientes del paso del tiempo y que, por ello, no son del dominio natural del juez de primera instancia (Rivero 2008).

Se elevan en casación al Tribunal Supremo Popular las actuaciones de procesos ordinarios recurridos del tribunal provincial. Los asuntos pueden interponerse por Infracción de la Ley y por

Quebrantamiento de Forma y se priorizan para la asignación a un juez, en relación a si tienen preso o no, dando la preferencia a aquellas causas con presos. Luego de analizar el caso se dicta sentencia y se genera la resolución para la devolución de las actuaciones al Tribunal Provincial Popular (Rivero 2008).

En el desarrollo diario de este procedimiento en el Tribunal Supremo se presentan varias dificultades, pues el mismo se realiza de forma manual. El almacenamiento de los expedientes se realiza en estantes lo que provoca su deterioro debido a la humedad, insectos y roedores. Durante la ejecución de los procesos se generan documentos como escritos, providencias y autos, en los cuales no se reutiliza la información que es común para todos ellos, teniendo que escribirse la misma repetidas veces. Los problemas antes mencionados influyen negativamente en la rapidez del proceso y en ocasiones el tiempo de tramitación es elevado, debido a que los documentos que se necesitan aún no han sido generados.

El acceso simultáneo por el personal autorizado a los expedientes de cada uno de los procesos legales que se realizan, se dificulta, puesto que se realiza a través de la secretaria del tribunal, conllevando a demoras en su respuesta por el esfuerzo humano requerido, debido a la organización actual de los documentos.

Los expedientes se almacenan en formato duro en los archivos que se encuentran en la secretaría del tribunal y aunque están ubicados en estantes agrupados por años, el acceso a cualquier expediente se torna complejo, por el gran volumen de información. Los trabajadores necesitan tener a su alcance los expedientes que se encuentran abiertos y otros que pueden estar cerrados para realizar su consulta, estudio y análisis. Además esta actividad se puede ver afectada en el transcurso del tiempo por el inevitable deterioro.

Según la ley, los trámites tienen un término o plazo para realizarse y una vez que se vencen, la secretaria tiene que dar cuenta del vencimiento de este término al juez para que dicte una resolución. Actualmente por el alto contenido de trabajo de los jueces y secretarias, se vencen los términos sin que el proceso sea resuelto, provocando así la existencia de información inconsistente y malestar en los involucrados.

Ante la problemática anteriormente planteada surge el siguiente **problema a resolver**: ¿Cómo mejorar la gestión de la información del Recurso de Casación Penal de manera que estandarice la tramitación en el Tribunal Supremo Popular?

Objeto de estudio: El procedimiento Recurso de Casación Penal del Tribunal Supremo Popular.

Objetivo general: Desarrollar el módulo que informatice el procedimiento Recurso de Casación Penal de manera que contribuya a mejorar la gestión de la información y a la estandarización de la tramitación del mismo en el Tribunal Supremo Popular.

Campo de acción: Informatización de la gestión de los procesos judiciales relacionados con el procedimiento Recurso de Casación Penal en los Tribunales Populares Cubanos.

Objetivos específicos:

- Elaborar el marco teórico de la investigación para sustentar el desarrollo de software y la aplicación de la Informática Jurídica mediante el estudio de los principales referentes teóricos.
- Realizar el levantamiento de los procesos de negocio y la Ingeniería de Requisitos hasta la especificación para identificar las necesidades del procedimiento que se desea informatizar.
- Obtener el modelo de diseño para lograr la aproximación a la implementación.
- Realizar la implementación de los componentes de software para obtener el producto que el cliente desea.
- Validar los resultados obtenidos mediante la aplicación de métricas y la realización de pruebas para satisfacer las necesidades del cliente.

Tareas a cumplir:

1. Caracterización del Sistema de Tribunales Populares Cubanos.
2. Análisis del procedimiento Recurso de Casación Penal del Tribunal Supremo Popular.
3. Fundamentación del uso de la metodología de desarrollo de software RUP en el marco del proyecto de informatización para la gestión de los Tribunales Populares Cubanos.
4. Fundamentación del uso de la herramienta CASE Visual Paradigm para la representación gráfica de los artefactos que se obtienen en el proyecto.
5. Fundamentación del uso de la notación BPMN y el Lenguaje Unificado de Modelado (UML) según los artefactos que se obtienen de la aplicación de la metodología del proyecto.
6. Fundamentación del uso de los marcos de trabajo Symfony2 y Bootstrap para la implementación de la solución propuesta.
7. Fundamentación del uso de PHP como lenguaje de programación del proyecto y JQuery para el trabajo con HTML.
8. Fundamentación del uso de PostgreSQL y Doctrine como sistema gestor de base de datos y diseñador del modelo de datos y consultas respectivamente.
9. Descripción de la arquitectura de software base para la implementación del sistema.
10. Descripción del negocio del procedimiento Recurso de Casación Penal.
11. Levantamiento de los requisitos funcionales y realización de diagrama de casos de uso del procedimiento Recurso de Casación Penal.

12. Diseño de los prototipos de interfaz de usuario del procedimiento Recurso de Casación Penal.
13. Realización de los diagramas de clases del diseño del procedimiento Recurso de Casación Penal.
14. Realización de los diagramas de secuencia del procedimiento Recurso de Casación Penal.
15. Implementación de los casos de uso del procedimiento Recurso de Casación Penal.
16. Realización del diagrama de componentes y el de despliegue.
17. Aplicación de métricas para validar el diseño del procedimiento Recurso de Casación Penal.
18. Diseño de casos de prueba para la verificación de las funcionalidades del procedimiento Recurso de Casación Penal.
19. Realización de pruebas funcionales a los requisitos acordados con el cliente del procedimiento Recurso de Casación Penal.
20. Solución de no conformidades al finalizar cada iteración de pruebas.
21. Validación de la propuesta de solución.

Durante la realización de este trabajo se utilizarán diferentes métodos científicos para estudiar las características objeto de la investigación.

Los métodos científicos son la forma de abordar la realidad, de estudiar la naturaleza, la sociedad y el pensamiento, con el propósito de descubrir su esencia y sus relaciones. Para dar cumplimiento a las tareas se utilizarán los siguientes métodos:

Métodos teóricos: Permiten estudiar las características del objeto de investigación que no son observables directamente.

-Analítico–Sintético: Se refleja su uso en el estudio teórico de la investigación y un análisis previo sobre el funcionamiento del Recurso Casación asociado al procedimiento Penal.

-Modelación: Se utiliza para la creación de modelos que representen abstracciones que son objeto de investigación, con el objetivo de comprender el funcionamiento del Recurso Casación asociado a la materia Penal que se lleva a cabo en los Tribunales Populares de Cuba.

Métodos empíricos:

-Medición: Se realiza con el objetivo de obtener información numérica acerca de una propiedad o cualidad del objeto de investigación a partir de la aplicación de métricas.

-Entrevistas: Se realizan con el objetivo de adquirir una comprensión detallada de lo que necesitan los clientes y para poder conocer cómo podrían interactuar con el sistema y las dificultades a las que se enfrentan en la actualidad.

Estructura de la Tesis

El presente trabajo de diploma consta de 3 capítulos:

Capítulo 1. Fundamentación Teórica: Este capítulo aborda lo relacionado con la Ingeniería de Requisitos y los sistemas informáticos enmarcados en el dominio de las aplicaciones web existentes a nivel internacional y nacional. Además, describe el proceso y la metodología de desarrollo, la herramienta CASE, los lenguaje de modelado, las herramientas para el desarrollo del módulo, así como técnicas y métricas utilizadas para medir la calidad de un sistema.

Capítulo 2. Propuesta de Solución: En este capítulo se desarrollan todas las actividades de elicitación, análisis y especificación de la Ingeniería de Requisitos, obteniendo la especificación de los requisitos del sistema, el diagrama de casos de uso, diagramas de clases del diseño, diagramas de secuencia, modelo de datos, además del modelo de despliegue del Recurso Casación. Se exponen los artefactos generados producto del análisis y el diseño del Recurso Casación y las descripciones de éstos. Además se presenta el resultado de la implementación haciendo uso de patrones.

Capítulo 3. Validación de resultados: A partir de los resultados obtenidos en el capítulo anterior y para dar continuidad a la aplicación de la Ingeniería de Requisitos, en el presente capítulo se aplican métricas y métodos de caja blanca y caja negra para garantizar la calidad de la solución desarrollada. También se exponen los resultados de las pruebas funcionales realizadas a los requisitos acordados con el cliente.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se realiza la fundamentación de la metodología de desarrollo, las herramientas, los lenguajes y otras tecnologías que brindan las TIC, que resultan útiles para el desarrollo de la solución que se propone y de la aplicación como tal. Para ello se tiene en cuenta que la transformación de los procedimientos penales a una aplicación web lleva implícito una serie de conceptos y conocimientos que resultan de vital importancia dominar, como los que se relacionan a continuación.

1.2 Sistema de Tribunales Populares Cubanos

El Sistema de Tribunales Populares Cubanos está constituido por el Tribunal Supremo Popular (en lo adelante TSP), los Tribunales Populares Provinciales (en lo adelante TPP) y los Tribunales Populares Municipales (en lo adelante TMP). El TSP es la institución encargada de ejercer la máxima autoridad judicial en la República de Cuba y sus decisiones en este orden son definitivas. Tiene su sede en Ciudad de La Habana y ejerce jurisdicción en toda la República. A través de su Consejo de Gobierno, ejerce la iniciativa legislativa en materia relativa a la administración de justicia y la potestad reglamentaria, del mismo modo toma decisiones y dicta normas generales de obligado cumplimiento por todos los tribunales.

La estructura del órgano superior de justicia cubano comprende: el Consejo de Gobierno y 6 Salas de Justicia («Tribunal Supremo Popular de la República de Cuba» 2012):

1. Sala de lo Penal
2. Sala de lo Civil y de lo Administrativo
3. Sala de los Delitos contra la Seguridad del Estado
4. Sala de lo Laboral
5. Sala de lo Económico
6. Sala de lo Militar

Las atribuciones y facultades jurisdiccionales del TSP radican en las Salas de Justicia, las que ejercen su jurisdicción en todo el territorio nacional y son competentes para conocer, de acuerdo con lo establecido en las leyes de procedimiento de las diversas materias, de los recursos de Apelación y Casación contra las sentencias dictadas por los TPP. Por otra parte, los TPP ejercen su jurisdicción en el territorio de las correspondientes provincias y tienen sus sedes donde determine el Consejo de Gobierno del TSP.

Tanto para el desarrollo de su actividad jurisdiccional como para las actividades de apoyo a ésta, todos los años el Consejo de Gobierno del Tribunal Supremo Popular aprueba las proyecciones estratégicas y los principales objetivos, por áreas de resultados clave, como son, la actividad jurisdiccional, la ética judicial, la atención a la ciudadanía, los recursos humanos, y el aseguramiento material, financiero y tecnológico.

El cumplimiento de estos objetivos es controlado sistemáticamente por los presidentes de las diferentes salas de justicia de la instancia y ha sido objeto de análisis periódico por el Consejo de Gobierno. Como resultado de la búsqueda permanente de una mayor racionalidad y eficacia en las decisiones judiciales el Tribunal Supremo ha proyectado sus esfuerzos en el desarrollo de la Informática Jurídica en Cuba.

1.3 Informática Jurídica en el Sistema de Informatización de los Tribunales Populares Cubanos

La necesidad imperante de acoplar el desarrollo de los procedimientos judiciales con el auge de las Tecnologías de la Informática y las Comunicaciones (TIC), propició que en nuestro país los tribunales populares comenzaran a prestar especial atención a la Informática Jurídica. Esta constituye un término común en nuestros días, pues es la aplicación de los medios informáticos en la rama del derecho, tanto para la informatización de los procesos judiciales como para el seguimiento y control de los mismos.

Para Emma Riestra la Informática Jurídica (RIESTRA 2011):

"…es la interrelación entre las materias informática y derecho que tiene como fin el análisis, la estructuración lógica y ordenada, la deducción e interpretación de la información jurídica a través de la utilización de la máquina computadora para su efectivo y eficaz tratamiento, administración, recuperación, acceso y control, y cuyos alcances están predeterminados al auxilio en la toma de decisiones jurídicas."

Ramas de la Informática Jurídica:

- **La Informática Jurídica Documental:** Procesamiento automático de documentos jurídicos, proveniente de cualquiera de las fuentes formales del derecho: Legislativa (en sentido amplio), jurisprudencial (producción de los órganos jurisdiccionales, comprendidos los individuales y los colectivos) y doctrinaria (conceptos de los expertos en derecho).
- **La Informática Jurídica de Gestión:** Se refiere a la automatización de procedimientos en las oficinas de los operadores jurídicos. Además de procesar información administrativa, provee herramientas auxiliares para las labores que se desarrollan en tales despachos.

El proyecto para el Sistema de Informatización de los Tribunales Populares Cubanos, del cual es parte esta investigación, centra su desarrollo en la rama de la informática jurídica de gestión. Por las aspiraciones y características del mismo, pues su principal objetivo es informatizar los procedimientos judiciales en cada instancia de los tribunales cubanos y específicamente este trabajo está enmarcado en el Recurso de Casación de la materia Penal.

1.4 Procedimiento Casación de la Materia Penal

Según lo dispuesto en la Ley de procedimiento Penal de 2008 en su Capítulo IV, el recurso de Casación puede interponerse por Infracción de la Ley (en lo adelante IL) y por Quebrantamiento de Forma (en lo adelante QF). Se elevan en casación al Tribunal Supremo Popular las actuaciones de procesos ordinarios recurridos del tribunal provincial (Rivero 2008).

Por generalidad si el recurso es con lugar de QF por los artículos: 70.1, 2, 3, 7,8 se anula la sentencia y/o el juicio. Si el recurso es con lugar de QF por los artículos: 70.4, 5, 6,9 se anula la sentencia. Si el recurso es con lugar por IL se dicta sentencia anulando la sentencia del tribunal de instancia y se dicta una nueva sentencia. Si se declara Mixto con lugar por QF, se dicta sentencia sin entrar a conocer la IL. Si el QF es por los artículos: 70.1, 2, 3, 7,8 se anula la sentencia y el juicio. Si el lugar QF es por los artículos: 70.4, 5, 6,9 se anula la sentencia. Si se declara Mixto sin lugar por QF y con lugar por IL, se dicta sentencia anulando la del tribunal de instancia y se dicta una nueva (Rivero 2008).

La nueva sentencia, no puede ocasionar mayor gravamen que la anulada. Tampoco puede el tribunal de casación, en un recurso del acusado, declarar la casación de oficio, para empeorar su situación. Por lo antes expuesto, este recurso es el medio de impugnación por el cual, una parte postula la revisión de los errores jurídicos atribuidos a la sentencia de mérito que la perjudica, reclamando la correcta aplicación de la ley (Rivero 2008).

1.5 Sistemas de informatización de los procesos en la aplicación de la justicia en el mundo

Aunque el avance tecnológico en la rama del derecho es lento, esto no quiere decir que no existan sistema informáticos para la gestión procesal. Hoy en día se han desarrollado algunas soluciones informáticas para esta ciencia, en aras de agilizar la tramitación procesal. Sin embargo hasta el momento no existe un sistema completamente íntegro que facilite el desarrollo de un proceso judicial.

A continuación se relacionan tres sistemas judiciales que existen a nivel mundial y que presentan características similares al que se quiere desarrollar:

- ✓ **Minerva NOJ (Nueva Oficina Judicial)**

Sistema informático creado en España e incorporado al sistema de justicia español en el 2010.

Principales funcionalidades (Iglesias Canle 2010):

- Registrar, repartir y gestionar documentos.
- Permitir la comunicación entre diferentes órganos.
- Consultar por parte del ciudadano la información sobre el desarrollo del proceso.

Este sistema ha ocasionado el desencanto en el estado de Murcia (donde se puso en práctica), debido a que "la tramitación ha sido interrumpida, la caída del servidor no permite que se realicen las notificaciones telemáticas a los procuradores en los procedimientos penales, donde una omisión en cualquiera de los elementos puede ser devastadora" (Pérez 2010).

Estas precisiones son fundamentales para el desarrollo de un sistema informático, de ahí que para la creación del Sistema de Informatización de los Tribunales (en lo adelante SIT) se tengan en cuenta estos elementos. Sin embargo, el sistema judicial no es igual al que rige en Cuba, de ahí que los módulos automatizados son completamente diferentes en cuanto a término, contenido, órganos que se pueden comunicar y la manera de hacerlo.

✓ **Lexnet**

Sistema informático Español. Principales funcionalidades (Menéndez 2007):

- Presentar escritos y documentos.
- Trasladar copias y realizar actos de comunicación procesal.
- Acceso autenticado y seguro mediante navegador web.
- Permitir la incorporación de otros usuarios como los notarios, registros, graduados sociales, la Abogacía del Estado, letrados de la Seguridad Social, Ministerio Fiscal.

A diferencia del anterior, este "reduce a pocos segundos las operaciones de envío y recepción de notificaciones, recursos, autos o sentencias a las partes de un procedimiento", sin embargo, ha colapsado también, "acumulando notificaciones generadas a lo largo de los días", provocando "pérdida de horas de trabajo". Otra de las quejas generadas es que solamente "se ha podido contar con los conocimientos de un único informático para la solución de los problemas, el sistema estuvo casi una mañana entera paralizado, provocando que se dejara constancia de la declaración de detenidos en un documento Word en el ordenador personal de un juez" (Pérez Cebadera 2011).

Muchos son los fallos detectados y que se deben tener en cuenta a la hora de concebir el SIT, considerando también, que la estructura y organización de los tribunales cubanos no es igual a los de España. Además, costaría mucho a nuestro país lograr el funcionamiento de Lexnet, pues se basa en un sistema de correo electrónico seguro, con firma electrónica, a través del cual el usuario recibe las notificaciones emitidas por el juzgado y presenta los escritos por vía telemática.

✓ **Atlante**

Sistema informático realizado en Islas Canarias. Principales funcionalidades (Consejería de Presidencia, Justicia e Igualdad 2012):

- Registrar las diligencias que se realizan en diferentes instituciones.
- Remitir la información necesaria al siguiente paso.

Este sistema aún no rompe las barreras del papel, elemento fundamental a la hora de automatizar cualquier proceso, que tiene como principal objetivo eliminar la dependencia de este. Ocurre pérdida del servicio y su tiempo de respuesta es lento, acciones primordiales cuando de justicia se trata, pues con la informatización del sector judicial se pretende lograr una justicia pronta y cumplida (Consejería de Presidencia, Justicia e Igualdad 2012).

Luego del estudio de los sistemas similares que existen en el mundo se evidencia que ellos no satisfacen las necesidades de los tribunales cubanos porque solo representan algunas de las funcionalidades que se desean informatizar. Además, los procedimientos a los que responden no son aplicables a las leyes de nuestro país, pero se pueden tener en cuenta los errores cometidos para no incurrir en ellos cuando se desarrolle el SIT.

1.6 Sistemas de informatización de los procesos en la aplicación de la justicia en Cuba

Cuba no deja escapar la oportunidad de informatizar sus procesos judiciales, en dos ocasiones ha realizado el intento, primero la materia penal y luego la económica. Sin embargo no se ha logrado satisfacer ninguna de las expectativas iniciales, en ninguno de los casos.

✓ **Sistema de Control de Procesos Penales (SISPRO)** (Rodríguez 2008):

Sistema informático desarrollado en la provincia de Villa Clara. La propuesta inicial fue que abarcara la instancia suprema y provincial de la materia penal, desarrollándose solamente la tramitación de los procesos en la última instancia.

Deficiencias:

- No supera la barrera del papel.
- No aporta estadística, ni información alguna.
- No capta ningún dato de la fase judicial de la tramitación y decisión del tribunal.
- En los textos y modelos mostrados presenta errores de concordancia, redacción o contenido.
- Programado en Delphi, corre sobre SQL Server por lo que no es compatible con el software libre en el que se están programado los sistemas generales de cada materia judicial.

Este sistema no se puede unir al SIT ya que las tecnologías de desarrollo no se ajustan con las que se están estudiando para desarrollar este último, pues son más avanzadas. Por otro lado el informático que desarrolló esta aplicación no documentó sobre el desarrollo de la misma, por lo que no se puede reutilizar tan fácilmente. Además, al no permitir registrar ningún dato de la tramitación queda por debajo de las expectativas que se tienen con la creación del SIT.

✓ **Sistema de Procesos Económicos (SISECO)** (Rodríguez 2010):

Sistema informático desarrollado en Ciudad de La Habana en el 2002, concebido para el área económica. Es muy sencillo y lento.

Deficiencias:

- Inserta documentos radicados manualmente. La secretaria inserta los datos en la aplicación y cada cinco años borra los datos de los años anteriores quedando solamente la información asentada en los libros.
- Las salvas se guardan en discos.

Al igual que la anterior, esta aplicación no cumplió las expectativas iniciales de su creación. Posee elevados tiempos de respuesta a los principales procesos, cualidad que no va aparejada con la estadística en tiempo real. Prácticamente no informatiza la mayoría de los procedimientos, pues la radicación se realiza de forma manual, las salvas se guardan en discos, que en la actualidad ya no son de uso, la información almacenada es borrada cada cinco años, algo incomprensible para un sistema judicial, ya que la información es el mecanismo principal para la justicia. Por tanto, tampoco se puede pretender unir al sistema SIT este pequeño módulo, que no logra captar, ni guardar los datos.

Al culminar el análisis de los sistemas similares existentes en Cuba se evidencia que ellos no satisfacen las necesidades de los tribunales cubanos en su totalidad porque solo responden a algunas de sus áreas. Además, la tecnología con la que se desarrolló SISPRO no es compatible con la política de software libre y los métodos de salvas que emplea SISECO han quedado obsoletos.

1.7 Necesidad de un sistema para la gestión del Recurso de Casación

Luego del análisis de los sistemas existentes en el mundo y en Cuba y de analizadas sus principales funcionalidades y aspectos negativos, se plantea la necesidad de implementar el módulo que permita la informatización del Recurso de Casación en los TPC, debido a que las aplicaciones extranjeras en su mayoría son propietarios, además están concebidos bajo los estatutos legislativos del país al que pertenecen y resulta muy difícil adecuarlos al sistema cubano. Por otra parte, los sistemas que hasta ahora se han desarrollado en el país están construidos con tecnologías

propietarias, presentan dificultades para gestionar las notificaciones y los documentos. También, muestran problemas en la tramitación y no brindan las funcionalidades asociadas a dicho procedimiento. Es por ello que es necesario estudiar las metodologías de desarrollo de software y las tecnologías con el fin de escoger la más adecuada para el correcto desarrollo de un sistema que responda a las necesidades del cliente.

1.8 Modelo de desarrollo de software

Una parte importante de la ingeniería de software es el desarrollo de modelos y metodologías. La experiencia ha demostrado que los proyectos exitosos son aquellos que son administrados siguiendo una serie de procesos que permiten organizar y luego controlar el proyecto, considerando válido destacar que aquellos procesos que no sigan estos lineamientos corren un alto riesgo de fracasar. Es necesario destacar la importancia de los métodos, pero el éxito del proyecto depende más de la comunicación efectiva con los interesados, el manejo de las expectativas y las personas que participan en el proyecto (Valdéz et al. 2014).

El objetivo principal que persigue la ingeniería de software es convertir el desarrollo del mismo en un proceso formal, con resultados predecibles, que permitan obtener un producto final de alta calidad y que satisfaga las necesidades y expectativas del cliente. De esta forma se evidencia la necesidad de definir en un proyecto un modelo de desarrollo que guíe el proceso.

1.8.1 Integración de Modelos de Madurez de Capacidades

Integración de Modelos de Madurez de Capacidades (CMMI por sus siglas en inglés) 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 (Blanco y Aguero 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 *Business Process Modeling Notation* (BPMN por sus siglas en inglés).

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 durante la fase de Requisitos 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 Industrial de Pruebas de Software 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.

En el ciclo de vida pueden realizarse iteraciones según las necesidades del proyecto a partir de la fase de negocio, en este caso se adecúan a las características del proyecto Tribunales del centro CEGEL de la Facultad 3 y en esta investigación se pretende llegar hasta la fase de pruebas internas.

Teniendo en cuenta lo antes expuesto, los productos de trabajo que se obtendrán en esta investigación debido a la ejecución de las actividades, se desarrollan acorde a lo definido en la metodología que se describe a continuación.

1.8.2 Proceso Unificado de Software (Rational Unified Process)

Para la construcción de un software de alta calidad desarrollado en el tiempo planificado y con los costos establecidos, se necesita trabajar de forma organizada de manera que se controle y documente todo lo relacionado con el proyecto en cuestión, esto conlleva a eliminar algunos de los riesgos presentados durante el desarrollo del mismo. “Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos de software” (Menéndez y Asensio 2013).

Rational Unified Process (en lo adelante RUP, por sus siglas en inglés) es un proceso de ingeniería de software bien definido y estructurado, a la vez que es un producto que provee un marco de proceso adaptable a las necesidades y características de cada proyecto específico. Los autores de RUP destacan que el proceso de software tiene tres características esenciales (RUP 2011):

- **Dirigido por casos de uso:** los casos de uso representan los requisitos funcionales, y todos los casos de uso juntos forman el modelo de casos de uso el cual describe la funcionalidad total del sistema y guían el proceso de desarrollo, ya sea el diseño, la implementación o las pruebas.
- **Centrado en la Arquitectura:** donde dicha arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema. RUP se desarrolla mediante iteraciones, comenzando por los CU relevantes desde el punto de vista de la arquitectura.
- **Iterativo e Incremental:** RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. Cada iteración se realiza de forma planificada.

1.8.3 Adaptación para el desarrollo de la aplicación

Se decidió trabajar con RUP, ya que es una metodología robusta que describe los elementos del modelo que son más importantes para su construcción, pues se desarrolla mediante iteraciones, comenzando por los casos de uso relevantes desde el punto de vista de la arquitectura. Por las características del proyecto el cliente no siempre se encuentra presente y es de gran importancia la extensa documentación que ofrece esta metodología, lo que garantiza poder brindarle al usuario una visión clara de lo que se está realizando en cada etapa del desarrollo. También, se apoya en UML, lo que permite realizar el modelado de los casos de uso que representan los requisitos funcionales, y obtener el modelo de casos de uso que describe la funcionalidad total del sistema y guía el proceso de desarrollo, ya sea el diseño, la implementación o las pruebas.

A partir de lo antes expuesto y teniendo en cuenta el Programa de Mejoras de la UCI y el expediente del proyecto de Informatización de los TPC, que integra las fases del ciclo básico de CMMI con la metodología RUP, se pretende desarrollar el módulo del Recurso de Casación aplicando además, la Ingeniería de Requisitos que se presenta a continuación.

1.8.4 Ingeniería de Requisitos

La Ingeriría de Requisitos (en lo adelante IR) ayuda a los ingenieros de software a entender mejor el problema en cuya solución trabajarán. Incluye el conjunto de tareas que conducen a comprender cuál será el impacto del software sobre el negocio, qué es lo que el cliente quiere y cómo interactuarán los usuarios finales con el software (Chaves 2011).

Con el propósito de garantizar un producto de calidad que esté basado en las necesidades del cliente, se describen a continuación las actividades de la IR: Elicitación, Análisis, Especificación, Validación y Gestión de requisitos, las cuales se aplican en este trabajo teniendo en cuenta lo que plantea el Plan de desarrollo de la IR del proyecto de Informatización de los TPC.

- **Elicitación de requisitos:** Esta etapa abarca la primera y quizás más importante fase dentro del desarrollo de un sistema informático. Uno de sus retos más importantes es garantizar que los requisitos del sistema sean consistentes con las necesidades de la organización.

En esta actividad se realiza el modelado del negocio por procesos y no por casos de uso como está definido en RUP, cumpliendo con lo definido en el Plan de desarrollo de la IR del proyecto de Informatización de los TPC que se rige por el Programa de Mejoras.

En la elicitación se realizan las siguientes tareas:

- Arqueología de documentos: Se estudia la documentación que se genera en el proceso: escritos judiciales, resoluciones y sentencias.

- Entrevistas: Los requisitos son identificados muchas veces mediante respuestas que emiten los jueces civilistas a preguntas realizadas en entrevistas, que permiten obtener una información general de lo que hace el cliente y de cómo podría interactuar con el sistema. Se utiliza también para identificar los requisitos no funcionales del sistema.
- Identificación de las reglas del negocio: Se realiza en conjunto con el cliente donde se recogen en un documento las normas y políticas por las que se rige la organización.

Los artefactos que se generan son:

- Descripción de procesos de negocio.
- Reglas del negocio.
- Especificación de requisitos de software.

- **Análisis de requisitos:** Una vez recopilados los requisitos, el producto obtenido configura la base del análisis de requisitos. Los requisitos se agrupan por categorías y se organizan en subconjuntos, se estudia cada requisito en relación con el resto, se examinan los requisitos en su consistencia, completitud y ambigüedad, y se clasifican en base a las necesidades de los clientes/usuarios.

Tareas que se realizan:

- Analizar requisitos del cliente: Se identifican ambigüedades, inconsistencias, requisitos comunes y escenarios arquitectónicos requeridos para dar soporte a los requisitos funcionales.
- Talleres de análisis: El propósito de estos talleres es analizar los requisitos para identificar otros o modificar los existentes.
- Reuniones con el cliente: En estas se analizan los requisitos identificados con el propósito de aprobarlos luego de llegar a un acuerdo con el cliente.

El artefacto que se actualiza durante esta actividad es:

- Especificación de requisitos de software.

- **Especificación de requisitos:** Una especificación puede ser un documento escrito, un modelo gráfico, un modelo matemático formal, una colección de escenarios de uso, un prototipo o una combinación de lo anteriormente citado.

Tareas que se realizan:

- Describir los casos de uso identificados a partir de los requisitos funcionales.
- Evaluar los casos de uso según su complejidad y prioridad.
- Elaborar prototipos de interfaz de usuario.

Los artefactos que se generan son:

- Especificación de requisitos de software (actualizado).
- Especificación de casos de uso.
- Prototipos de interfaz de usuario.

- **Validación de requisitos:** La calidad de los productos de trabajo procedentes de la IR se evalúa durante un paso de validación. La validación de requisitos examina la especificación para asegurar que todos los requisitos de software se han establecido de manera precisa; que se han detectado inconsistencias, omisiones y errores y que éstos han sido corregidos, y que los productos de trabajo cumplen con los estándares establecidos para el proceso, proyecto y producto.

Durante esta actividad se aplican métricas de software como:

- Métricas de la calidad de la especificación de requisitos.
- Modelo de métricas orientadas a objeto aplicadas al diagrama de casos de uso.

Los artefactos que se generan son:

- Acta de validación de la especificación de casos de uso.
- Acta de validación de la especificación de requisitos.

- **Gestión de requisitos:** Es el proceso de comprender y controlar los cambios en los requisitos del sistema.

Para gestionar los requisitos se llevan a cabo “un conjunto de actividades que ayudan al equipo de trabajo a identificar, controlar y seguir los requisitos y los cambios en cualquier momento” (García Ramírez y Puello Marrugo 2011). Es importante señalar que a pesar de ser la quinta actividad de la IR tiene lugar en todas las anteriores actividades.

1.8.5 Herramienta para el modelado

Las herramientas de Ingeniería de Software Asistida por Computadora (CASE por sus siglas en inglés) son las aplicaciones que participan en el desarrollo de programas informáticos, facilitando la planificación, el análisis y diseño, y hasta la generación del código fuente de los programas así como su documentación. El uso de estas herramientas tiene como base la búsqueda de un aumento de la productividad en el desarrollo de software reduciendo el costo de las mismas tanto en términos de tiempo como de dinero.

1.8.5.1 Visual Paradigm v8.0

Visual Paradigm es una herramienta CASE que utiliza como lenguajes, UML para el modelado del sistema y la notación BPMN para el negocio, soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El lenguaje de modelado UML ayuda a una construcción más rápida de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación (Paradigm 2010).

Principales características (Paradigm 2010):

- Disponibilidad en múltiples plataformas (Windows, Linux).
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.

- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Soporta aplicaciones web.
- Fácil de instalar y actualizar.
- Generación de bases de datos.

Para el modelado del SIT se utiliza como herramienta CASE el Visual Paradigm (suite 5.0 en su versión 8.0) debido a que soporta el ciclo de vida del software definido por CMMI y puede integrarse con UML y BPMN que serán utilizados para el desarrollo del sistema a realizar. Además, permite la elaboración de diagramas de casos de uso para dirigir el desarrollo, lo que corresponde a una de las características de RUP.

1.8.6 Notación y lenguaje para el modelado

Durante el desarrollo de un software se captan características o requisitos que deben ser traducidos en especificaciones del sistema para que puedan ser implementadas posteriormente por los desarrolladores. En función de esto es necesario emplear una notación para el modelado del negocio y un lenguaje para el modelado del sistema.

1.8.6.1 Notación para Modelado de Procesos de Negocio

La Notación para el Modelado de Procesos de Negocio (BPMN por sus siglas en inglés) permite que se expresen los procesos de negocio en un diagrama. Estos son interpretados fácilmente por las personas que hagan usos de ellos. De esta forma se puede definir, diseñar y generar una solución al proceso objeto de esta investigación.

Características («BPMNbyExampleSPA» 2014):

- Proporciona un lenguaje gráfico común, con el fin de facilitar su comprensión a los usuarios de negocios.
- Se adapta rápidamente a los cambios y oportunidades del negocio.
- Combina las capacidades del software y la experiencia de negocio para optimizar los procesos y facilitar la innovación del negocio.
- Crea un puente estandarizado para disminuir la brecha entre los procesos de negocio y la implementación de estos.

Esta notación es utilizada en este trabajo para la confección del diagrama de procesos de negocio del Recurso de Casación.

1.8.6.2 Lenguaje Unificado de Modelado

Lenguaje Unificado de Modelado (UML por sus siglas en inglés) se centra en la representación gráfica de un sistema. Este lenguaje indica cómo crear y leer los modelos. Esto último es el objetivo

de las metodologías de desarrollo. UML sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas software como para la arquitectura hardware donde se ejecuten (Orallo 2010).

Las funciones de UML son:

- Visualizar: Permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- Especificar: Permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

Se decide utilizar UML, además de estar definido en el Plan de desarrollo de la Ingeniería de Requisitos del proyecto de Informatización de los Tribunales Populares Cubanos, para la elaboración de los diagramas. Además por ser un lenguaje del que se tiene gran dominio por parte del equipo de desarrollo, reduciendo con ello los costos de tiempo que deben emplearse en la capacitación para el empleo de otro lenguaje.

1.9 Tecnologías de desarrollo

En este epígrafe se abordarán las principales tecnologías que se utilizarán para el desarrollo de la aplicación, haciendo énfasis en los marcos de trabajo, los lenguajes de programación, el servidor web, el sistema gestor de base de datos y el mapeo de objetos.

1.9.1 Marco de trabajo

El término marco de trabajo se traduce en una estructura conceptual y tecnológica de soporte definida, normalmente, con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado. Es propio de un marco de trabajo incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para facilitarle el desarrollo y unir los diferentes componentes de un proyecto.

En otras palabras, un marco de trabajo se puede considerar como una aplicación genérica incompleta y configurable a la que se le puede añadir las últimas piezas para construir una aplicación concreta. Los objetivos principales que persigue un marco de trabajo son: acelerar el proceso de desarrollo, reutilizar el código ya existente y promover buenas prácticas de desarrollo con el uso de patrones. Un marco de trabajo Web, por tanto, se puede definir como un conjunto de componentes que forman un diseño reutilizable que facilita y agiliza el desarrollo de sistemas Web (Gutiérrez 2011).

1.9.1.1 Symfony2

Symfony2 es un marco de trabajo PHP que ofrece soluciones a los retos de la programación web como son la seguridad, la persistencia de los datos, el enrutamiento y la validación; utiliza librerías, componentes y *bundles*. Además, tanto la configuración de los archivos, como las plantillas y el almacenamiento se puede realizar en distintos formatos que pueden ser comprendidos por programadores tanto expertos como novatos. Por tanto es versátil, flexible, útil, ofrece muy buen rendimiento y aplica buenas prácticas (Torres, Preval y Riquenes 2012).

Características:

- Twig es el único motor de plantillas activado.
- Doctrine ya está activado tanto en su ORM como en su DBAL (Acceso a Base de Datos).
- Las anotaciones están activadas para todos los casos (rutas, validación, entidades).
- Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en el proyecto que se desarrolla.

Ventajas:

- Permite la autenticación de usuarios con contraseña, la autorización segmentada del sitio web, la opción "*remember me*" en los formularios de autenticación, las listas de control de acceso o ACL, por sus siglas en inglés, la integración de los usuarios con las entidades de Doctrine2.
- Permite configurar un entorno de ejecución lo más parecido posible al entorno de producción en el que se ejecutará la aplicación real.

Symfony2 recomienda utilizar Twig para crear todas las plantillas de la aplicación porque es un motor y lenguaje de plantillas para PHP muy rápido y eficiente, que admite la herencia entre plantillas con la creación de un "esqueleto" de plantilla base que contenga todos los elementos comunes del sitio y define los bloques que las plantillas descendientes pueden sustituir. Además, se pueden crear nuevos bloques (Torres, Preval y Riquenes 2012).
Se pretende utilizar en la investigación este marco de trabajo que según su creador Fabien Potencier, no es Modelo-Vista-Controlador (MVC) porque sólo proporciona herramientas para la parte del Controlador y de la Vista. La parte del modelo es responsabilidad de los implementadores, aunque existen bibliotecas para integrar fácilmente los ORM más conocidos, como Doctrine en el caso de estudio.

1.9.1.2 Twitter Bootstrap v2.1

Es una colección de herramientas de software libre para la creación de sitios y aplicaciones web. Bootstrap fue desarrollado como un marco de trabajo y fomenta la consistencia a través de herramientas internas. Se emplea en este trabajo debido a sus características.

Características (Otto et al. 2011):

- Tiene un soporte relativamente incompleto para HTML5 y CSS3, pero es compatible con la mayoría de los navegadores web. La información básica de compatibilidad de sitios web o aplicaciones está disponible para todos los dispositivos y navegadores.
- Desde la versión 2.0 también soporta diseños sensibles, o sea, el diseño gráfico de la página se ajusta dinámicamente, tomando en cuenta las características del dispositivo usado (Computadoras, tabletas, teléfonos móviles).
- Los desarrolladores pueden adaptar el mismo archivo de Bootstrap, seleccionando los componentes que deseen usar en su proyecto. Los ajustes son posibles en una medida limitada a través de una hoja de estilo de configuración central. Los cambios más profundos son posibles mediante las declaraciones LESS. El uso del lenguaje de hojas de estilo LESS permite el uso de variables, funciones, operadores y selectores anidados.

La configuración de Bootstrap también tiene una opción especial de "Personalizar" en la documentación. Por otra parte, los desarrolladores eligen en un formulario los componentes y ajustes deseados, y de ser necesario, los valores de varias opciones a sus necesidades.

1.9.1.3 Biblioteca jQuery v1.8

jQuery es una biblioteca de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privativos. jQuery, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio. También integra funcionalidades para trabajar con *Asynchronous JavaScript And XML* (AJAX por sus siglas en inglés).

Otras características (Resig y others 2009):

- Selección de elementos DOM (Modelo de Objetos del Documento).
- Interactividad y modificaciones del árbol DOM, incluyendo soporte para CSS 3.
- Compatible con los navegadores Mozilla Firefox 2.0+.

La característica principal de la biblioteca es que permite cambiar el contenido de una página web sin necesidad de recargarla, mediante la manipulación del árbol DOM y peticiones AJAX, por lo que se emplea en el desarrollo del módulo del Recurso de Casación.

1.9.2 Lenguajes de programación

Un lenguaje de programación es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Puede usarse para crear

programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. A continuación se describen los lenguajes que se van a usar durante el desarrollo de la investigación.

1.9.2.1 PHP v5.3

Los lenguajes del lado del servidor son aquellos lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él. PHP, acrónimo de *Hipertexto Pre-Processor*, es un lenguaje de programación que se interpreta por un servidor web. Permite crear un programa que se pueda ejecutar en el servidor desde un programa visualizador de páginas web y dar respuestas en función de los datos que se introduzcan por el usuario (Vargas y Guevara 2013).

Ventajas fundamentales (Heurtel 2011):

- El código se encuentra protegido tanto de la manipulación de los usuarios como de la presencia de virus.
- Se caracteriza por su facilidad de aprendizaje en breve tiempo.
- Es multiplataforma y no requiere de recursos desmesurados para que funcione.

1.9.2.2 JavaScript

JavaScript es un lenguaje de programación interpretado y se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas (Flanagan 2002).

Es un lenguaje de alto nivel, multiplataforma y no necesita compilación. Está basado en objetos y maneja la mayoría de los eventos que se pueden producir sobre la página web. La mayoría de los navegadores en sus últimas versiones interpretan el código JavaScript integrado dentro de las páginas web (Flanagan 2002).

1.9.2.3 HTML 5

HTML (HyperText Markup Language, versión 5) es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML.

HTML5 establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos. Algunos de ellos son técnicamente similares a las etiquetas y , pero tienen un significado semántico, como por ejemplo <nav> (bloque de navegación del sitio web) y <footer>. Otros elementos proporcionan nuevas funcionalidades a través de una interfaz

estandarizada, como los elementos <audio> y <video>.También hay un renovado énfasis en la importancia del scripting DOM para el comportamiento de la web. 2.0

Novedades (Franganillo 2011):

- Etiquetas para manejar grandes conjuntos de datos: Datagrid, Details, Menu y Command. Permiten generar tablas dinámicas que pueden filtrar, ordenar y ocultar contenido en cliente.
- Mejoras en los formularios. Nuevos tipos de datos (eMail, number, url, datetime) y facilidades para validar el contenido sin Javascript.

1.9.3 Sistema Gestor de Bases de Datos

Un Sistema Gestor de Bases de Datos (SGBD) es un software o conjunto de programas que permiten crear y mantener una base de datos. El SGBD actúa como interfaz entre los programas de aplicación (usuarios) y el sistema operativo. El objetivo fundamental de un SGBD es proporcionar un entorno eficiente a la hora de almacenar y recuperar la información de la base de datos (Vera y Sánchez 2004).

1.9.3.1 PostgreSQL v9.3

Se elige PostgreSQL porque es un potente sistema de gestión de bases de datos, multiusuario, centralizado y de propósito general, que está liberado bajo la licencia Berkeley Software Distribution (BSD). Tiene interfaces de programación con diferentes lenguajes como PHP, C, C++, Java, .NET, Python y Perl. Además brinda una amplia documentación lo que hace más fácil trabajar con el mismo. Cumple con las propiedades ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad) y soporta el lenguaje común de acceso a los datos: SQL.

Características fundamentales (Group y others 2011):

- Presenta claves ajenas también denominadas llaves ajenas o claves foráneas en inglés (foreign keys).
- Tiene integridad transaccional.
- Posee herencia de tablas.
- Cuenta con tipos de datos y operaciones geométricas.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.

1.9.4 Mapeo de objeto-relacional

Object-Relational Mapping (en lo adelante ORM, por sus siglas en inglés), o lo que es lo mismo, mapeo de objeto-relacional, es un modelo de programación que consiste en la transformación de

las tablas de una base de datos, en una serie de entidades que simplifiquen las tareas básicas de acceso a los datos para el programador, es decir, lo abstrae de la base de datos y lo centra en el desarrollo de la aplicación.

1.9.4.1 Doctrine v2.0

Se elige doctrine porque es un ORM escrito en PHP que proporciona persistencia para objetos PHP y se encuentra activado en Symfony 2. Está por encima de la capa de abstracción a la base de datos, una de sus características es la posibilidad de escribir consultas a la base de datos a partir del tratamiento con objetos en PHP llamado *Doctrine Query Language* (DQL por sus siglas en inglés).

Doctrine además, puede generar clases a partir de una base de datos creada, y el programador puede especificar relaciones y agregar funcionalidades comunes para las clases generadas (FERNÁNDEZ 2012).

1.9.5 Servidor web

Un servidor web es un programa informático que procesa aplicaciones realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente, generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente. Es quien gestiona la mayor parte de las funciones de lógica de negocio y de acceso a los datos de la aplicación. Los principales beneficios de la aplicación de la tecnología de servidores web son la centralización y la disminución de la complejidad en el desarrollo de aplicaciones.

1.9.5.1 Servidor HTTP Apache v2.2

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

Se eligió Apache como software para el servidor de aplicaciones en su versión 2.2, por su flexibilidad, rapidez, porque es gratuito, multiplataforma e interpreta varios lenguajes como PHP, características estas que promueven la eficiencia y rendimiento del sistema que debe ser capaz de dar respuestas a las peticiones con un nivel aceptable de desempeño, teniendo en cuenta la concurrencia que pueda existir, debe prestar servicios sin que se amplíen los rangos de tiempo de repuesta (Esteban y Alejandro 2007).

1.9.6 Control de versiones

El control de versiones involucra procedimientos y herramientas para gestionar los cambios en los elementos creados durante el ciclo de vida del software. Se refiere a la gestión de los cambios que se realizan sobre los elementos de algún producto o sobre la configuración del mismo. Los sistemas

de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas. El principal objetivo es permitir editar de forma colaborativa y compartir información. Aunque un sistema de control de versiones puede realizarse de forma manual, es muy aconsejable disponer de herramientas que faciliten esta gestión (Ruiz-Bertol y Zarazaga-Soria 2007).

1.9.6.1 SubVersion v1.5

SubVersion (SVN) es un sistema de control de versiones usado para que varios desarrolladores puedan trabajar en un mismo proyecto de forma ordenada. Tiene una arquitectura cliente servidor con controles de concurrencia para cuando varios desarrolladores estén trabajando en el mismo archivo, por lo que se utiliza en este trabajo.

Características (Vallejo 2008):

- Mantiene versiones no sólo de archivos, sino también de directorios y los metadatos asociados a ellos.
- Uso eficiente del ancho de banda, ya que en las transacciones se transmiten sólo las diferencias y no los archivos completos.
- Además de los cambios en el contenido de los documentos, se mantiene el historial de todas las operaciones de cada elemento, incluyendo la copia, cambio de directorio o de nombre.
- Atomicidad de las actualizaciones. Una lista de cambios constituye una única transacción o actualización del repositorio. Esta característica minimiza el riesgo de que aparezcan inconsistencias entre distintas partes del repositorio.

1.10 Arquitectura de software

La Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema, programa o aplicación y tiene la responsabilidad de: definir los módulos principales, las responsabilidades que tendrá cada uno de estos módulos y la interacción que existirá entre dichos módulos. Aporta una visión abstracta de alto nivel, posponiendo el detalle de cada uno de los módulos definidos a pasos posteriores del diseño. Según la IEEE Estándar 1471-2000: “la arquitectura del software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución (Casanovas 2004).

1.10.1 Estilo arquitectónico en capas

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

un conjunto de conexiones entre dichos componentes y un conjunto de restricciones sobre cómo se comunican dos componentes cualesquiera conectados (de la Torre Llorente et al. 2010).

Dichos estilos arquitectónicos constituyen herramientas básicas a la hora de definir la estructura de un sistema, también tienen una serie de directivas para organizar los componentes del mismo con el objetivo de facilitar la tarea del diseño de la aplicación. Indican también cómo se va a dividir en partes y cómo será la interacción entre estas.

Arquitectura en Capas

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

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

La arquitectura en capas provee un buen número de ventajas, algunas de ellas se especifican a continuación (de la Torre Llorente et al. 2010):

- Facilita la reutilización de código y la estandarización.
- Permite mejorar la flexibilidad del sistema.
- Facilita el soporte y mantenimiento a las aplicaciones.
- Permite el desarrollo en paralelo.

1.10.2 Patrón Modelo-Vista-Controlador en Symfony.

Aunque Symfony no es un marco de trabajo Modelo-Vista-Controlador (en lo adelante MVC), proporciona herramientas para la parte del Controlador y de la Vista, dejando el Modelo como responsabilidad de los programadores. El patrón MVC está formado por tres niveles que se explican a continuación (Torres, Preval y Riquenes 2012):

- **Modelo:** esta capa es la encargada de administrar todo lo relacionado con los datos del sistema, brinda respuesta a las peticiones de información sobre el estado de la aplicación y responde a instrucciones de cambiar de estado. Representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- **Vista:** aquí se maneja la visualización de la información, es decir, gestiona lo relacionado con mostrar la información al usuario.

- **Controlador:** en esta capa se interpretan los eventos lanzados por la entrada del usuario, informando de los mismos al modelo y vista para que los cambios sean ejecutados correctamente.

Con la aplicación del MVC se consigue separar los datos de la aplicación, de la interfaz de usurario y la lógica del negocio, por lo que se logra un mejor mantenimiento del sistema, permitiendo con esto que si la plataforma necesita ser ejecutada en un navegador estándar y un navegador de dispositivo móvil solo se deben crear dos interfaces y mantener la controladora, con las funcionalidades para ambos dispositivos.

1.11 Patrones de diseño

Los patrones de diseño proporcionan una estructura conocida por todos los desarrolladores, de forma que la manera de trabajar no resulte distinta entre ellos. Si se incorpora un nuevo programador, no necesitará mucho conocimiento de lo realizado anteriormente por los otros. Permiten tener una estructura de código común a todos los proyectos que implemente una funcionalidad genérica. La utilización de patrones de diseño, permite ahorrar grandes cantidades de tiempo en la construcción de software. El producto obtenido es más fácil de comprender, mantener y extender (Juan 2000).

Los patrones para casos de uso representan comportamientos que deben existir en el sistema, ayudan a describir qué es lo que este debe hacer, es decir, describen el uso del sistema y cómo el mismo interactúa con los usuarios. En esta investigación se emplea el que se describe seguidamente (Övergaard, Gunnar y Palmqvist, Karin 2004):

- **Actores múltiples:** Es un patrón de estructura que plantea que cuando dos actores juegan el mismo papel hacia un caso de uso se representa otro actor, del que heredan los actores que comparten este rol.
- **Inclusión:** Se incluye una relación del caso de uso base al caso de uso de inclusión. Se relacionan dos casos de uso con un “include” cuando el primero (el caso de uso base) incluye al segundo (el caso de uso incluído). Es decir, el segundo es parte esencial del primero. Sin el segundo, el primero no podría funcionar bien; pues no podría cumplir su objetivo.
- **Extensión:** Consiste en dos casos de uso y una relación extendida entre ellos. Un caso de uso extiende a otro cuando sin alterar a este, se incorpora su funcionalidad como parte integral del primero. El caso de uso de extensión no es indispensable que ocurra, y cuando lo hace ofrece un valor extra (extiende) al objetivo original del caso de uso base.

Los Patrones Generales de Software para Asignación de Responsabilidades (GRASP por sus siglas en inglés) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. De este grupo se utilizan en la investigación los siguientes patrones (Craig 2003):

- **Experto:** es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Con su utilización se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil manteniendo.
- **Creador**: este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos como el sistema que se desea implementar. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte también al bajo acoplamiento lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización.
- **Bajo acoplamiento:** es un patrón que soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad. El bajo acoplamiento no puede verse separado de otros patrones como el experto y la alta cohesión.
- **Alta Cohesión:** con este patrón se espera que una clase tenga un número moderado de responsabilidades dentro de un área funcional y colabore con las otras para llevar a cabo una tarea.
- **Controlador:** En las clases controladoras este patrón sirve de mediador entre las interfaces y el algoritmo que la implementa. Se encarga de recibir los datos que el usuario introduce a la aplicación y enviarlos a las distintas clases de acuerdo a la tarea que se desea realizar. Su funcionamiento está basado en que la lógica de negocio debe estar separada de la capa de la vista y así aumentar la reutilización de código y tener un mayor control sobre los cambios.

Los patrones GOF[1] que se emplean en este trabajo son (Guerrero, Suárez y Gutiérrez 2013):

- **Decorador:** Este patrón 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.
- **Factory method:** Permite escribir aplicaciones que son más flexibles respecto de los tipos a utilizar difiriendo la creación de las instancias en el sistema a subclases que pueden ser extendidas a medida que evoluciona el sistema. Permite también encapsular el conocimiento referente a la creación de objetos. Define una interfaz para crear objetos pero deja que sean las subclases las que deciden qué clases instanciar.

[1] Grupo de los Cuatro (GOF por sus siglas en inglés)

1.12 Métricas de software

Las métricas son la maduración de una disciplina, que van a ayudar a la evaluación de los modelos de análisis y de diseño, en donde proporcionarán una indicación de la complejidad de diseños procedimentales y de código fuente, y ayudarán en el diseño de pruebas más efectivas (Pressman 1998).

Por tanto, la aplicación de métricas durante el análisis y el diseño permiten medir de forma cuantitativa ciertos atributos de calidad internos del software, para expresar una valoración sobre la calidad de la solución. A continuación se exponen las descripciones de las que se aplican en esta investigación:

- **Métricas de la calidad de la Especificación de requisitos**: Se emplean con el objetivo de determinar la especificidad de los requisitos (ausencia de ambigüedad). Esta se basa en la consistencia de la interpretación de los revisores para cada requisito. El equipo de revisores debe estar integrado por especialistas funcionales (no menos del 50%), analistas de sistemas y programadores (Sakipova 2011).
    - **Validación de requisitos mediante prototipos:** Se presentan los prototipos elaborados durante la especificación de requisitos a grupos especializados en los procesos, a fin de validar si el análisis realizado responde a las necesidades y aspiraciones del cliente. Para ello, se desarrollan varios escenarios posibles con el auxilio de juegos de datos, de forma tal que se visualicen las diferentes funcionalidades que tendrá el sistema. Luego se documentan y corrigen las no conformidades.
- **Modelo de métricas orientadas a objeto aplicadas al diagrama de casos de uso:** Se aplican con el objetivo de medir la calidad de la funcionalidad de este diagrama. Se consideran cuatro atributos: completitud, consistencia, correctitud y complejidad; los cuales cuentan con un conjunto de factores que tienen asociados una o más métricas, que establecen una medida cuantitativa del grado en que los factores indiquen una mala calidad (Sakipova 2011).
    - Completitud: Grado en que se ha logrado detallar todos los casos de uso relevantes.
    - Consistencia: Grado en que los casos de uso del sistema describen las interacciones adecuadas entre el usuario y el sistema.
    - Correctitud: Grado en que las interacciones actor-sistema soportan adecuadamente el proceso del negocio.
    - Complejidad: Grado de claridad en la presentación de los elementos que describen el contexto y la claridad del sistema.
- **Tamaño Operacional de las Clases (TOC):** Está dada por el número de operaciones asignados a una clase, donde un resultado alto de su aplicación indica que la clase posee bastante responsabilidad, implicando un bajo nivel de reutilización de la clase y complicando

la implementación y la realización de las pruebas. Los atributos de calidad que evalúa esta métrica son los siguientes (Grey y Viltres 2014):

- Responsabilidad: Un aumento del TOC está ligado a un incremento de la responsabilidad asignada a una clase.
- Complejidad de las pruebas: Un aumento del TOC indica un incremento de la complejidad de implementación de una clase y por tanto complejiza las pruebas.
- Reutilización: Un incremento del TOC implica una disminución del grado de reutilización de una clase.

1.13 Métodos de prueba

La metodología RUP propone dos métodos para la validación del software, estos son: las pruebas de caja blanca y las pruebas de caja negra. A continuación se describen ambos métodos:

✓ **Las pruebas de caja negra:** se llevan a cabo sobre la interfaz del software. El objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene (no se ve el código) (Jacobson, Booch y Rumbaugh 2000).

Un ejemplo de estas pruebas es el diseño de casos de prueba, donde los casos de prueba constituyen una guía principal para el probador del sistema, incluyendo los datos de entrada y resultados esperados. Son necesarios para verificar la aplicación exitosa y aceptable de los requisitos del producto.

✓ **La prueba de la caja blanca:** este método requiere del conocimiento de la estructura interna del programa y se deriva a partir de las especificaciones internas de diseño o el código, se comprueban los caminos lógicos del software proponiendo casos de prueba que se ejerciten sobre conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado o mencionado (sobre el código) (Jacobson, Booch y Rumbaugh 2000).

Se parte de los requisitos funcionales, a muy alto nivel, para diseñar pruebas que se aplican sobre el sistema sin necesidad de conocer cómo está construido por dentro (caja negra). Las pruebas se aplican sobre el sistema empleando un determinado conjunto de datos de entrada y observando las salidas que se producen para determinar si la función se está desempeñando correctamente por el sistema bajo prueba. Las herramientas básicas son observar la funcionalidad y contrastar con la especificación.

Es decir, conociendo el código y siguiendo su estructura lógica, se pueden diseñar pruebas destinadas a comprobar que el código hace correctamente lo que el diseño de bajo nivel indica y otras que demuestren que no se comporta adecuadamente ante determinadas situaciones.

Algunas técnicas de prueba de caja blanca son (Ruiz Tenorio 2010):

- Prueba de bucles: Es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles.
- Prueba del camino básico: Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar la misma como guía para la definición de un conjunto básico. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática.

La prueba de caja negra no es una alternativa a las técnicas de prueba de la caja blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la caja blanca.

Con este tipo de pruebas se intenta encontrar:

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

A partir del estudio realizado se pretenden aplicar las pruebas de caja negra utilizando la técnica del camino básico y las de caja blanca con particiones equivalentes.

1.14 Conclusiones parciales

El estudio realizado en este capítulo permitió determinar lo siguiente:

- El análisis del estado de la informática jurídica en el mundo y en Cuba permitió demostrar la necesidad de implementación de esta ciencia en el sector jurídico cubano y específicamente en el procedimiento del Recurso de Casación de la materia Penal.
- El análisis del modelo de desarrollo del SITPC y las tecnologías a usar para la ejecución del proyecto permitió sentar las bases, desde el punto de vista técnico, para iniciar el desarrollo del módulo Casación del subsistema Penal.

Capítulo 2: Propuesta de solución

2.1 Introducción

A partir del estudio realizado en el capítulo 1 y luego de haber analizado los lenguajes y herramientas que se utilizarán en el curso de la investigación, se propondrá una solución técnica. En el presente capítulo se brinda un mejor entendimiento del Recurso de Casación, se identificarán los participantes y las actividades que se van a informatizar, obteniéndose al final de este proceso los siguientes artefactos: Descripción de procesos de negocio, Reglas del negocio, Especificación de requisitos de software, Especificación de casos de uso del sistema y otros artefactos definidos en el Plan de desarrollo del proyecto Informatización de los TPC.

2.2 Flujo actual del proceso

Los Tribunales Populares Cubanos están divididos en tres instancias: Supremo, Provincial y Municipal, donde son atendidas cinco materias: Laboral, Penal, Civil, Administrativo y Económico. La materia Penal está conformada por varios procesos, uno de ellos es el Recurso de Casación el cual tiene lugar en la instancia suprema. Este proceso inicia cuando se le hace llegar al Tribunal Supremo la causa, interponiendo un recurso de casación, que es la impugnación presentada por la parte, fiscal o acusado, contra las sentencias definitivas dictadas según lo que regula la Ley, en primera instancia, por las salas correspondientes del Tribunal Supremo y por las salas de los Tribunales Provinciales.

El recurso de casación puede interponerse por infracción de ley, quebrantamiento de forma, o mixto y puede acogerse de oficio. Una vez recibida la causa en el Tribunal Supremo, se radica, se confecciona el rollo y se turna al ponente mediante providencia. Si el recurso es admitido el ponente este dicta la sentencia luego de colegiar la decisión, realizando la vista si es necesario. Todos los casos se terminan mediante una resolución ya sea auto, providencia o sentencia. Una vez terminado el proceso corresponde a las secretarias mecanografiar la resolución que se dicta, registrar las sentencias y confeccionar los paquetes que serán enviados a provincia, terminando así el proceso.

2.3 Objeto de informatización

A partir del análisis del flujo actual del proceso de Casación, se determinó la necesidad de informatizarlo para que los involucrados en el desarrollo diario de este procedimiento, según su cargo y autorización, puedan realizar las actividades judiciales desde su estación de trabajo. Se pretende que las comunicaciones entre las instancias provinciales y el tribunal supremo se establezcan a través de la aplicación y que los usuarios reciban notificaciones sobre el vencimiento de los términos de sus casos. Además, el sistema permite que la información sea captada y

gestionada de la misma forma en todo el país y que la organización digital de los documentos posibilite que no existan demoras por el esfuerzo humano requerido.

El proceso comienza desde que una parte postula la revisión de los errores jurídicos atribuidos a la sentencia en el TPP, se dispone sobre él, se notifica a la parte afectada en caso de estar en término, se presenta la oposición y se dispone sobre el vencimiento de términos hasta que se turna al ponente en el TSP, donde se dispone sobre el recurso, se celebra la vista, se dicta sentencia y se devuelven las actuaciones al tribunal de instancia con la nueva decisión. En la figura 1 se describe el proceso macro del negocio que constituye el objeto de informatización de este trabajo, pero solo se desarrolla la aplicación hasta la disposición sobre el vencimiento de términos, que se corresponde con el flujo del TPP.

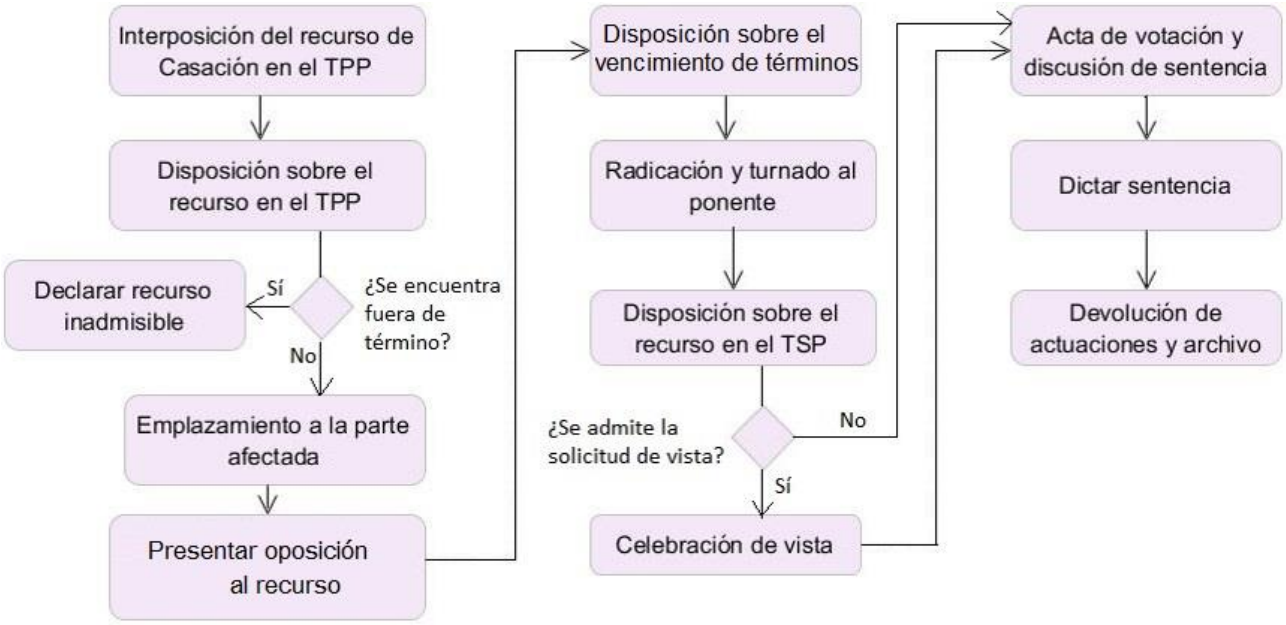

Figura 1 Diagrama macro del negocio

2.4 Modelado del negocio

El modelado del negocio se define como un proceso de representación de uno o más aspectos o elementos de una empresa, tales como su propósito, su estructura, su funcionalidad, su dinámica, su lógica de negocios y sus componentes. Es una actividad previa y complementaria a la Ingeniería de Requisitos y con esta disciplina se pretende llegar a un mejor entendimiento de la organización donde se va a implantar el sistema de software.

Los objetivos específicos del modelado de negocio son:

- Entender la estructura y la dinámica de la organización para la cual el sistema va a ser desarrollado.
- Tener un marco común para el equipo de proyecto, los clientes y los usuarios finales.
- Derivar los requerimientos del sistema necesarios para apoyar a la organización objetivo en su mejora.

- Entender el problema actual en la organización objetivo e identificar potenciales mejoras.

2.4.1 Responsables del negocio

Con el objetivo de entender la estructura y la dinámica de la organización para la cual el sistema va a ser desarrollado, se presentan en la siguiente tabla los involucrados en el desarrollo diario del procedimiento del Recurso de Casación.

Tabla 1 Responsables del negocio

Responsable del negocio	Descripción
Secretaria	Envía los documentos de emplazamientos para que la parte que resulte afectada pueda oponerse al recurso de casación. Da salida a la causa en el TPP y da entrada a la misma en el TSP. Mecanografía la sentencia y envía al departamento de correspondencia los paquetes con las causas por provincias.
Juez ponente	Realiza el acta de votación y discusión. Resuelve el asunto, disponiendo lo que en derecho procede y en este acto da solución al asunto que se presenta. Dicta sentencia y dispone la devolución de las actuaciones al tribunal de instancia.
Fiscal	Presenta un recurso de casación ante el tribunal de instancia.
Abogado	Presenta en el tribunal el convenio jurídico y emite un informe durante la celebración de la vista.

2.4.2 Procesos de negocio

A continuación se muestra un resumen del flujograma del proceso Casación en el TPP y en TSP, con el objetivo de obtener un marco común para el equipo de proyecto, los clientes y los usuarios finales y poder derivar posteriormente los requerimientos del sistema. Para ver el flujograma en su totalidad ver el artefacto “CEGEL-SITPC Descripción de procesos de negocio”.

Tabla 2 Descripción del proceso de Casación.

Objetivo	Revisar esencialmente las cuestiones de derecho referente al proceso Casación tramitado en el TSP.
Evento que lo genera	Presentación del recurso en el tribunal provincial.
Pre condiciones	Se elevan en casación las actuaciones de procesos ordinarios recurridos del tribunal provincial.
Reglas de negocio	Los asuntos se priorizan para el turnado en relación a si tienen preso o no, dando la preferencia a aquellas causas con presos.

	Si se trata de una causa que ya se resolvió en casación, la resuelve el mismo ponente. Una vez celebrada la vista, se dicta sentencia dentro de los 10 días hábiles siguientes. Si el recurso es con lugar o sin lugar de QF[2], sin lugar de IL[3] o sin lugar Mixto se dicta una única sentencia. Por generalidad si el recurso es con lugar QF por los artículos: 70.1, 2, 3, 7,8 se anula la sentencia y/o el juicio. Si el recurso es con lugar QF por los artículos: 70.4, 5, 6,9 se anula la sentencia. Si el recurso es con lugar IL se dicta sentencia anulando la sentencia del tribunal de instancia y se dicta una nueva sentencia. Si se declara Mixto con lugar por QF, se dicta sentencia sin entrar a conocer la IL. Si el QF es por los artículos: 70.1, 2, 3, 7,8 se anula la sentencia y el juicio. Si el lugar QF es por los artículos: 70.4, 5, 6,9 se anula la sentencia. Si se declara Mixto sin lugar por QF y con lugar por IL, se dicta sentencia anulando la sentencia del tribunal de instancia y se dicta una nueva sentencia. Si se acoge de oficio el recurso, se dicta una sentencia donde se anula la sentencia del tribunal de instancia y se dispone retrotraer hasta los trámites de: dictar una nueva sentencia, celebrar un nuevo juicio, auto de admisión de pruebas o hasta la fase de instrucción. La nueva sentencia, no podrá ocasionarle mayor gravamen que la anulada. Tampoco puede el Tribunal de casación, en un recurso del acusado, declarar la casación de oficio, para empeorar su situación.

[2] Quebrantamiento de Forma
[3] Infracción de la Ley

	La parte que haya establecido un recurso puede desistir de él mientras no recaiga resolución sobre el mismo. El criterio del turnado se da por la complejidad del delito (asunto) y por la cantidad de acusados. Cuando hay más de 4 acusados se da un término común de 20 días para la oposición al recurso de casación. Si hay varios acusados que apliquen recursos por materias diferentes (QF, IL) entonces el recurso es mixto. Del recurso se da traslado solamente a la parte que resulta afectada. De los recursos de los acusados siempre se le da traslado al Fiscal. Si el desistimiento se presenta en el TPP este debe dictar auto decretando el desistimiento, si se presenta ante el TSP a este es al que corresponde dictar el auto.
Responsable	Secretaria, Juez ponente, Fiscal, Abogado
Entradas	Causa, Expediente de Fase Preparatoria (EFP)

2.4.3 Diagrama de procesos de negocio

La figura 2 muestra el flujo de actividades que se realizan durante todo el proceso de Casación en el TPP, para observar el flujo en el TSP ver el anexo 1.

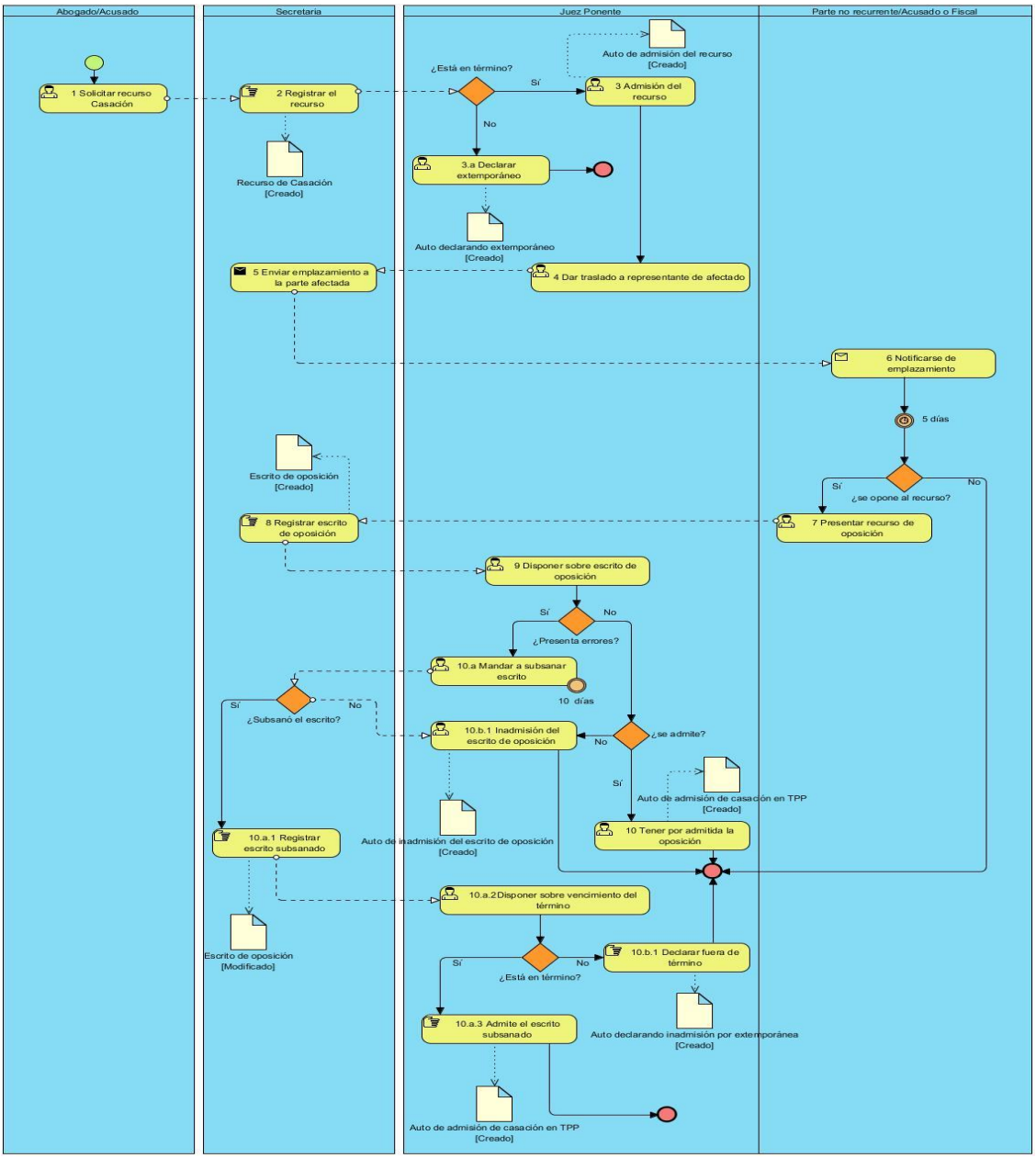

Figura 2 Flujograma del proceso Casación en el TPP

2.5 Propuesta del sistema

Antes de comenzar el diseño y la implementación es necesario tener conocimiento de las funcionalidades que debe cumplir el sistema para satisfacer las necesidades del cliente. En este epígrafe se presentan los actores, los requisitos funcionales y no funcionales de la aplicación a desarrollar.

2.5.1 Especificación de requisitos de software

Durante el desarrollo de las tareas de elicitación, análisis y especificación de requisitos se identificaron 56 requisitos funcionales y 45 requisitos no funcionales. Posteriormente se muestra el

listado de los requisitos funcionales y no funcionales identificados agrupados por categorías, los actores del sistema y el diagrama de casos de uso.

2.5.1.1 Requisitos funcionales

Los requisitos funcionales describen lo que el sistema debe hacer; dependen del tipo de software que se desarrolle, (…) describen con detalle la función del software, sus entradas y sus salidas, excepciones (Sommerville 2005).

Para dar cumplimiento a las fases de elicitación y análisis de requisitos se llevaron a cabo reuniones con el cliente, entrevistas y la arqueología de documentos, luego de haberse modelado el negocio fueron identificadas las actividades que se iban a informatizar en el sistema, a partir de las cuales se identificaron los requisitos funcionales, algunos de ellos son los relacionados con el escrito de casación:

RF1 Registrar escrito de casación.
RF2 Listar acusados.
RF5 Registrar documentos que acompañan.
RF12 Visualizar datos primarios.
RF20 Visualizar documento del escrito de casación.

Para consultar información detallada de los mismos ver el documento: “CEGEL-SITPC Especificación de requisitos de software”.

2.5.1.2 Requisitos no funcionales

Los requisitos no funcionales son aquellos que no se refieren directamente a las funcionalidades específicas que proporciona el sistema, sino a las propiedades emergentes de este como la fiabilidad, el tiempo de respuesta y capacidad de almacenamiento (Sommerville 2005).

El equipo de arquitectura del proyecto de Informatización de los TPC 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 obtener información más detallada ver documento: “CEGEL-SITPC Especificación de requisitos no funcionales” del proyecto.

2.5.2 Actores y casos de uso del sistema

A continuación se presenta una tabla que contiene una breve descripción de las actividades que realiza cada actor.

Tabla 3 Actores del sistema

Actor	Objetivos
Registrador	Actor encargado de registrar el escrito de casación y los escritos subsanados.
Generalización del abogado y el fiscal	Actor encargado de presentar un recurso de casación u oposición y los escritos subsanados.
Secretaria	Actor encargado de presentar un recurso de casación, así como de registrar los escritos subsanados, crear emplazamiento para las partes, dar salida a la causa en el TPP y dar entrada a la causa en el TSP, crear citaciones para vista, crear acta de vista y realizar las modificaciones necesarias en algunos documentos.
Juez ponente	Actor encargado de disponer sobre recurso de casación, disponer sobre vencimiento del término para subsanar recurso de casación, disponer sobre escrito de oposición, disponer elevación de actuaciones al TSP, disponer sobre recurso de casación en el TSP, declarar recurso inadmisible, crear acta de votación, dictar sentencia y crear evaluación de la tramitación.

2.5.2.1 Diagrama de casos de uso del sistema

Durante la fase de análisis y diseño se confecciona el diagrama de casos de uso del sistema, para ello se emplearon los patrones siguientes:

- **Actores múltiples:** Se tuvo en cuenta el rol común de registrador de escritos que tienen los actores: secretaria y generalización del abogado y el fiscal, como se observa en la figura 3
- **Inclusión:** Se evidencia en los casos de uso que incluyen los datos primarios de la causa, por ejemplo, el caso de uso Registrar recurso de Casación como se muestra en la figura 3.
- **Extensión:** Se utiliza para representar la extensión de los casos de uso que pueden o no tener documentos que acompañan el proceso que se realiza, por ejemplo, el caso de uso Registrar recurso de Casación como se muestra en la figura 3.

Luego de la fase de requisitos y teniendo en cuenta los patrones explicados anteriormente, el diagrama de casos de uso quedó confeccionado como se observa en la figura 6. Para obtener información más detallada sobre los casos de uso, consultar el documento “CEGEL-SITPC Especificación de casos de uso”. Además, para ver la descripción del caso de uso “Registrar recurso de Casación”, ver el anexo 2.

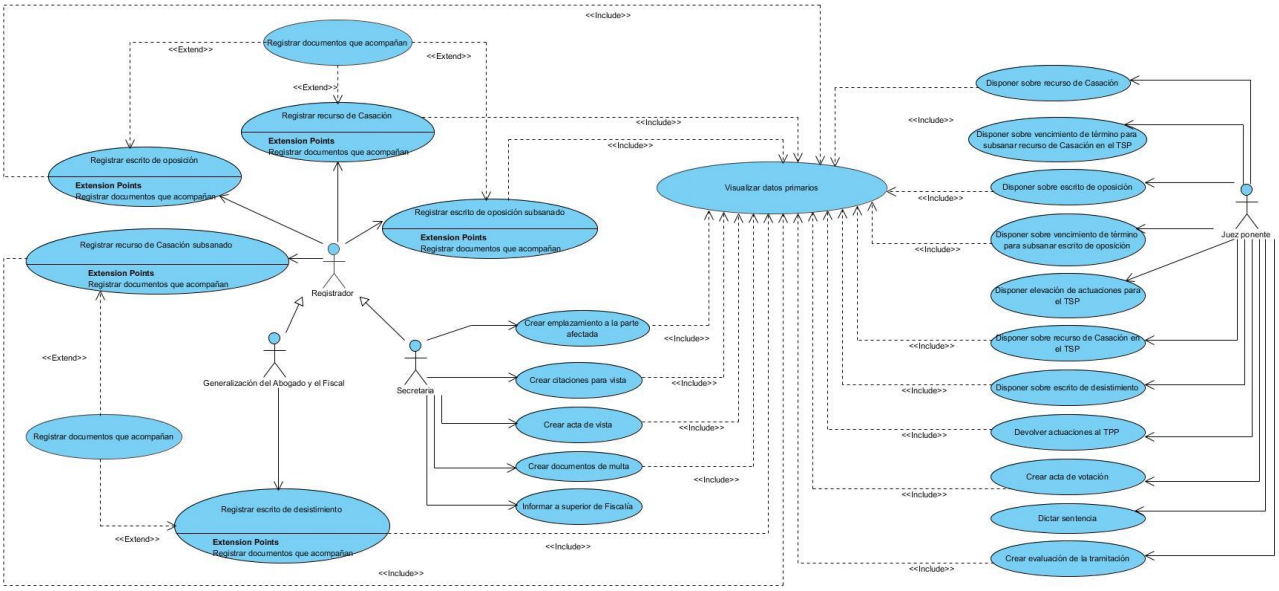

Figura 3 Diagrama de casos de uso

2.6 Arquitectura base

La arquitectura a la que responde el proyecto de Informatización de los TPC implementa el patrón Modelo-Vista-Controlador (MVC), pero también incorpora la capa de datos. Por tanto, la misma está compuesta por las capas horizontales: vista, controladora, modelo y datos. En la figura 4 se observa cómo se encuentran estructuradas y relacionadas cada una de estas capas. Se evidencian además, los componentes verticales que son: seguridad, validaciones, contenedor de servicios y tratamiento de excepciones. Para obtener información más detallada, consultar el documento del proyecto “CEGEL-SITPC Descripción de la arquitectura de software”.

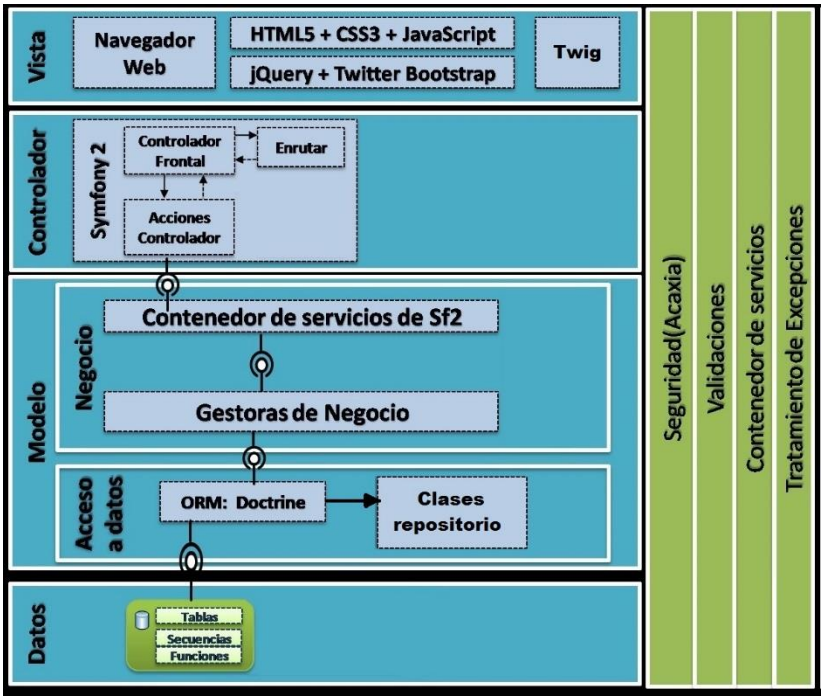

Figura 4 Arquitectura base

Capa vista: Permite la interacción del usuario con el sistema, capturando o mostrando la información que este introduce o solicita respectivamente mediante las interfaces de usuario. También contiene todo lo referente a la visualización de la información, el diseño, los colores, los estilos y la estructura visual de las páginas. La implementación de esta capa se encuentra ubicada en el paquete Resources, específicamente en views, donde se encuentran todas las páginas de la aplicación, las cuales importan las imágenes, archivos CSS y JavaScript necesarios, como lo muestra la figura 5.

Además, se aprovecha la herencia de plantillas a tres niveles que permite twig porque la plantilla_SIT.html.twig, que hereda de base.html.twig, incluye el menú y las funcionalidades en dependencia de cada materia y las del tercer nivel que son las de cada uno de los módulos permiten implementar el contenido del área de trabajo.

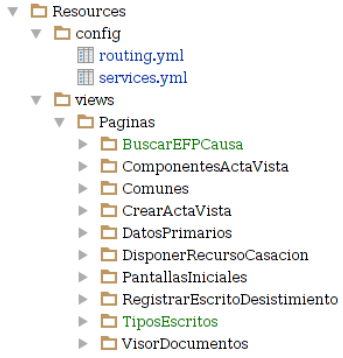

Figura 5 Contenido de las carpetas Resources y views

Capa controlador: Maneja la lógica de control, así como la construcción de páginas y formularios. Está compuesta por el controlador frontal de Symfony2, el cual maneja todas las peticiones realizadas por el usuario. Estas son enviadas por el controlador frontal al enrutador o routing, quien se encarga de enviar la ruta correspondiente, indicando así cuál es el controlador responsable de realizar la acción. Este va a mostrar la información correspondiente a través de las vistas, haciendo uso del motor de plantillas twig y puede acceder a la capa de negocio. Los controladores se encuentran ubicados en la carpeta Controller como muestra la figura 6:

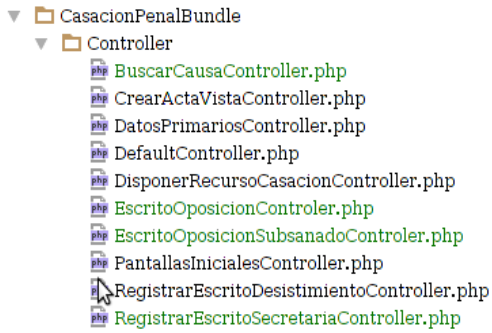

Figura 6 Contenido de la carpeta Controller

Capa de negocio: Está compuesta por los componentes del negocio, donde se encuentran las clases gestoras, encargadas del manejo de la lógica de negocio y ubicadas en la carpeta Gestor como muestra la figura 7. El acceso a estas clases gestoras por parte del controlador no se realiza directamente, para ello se hace uso del contenedor de servicios de Symfony2.

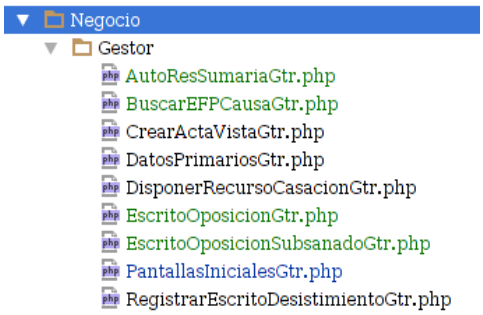

Figura 7 Contenido de la carpeta Gestor

Capa de acceso a datos: Gestiona las peticiones de la capa de negocio al consultar la base de datos y devolver la información requerida al gestor correspondiente. Está compuesta por las clases repositorio, que contienen las consultas para acceder y realizar acciones en las tablas.

Capa de datos: Contiene los esquemas, tablas, vistas y procedimientos que garantizan el almacenamiento y persistencia de la información utilizada por la aplicación.

Componentes verticales:

Seguridad: El sistema Acaxia es el que se encarga de la seguridad en la aplicación, garantizando 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, la cual se encuentra en el directorio: SIT\app\Resources\TwigBundle\views\Exception\error.html.twig como se muestra en la figura 8.

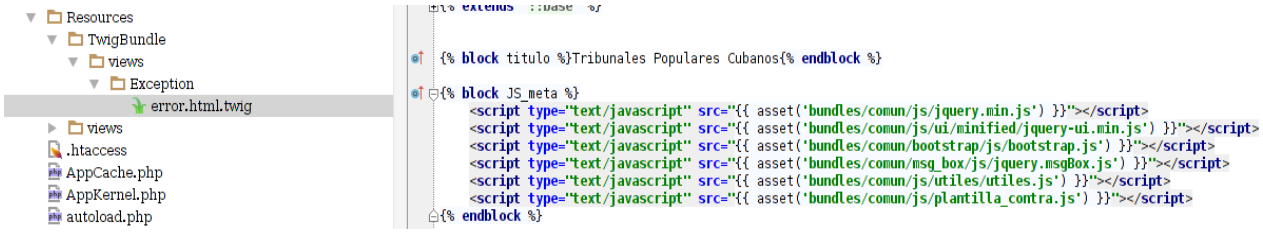

Figura 8 Tratamiento de excepciones

Contenedor de servicios: Permite que las respuestas de la aplicación sean más rápidas, potencia la reutilización de código, además estandariza y centraliza la forma de construir objetos en toda la aplicación. En la solución propuesta los servicios se encuentran declarados y publicados en el archivo services.yml, los cuales pueden ser instanciados desde cualquier clase controladora haciendo uso del atributo container. En la figura 9 se pueden observar los servicios que se emplean en las clases gestoras.

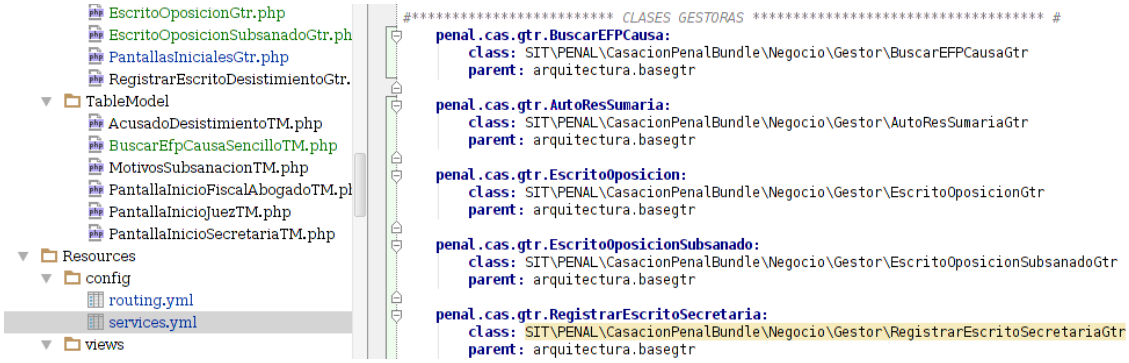

Figura 9 Contenedor de servicios

2.7 Modelo de diseño e implementación

En el proceso de desarrollo de software se define el modelo de diseño como un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales tienen impacto en el sistema. El mismo está divido por módulos y dentro de estos se cuenta con un paquete para la realización de cada caso de uso. El modelo de implementación, por otra parte, describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue.

2.7.1 Patrones de diseño

Teniendo en cuenta el estudio realizado en el capítulo anterior sobre los patrones de diseño, se explican a continuación cómo estos se emplean en la solución propuesta.

- **Experto:** Se utiliza este patrón principalmente en las clases controladoras, por ejemplo en EscritoRecursoCasacionController porque es la experta en crear las vistas relacionadas con el escrito del recurso de Casación. También es utilizado en las clases gestoras como en EscritoRecursoCasacionGtr, ya que se encargan de gestionar toda la información necesaria acerca de ellas. De esta forma se asigna una responsabilidad al experto en información, es decir, a la clase que cuenta con la información necesaria para cumplir con la responsabilidad.
- **Creador:** En este caso el patrón fue utilizado en las clases gestoras, como en EscritoRecursoCasacionGtr para crear las entidades y la clase controladora EscritoRecursoCasacionController a la hora de crear los formularios asociados a las vistas. De esta manera se guía la asignación de responsabilidades relacionadas con la creación de objetos, pues se establece un creador para conectar el objeto que se quiere producir con un evento específico.
- **Bajo acoplamiento:** Es un patrón que soporta el diseño de las clases más independientes, que reducen el impacto de los cambios, y también son más reutilizables. En la aplicación las entidades son las clases más reutilizadas y al observar la arquitectura de forma vertical se

puede apreciar que la vista se comunica con la clase controladora, esta se comunica con la clase gestora y esta clase lo hace con las entidades a través de los repositorios. De esta forma podemos concluir que las entidades no tienen asociación con la vista ni con el controlador, como se observa en el diagrama de clases del diseño.

- **Alta Cohesión:** En el caso del diseño propuesto, utilizando el patrón experto se le asignan a cada clase las responsabilidades que le corresponden y se establecen las condiciones para que cada clase colabore con las demás en la resolución de tareas que las implican a todas y que no son capaces de resolver por sí solas. Esto se evidencia cuando las clases controladoras luego de captar las entradas del usuario en las vistas, le proporcionan estos datos a las gestoras para que se encarguen de buscar o guardar en las entidades la información según corresponda. De esta forma la alta cohesión garantiza que exista un número moderado de responsabilidades dentro de un área funcional y que se colabore con las otras para llevar a cabo una tarea.
- **Controlador:** En el diseño este patrón se evidencia principalmente en las clases controladoras y en el controlador frontal de Symfony. Todas las peticiones realizadas por el usuario son manejadas por el controlador frontal, siendo este el único punto de entrada a la aplicación en un entorno determinado. En las clases controladoras este patrón sirve de mediador entre las interfaces y la clase gestora donde reside la lógica de negocio de la aplicación, como se observa en el diagrama de clases del diseño.

  El controlador frontal se encarga de recibir los datos que el usuario introduce a la aplicación y enviarlos a las distintas clases de acuerdo a la tarea que se desea realizar. Su funcionamiento está basado en que la lógica de negocios debe estar separada de la capa de la vista y así aumentar la reutilización de código y tener un mayor control sobre los cambios.
- **Factory method:** En el diseño se aprecia este patrón porque hay entidades que tienen asociada una clase repositorio que se encarga de realizar las operaciones de búsqueda y filtrado en la base de datos. Para hacer uso de los métodos de cada una de esas clases, el EntityManager de Doctrine hace uso del patrón *factory* para proporcionar una instancia de la clase repositorio. Es decir, se utiliza una fábrica para crear los objetos y decirle al contenedor de servicios que llame a un método en la fábrica y no directamente a una instancia del objeto.
- **Decorador:** En la solución propuesta se utiliza para el trabajo con plantillas, estas se manejan a través del motor de plantillas Twig. Este implementa una herencia que permite crear un “esqueleto” de plantilla base que contenga todos los elementos comunes del sistema y define los bloques que las plantillas descendientes pueden redefinir. La plantilla base.html.twig define un simple documento HTML con un esqueleto básico y es trabajo de las plantillas “descendientes” rellenar los bloques vacíos con contenido.

- En la composición de las vistas de la aplicación la plantilla base.html.twig define varios bloques: el *banner*, el menú superior y el pie de página que se mantienen iguales para todos los subsistemas y módulos dentro del SITPC, también se definen dos bloques más; uno para el menú lateral izquierdo y otro para el contenido. Las plantillas pertenecientes a cada módulo (Casacion.html.twig) heredan de la plantilla base y redefinen el menú lateral izquierdo. Las plantillas usadas en cada caso de uso (RegistrarEscrito.html.twig) heredan de la plantilla del subsistema al que pertenecen y redefinen el bloque de contenido, como se observa en el diagrama de clases del diseño.

  En la figura se separan los bloques que se redefinen en cada herencia, en azul los que define la plantilla base, en amarillo los que redefinen las plantillas de cada subsistema y en rojo el que redefine la plantilla de cada caso de uso.

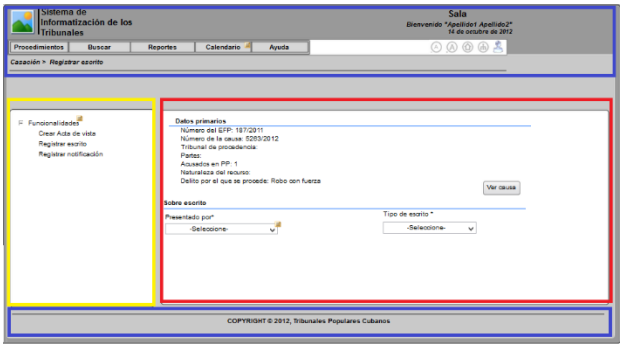

Figura 10 Patrón decorador

- **Llaves subrogadas:** Este patrón permite generar una llave primara única para cada entidad, en vez de usar un atributo identificador en el contexto dado. Por tanto, se usan enteros en columnas identidad, es decir, se generan las llaves de forma automática usando secuencias, garantizando así que no se cometan errores. Permite que las tablas sean más fáciles de consultar por el identificador y la uniformidad en la creación de las llaves. En la figura se muestra el ejemplo de la llave “idEFP” de la entidad EFP (Expediente de la Fase Preparatoria).

Figura 11 Patrón llaves subrogadas

2.7.2 Diagrama de clases del diseño

El diagrama de clases se realiza para tener en cuenta los detalles concretos de la implementación del sistema. En él, se describe la estructura de clases del sistema, las funcionalidades, los atributos

correspondientes a cada una de las clases, así como las relaciones entre ellas, generalmente de un solo caso de uso (Jacobson, Booch y Rumbaugh 2000).

A continuación se muestra el diagrama de clases del diseño en la figura 12, donde se representa la clase controladora EscritoRecursoCasacionController.php que construye la vista RecursoCasacion.html.twig y además, se comunica con la clase gestora EscritoRecursoCasacionGtr.php, esta última permite acceder a las entidades de la base de datos a través de las clases repositorio.

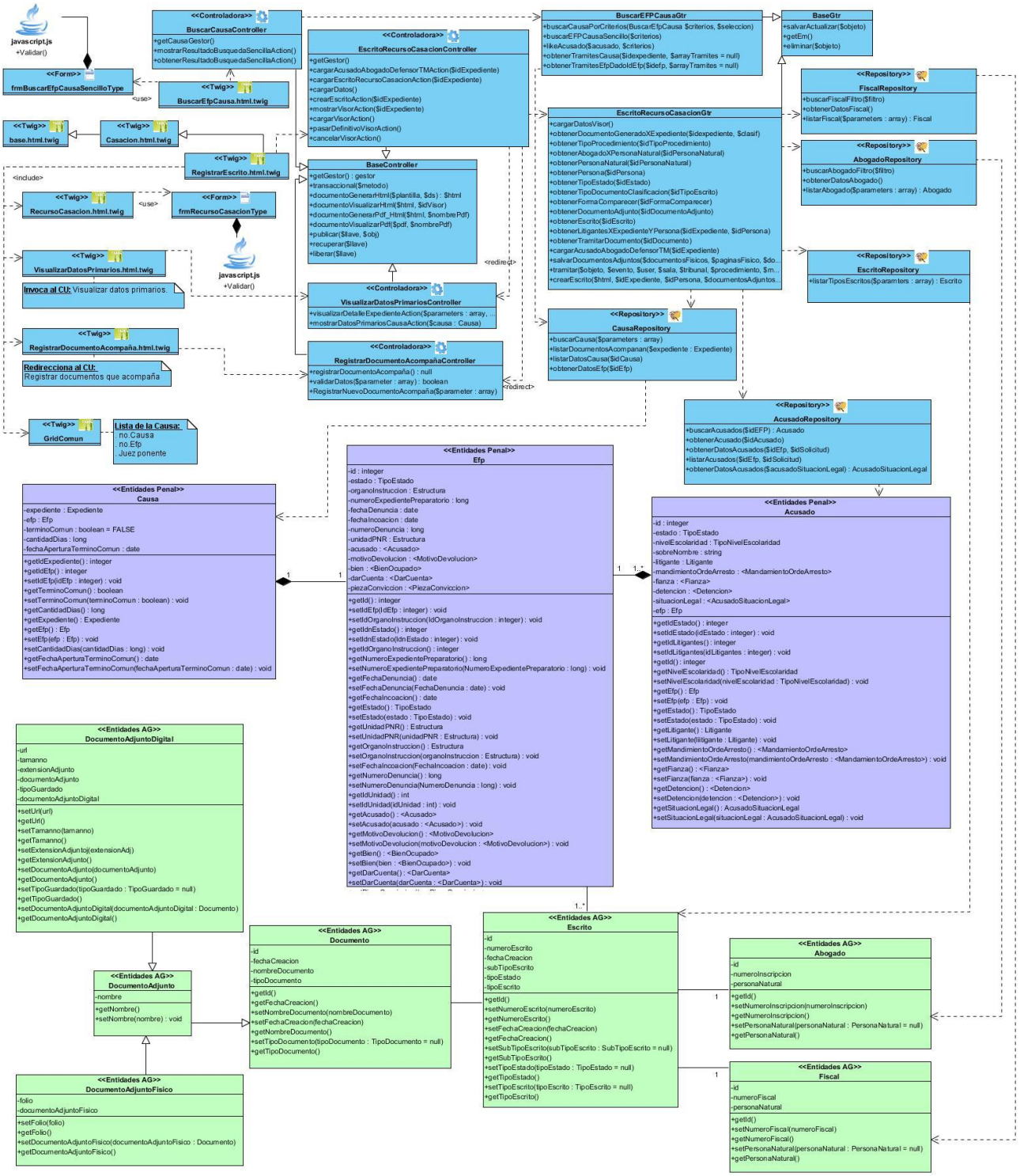

Figura 12 Diagrama de clases del diseño del caso de uso "Registrar recurso de Casación"

2.7.3 Diagrama de secuencia

Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso. Este contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes intercambiados entre los objetos. Además, describen la secuencia de acciones en un caso de uso que comienza cuando el actor invoca el caso de uso mediante el envío de algún tipo de mensaje al sistema (Jacobson, Booch y Rumbaugh 2000).

En la figura 13 se muestra el diagrama de secuencia del caso de uso Registrar recurso de Casación, donde el actor Generalización del abogado y el fiscal comienza la interacción al elegir la opción "Registrar escrito". También se representa la clase controladora EscritoRecursoCasacionController.php que construye la vista RecursoCasacion.html.twig y además, se comunica con la clase gestora EscritoRecursoCasacionGtr.php, esta última permite acceder a las entidades de la base de datos a través de las clases repositorio. Esta secuencia de eventos permite mostrar al usuario la información que solicita y registrar la que desea introducir en el sistema.

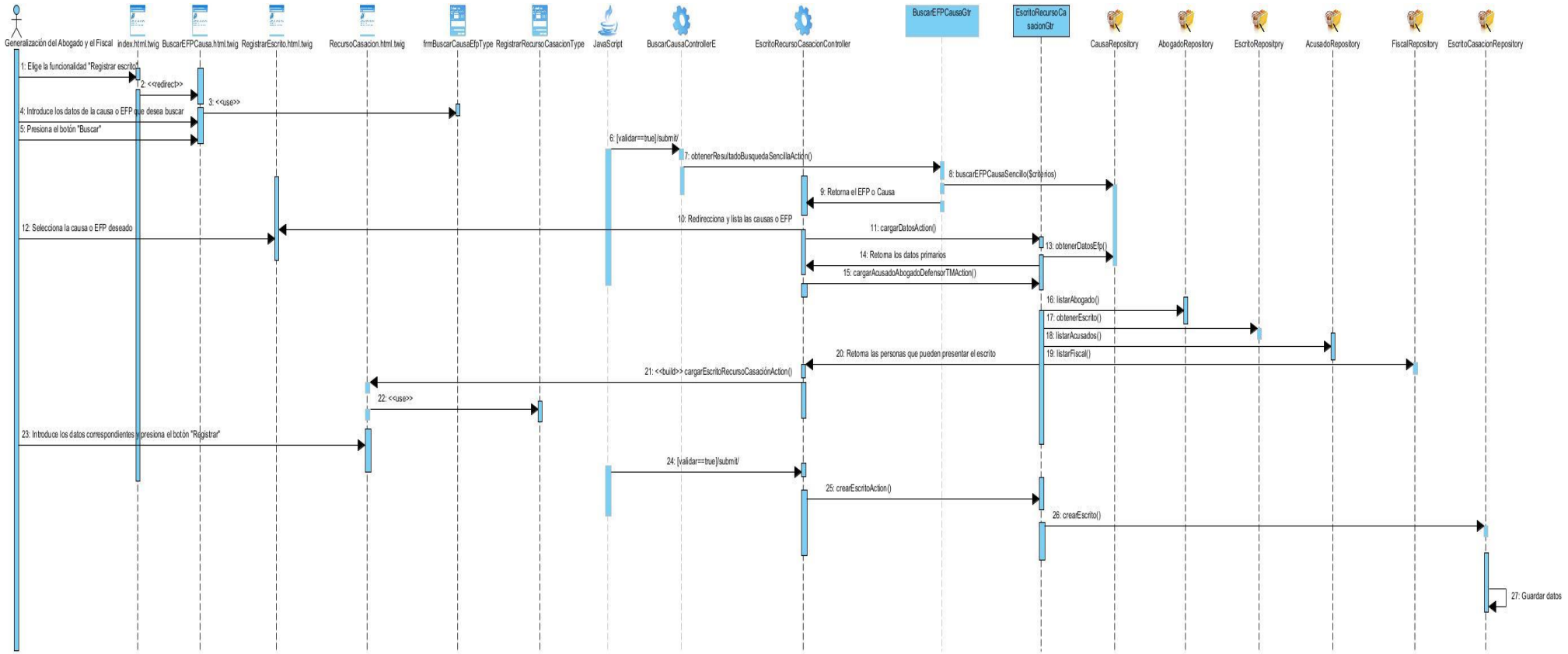

Figura 13 Diagrama de secuencia del caso de uso "Registrar recurso de Casación"

2.7.4 Modelo de datos

El modelo de datos se emplea para describir la estructura lógica y física de la información persistente gestionada por el sistema. Además, se utiliza para definir las estructuras de datos persistentes teniendo en cuenta las clases de diseño persistentes. En la figura 14 se muestra el modelo de datos del subsistema Penal adaptado al proceso del Recurso de Casación.

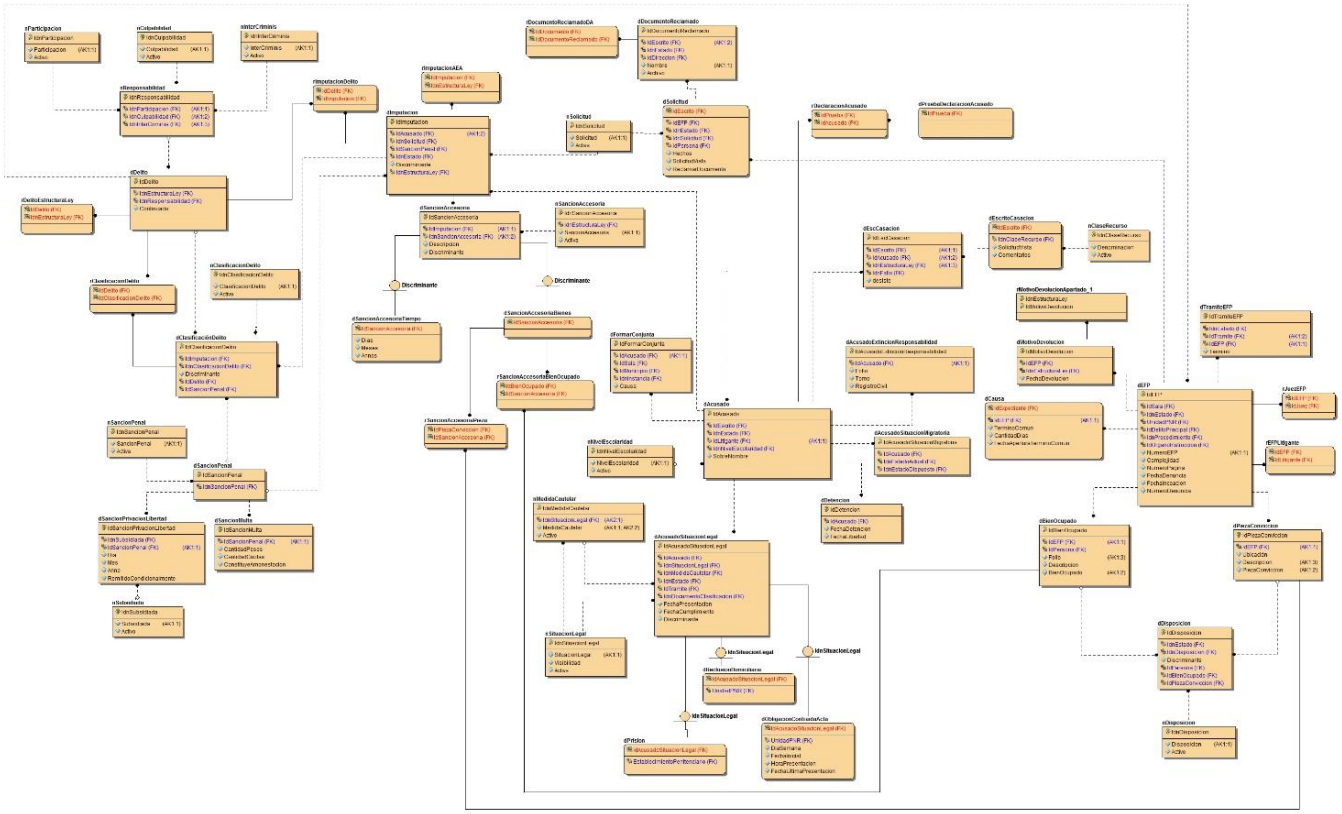

Figura 14 Modelo de datos

2.7.5 Diagrama de componentes

Los diagramas de componentes se utilizan para mostrar la estructura a alto nivel del modelo de implementación en términos de subsistemas y las relaciones entre los componentes. Describen los elementos físicos del sistema y sus relaciones, donde los componentes representan todos los tipos de elementos software que entran en la fabricación de la aplicación y que pueden ser archivos, paquetes y librerías. Con el fin de simplificar los diagramas, los diferentes componentes suelen agruparse en paquetes denominados subsistemas, los cuales a su vez pueden contener otros subsistemas y de manera general organizan la vista de implementación de un sistema (Jacobson, Booch y Rumbaugh 2000).

En la figura 15 se presenta el diagrama de componentes teniendo en cuenta los principales elementos que lo conforman:

- **Capa Vista:** Contiene el paquete de las vistas que se muestran al usuario, estas se ubican en CasacionPenalBundle/Resources/views/Paginas.
- **Capa Controladora:** Agrupa el controlador frontal y el paquete de las clases controladoras del *bundle* Casación de Penal donde se encuentran las acciones de los casos de uso.
- **Capa Modelo:** Agrupa las clases gestoras en el paquete ubicado en CasacionPenalBundle/Negocio/Gestor, estas establecen la comunicación con las clases del paquete Entity, que permiten el acceso a los datos almacenados en la base de datos.
- **Capa de Datos:** Encapsula todos los datos del sistema en la base de datos.

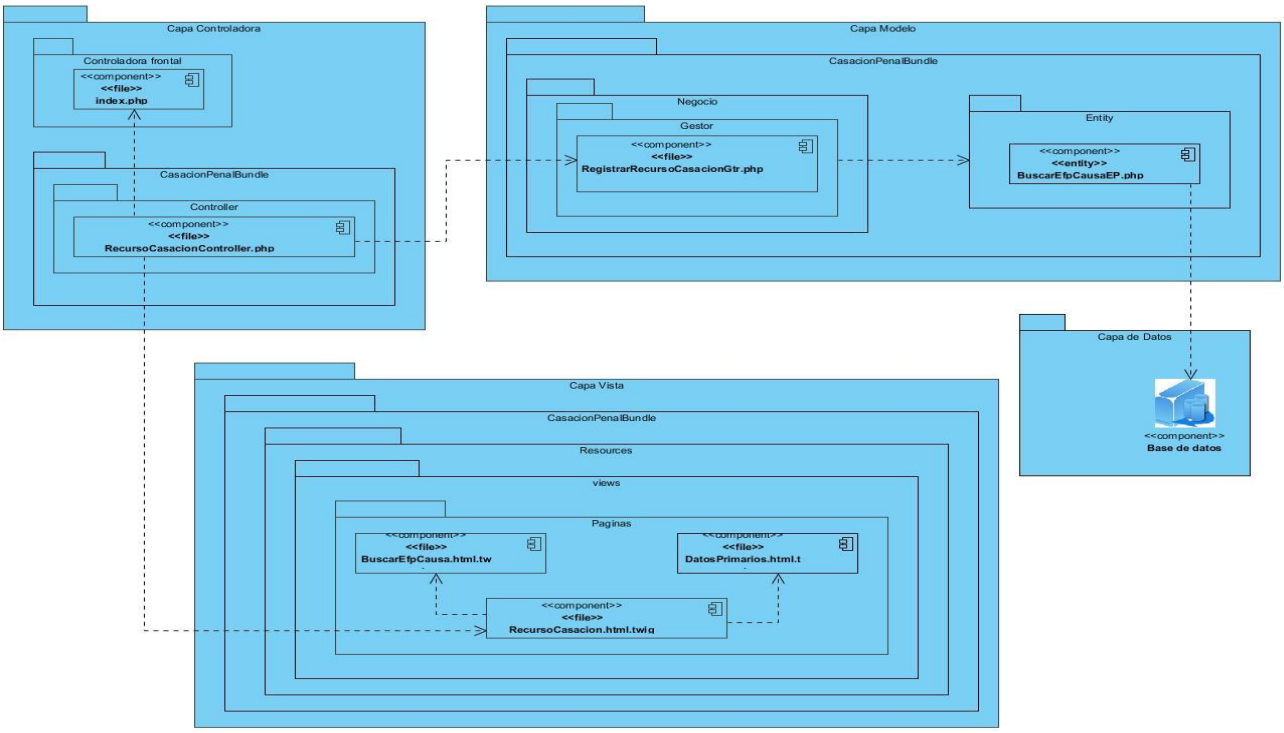

Figura 15 Diagrama de componentes

2.7.6 Diagrama de despliegue

El diagrama de despliegue describe como una aplicación se despliega a través de una infraestructura, muestra las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La funcionalidad de un nodo va a estar dada por los componentes que estarán distribuidos en él. Cada nodo es un elemento físico que representa un recurso computacional y que se utiliza para modelar la topología del hardware sobre el que se ejecuta el sistema y es donde se ejecutan los componentes, los cuales no son más que la representación de los elementos lógicos del sistema (Jacobson, Booch y Rumbaugh 2000).

Por tanto, la vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. El siguiente diagrama de despliegue tiene la intención de describir cómo va a ser desplegada la aplicación para las instancias del tribunal supremo y los tribunales provinciales en Cuba.

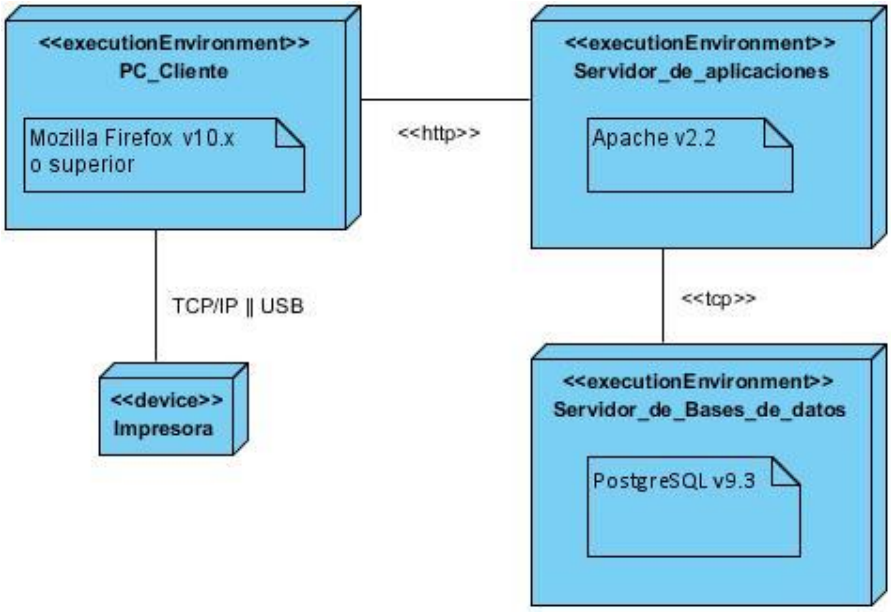

Figura 16 Diagrama de despliegue

2.7.7 Estándares de codificación

Con el propósito de respetar los estándares de codificación del proyecto de Informatización de los TPC se siguieron las siguientes pautas:

- **Cabecera del archivo:** Los archivos .php inician con una cabecera específica que indica la información de la versión y autor de los últimos cambios como se muestra a continuación.

  /**

  * @Clase controladora escrito. "*EscritoRecursoCasacionController.php*"

  * @modificado: 21 de Febrero del 2015

  * @autor: Ener

*/

- **Comentarios en las funciones:** Todas las funciones antes de su declaración tienen un comentario explicando lo que hacen, como se puede apreciar en la figura 17. De este modo ningún programador tendrá que analizar el código de una función para conocer su utilidad.

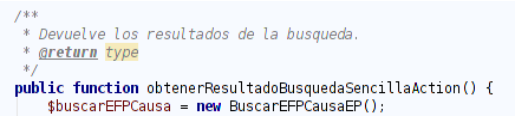

Figura 17 Comentario de una función

- **Ubicación y denominación de archivos:** Se respetaron las convenciones establecidas por Symfony2 y el equipo de arquitectura como se describe seguidamente:
  - Para las clases de gestión del negocio se usó el sufijo Gtr: RecursoCasacionGtr
  - Para los table model definidos para los grid se usó el sufijo Tm: BuscarEfpCausaSencilloTm
  - Para las entidades de presentación se usó el sufijo Ep: BuscarCausaEfpEp
  - En las páginas y plantillas html.twig definidas para la vista el nombre del archivo sigue el estándar de la denominación de las clases. En caso de estar formado por más de una palabra: RecursoCasacion.html.twig

Para obtener información más detallada sobre los estándares, ver el documento “CEGEL-SITPC Estándares de codificación” del proyecto.

2.7.8 Interfaces de la aplicación

En la figura 18 se muestra la interfaz de usuario correspondiente al caso de uso descrito anteriormente.

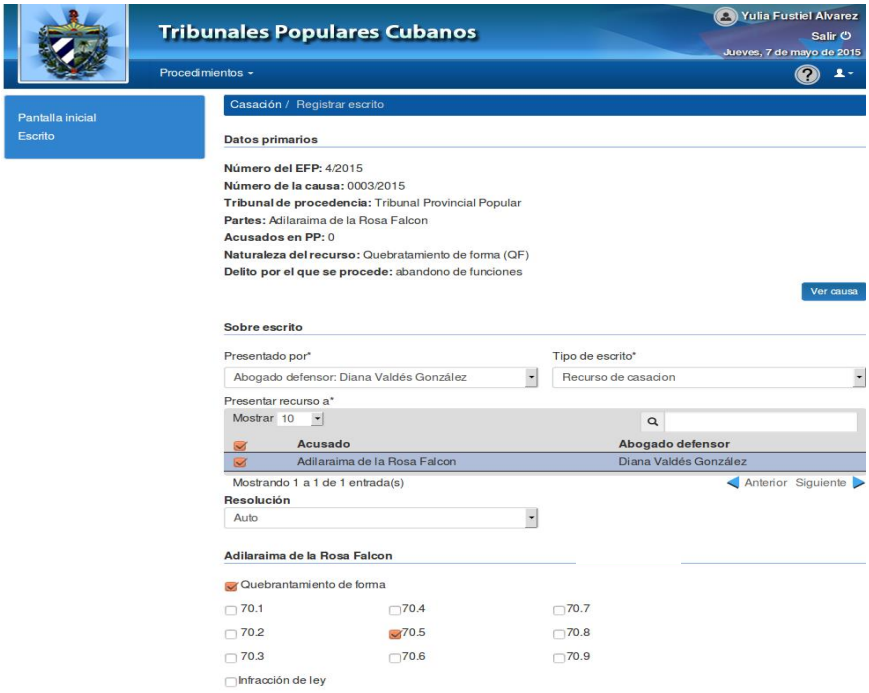

Figura 18 Interfaz de usuario del caso de uso "Registrar recurso de Casación"

2.8 Conclusiones parciales

El estudio realizado en este capítulo permitió determinar lo siguiente:

- La identificación de los principales procesos que rigen el negocio del Recurso de Casación de la materia Penal en el Tribunal Supremo Popular facilitó el entendimiento del entorno en el cual se desarrolla el problema a resolver.
- El análisis y especificación de los requisitos funcionales y no funcionales, así como la especificación de casos de uso y prototipos, tributó a un mejor trabajo en las próximas fases del modelo de desarrollo y posibilitó a los desarrolladores tener una mejor visión de las características que debe tener el sistema.
- La descripción de la arquitectura del sistema y el desarrollo de los diagramas de clases, secuencia y despliegue, posibilitó establecer una guía para lograr la correcta implementación del proceso de Casación, teniendo siempre en cuenta los patrones de diseño definidos, así como los estándares de implementación.
- Se obtuvo la propuesta de solución, permitiendo sentar las bases para realizar posteriormente la validación del sistema en cuanto a diseño e implementación.

Capítulo 3: Validación de resultados

3.1 Introducción

En el presente capítulo se realiza la validación de la solución propuesta con el objetivo de evaluar los resultados obtenidos tras realizar el diseño y la implementación del sistema. Esta se lleva a cabo mediante los prototipos de interfaz y la utilización de diferentes métricas, como las referentes a la calidad de la especificación de requisitos, las encargadas de medir la funcionalidad del diagrama de casos de uso del sistema y las concernientes al tamaño operacional de las clases. Además, los resultados se validan mediante la aplicación de métodos de caja blanca y caja negra.

3.2 Validación mediante métricas

Las métricas del software proporcionan una manera cuantitativa de medir los atributos internos del producto, permitiendo por tanto al ingeniero valorar la calidad durante la construcción del sistema. Con este objetivo a los requisitos y casos de uso obtenidos se les aplican la métrica para determinar la especificidad de los requisitos y la calidad del diagrama de casos de uso del sistema, además se realizan revisiones a los prototipos de interfaz. También, para la validación del diseño de la solución propuesta se emplea la métrica de tamaño operacional de las clases. A continuación se muestra un ejemplo del resultado de la aplicación de estas métricas.

3.2.1 Métrica para la calidad de especificación de los requisitos

Se emplea la métrica para la calidad de especificación de los requisitos teniendo como objetivo demostrar que la definición de los mismos cumple con lo que quiere el usuario y para que los errores no se propaguen a las fases siguientes. Por lo que luego de identificar los requisitos se realizó una comprobación para determinar su especificidad basada en la consistencia de la interpretación del equipo de revisores, constituido por la analista principal del módulo, una analista con experiencia en el rol y dos juezas de la materia Penal.

Para realizar este proceso inicialmente se determina el total de requisitos mediante la suma de los requisitos no funcionales y los requisitos funcionales identificados.

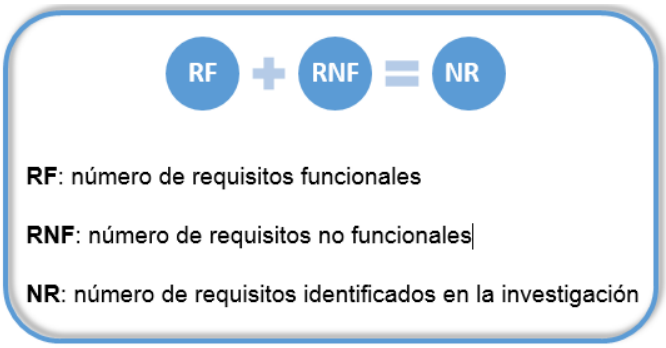

Figura 19 Fórmula para la cantidad de requisitos funcionales

Al aplicar la fórmula de la figura 19 se obtiene lo siguiente:

NR = RF + RNF

NR = 56 + 45

NR = 101

Luego se procede a medir la especificidad que no es más que el resultado de la división entre el total de requisitos con igual interpretación por parte de los revisores y la cantidad total de requisitos como se muestra en la figura 20.

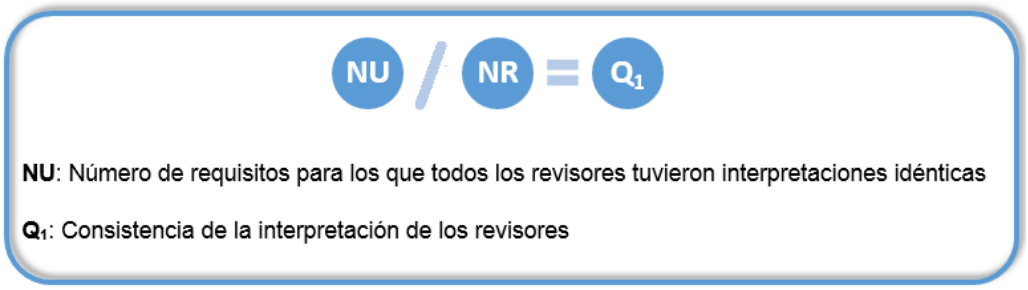

Figura 20 Fórmula para calcular la consistencia de la interpretación de los revisores

Al aplicar la fórmula de la figura 20 se obtiene lo siguiente:

Q_1 = NU / NR

Q_1 = 99 / 101

Q_1 = 0.98

Mientras más cerca de 1 esté el valor de Q_1 mayor será la consistencia de la interpretación de los revisores para cada requerimiento y menor será la ambigüedad en la especificación de los requerimientos (Villamizar 2012). Por tanto, teniendo en cuenta que el valor de Q_1 es igual a 0.98, se demuestra que el grado de ambigüedad es bajo.

3.2.1.1 Validación de requisitos mediante prototipos

Los prototipos son un método de validación utilizado para realizar una simulación del sistema que se quiere implementar. En el caso de la actividad de requisitos, se utilizan para comprobar la corrección y completitud de la especificación de requisitos.

Para validarlos mediante prototipos no funcionales, se realizaron 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 juezas de la materia Penal.
- **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 tres no conformidades en la primera, las cuales fueron:

- Nombre incorrecto en uno de los prototipos correspondientes al juez ponente.
- Adicionar el prototipo correspondiente al escrito de desistimiento.
- Cambiar el nombre en un prototipo de la vista.

Finalmente, como parte de las pruebas de aceptación de los prototipos no funcionales de interfaz de usuario, se obtuvieron resultados satisfactorios para las 21 propuestas en la segunda iteración como se muestra en la figura 21.

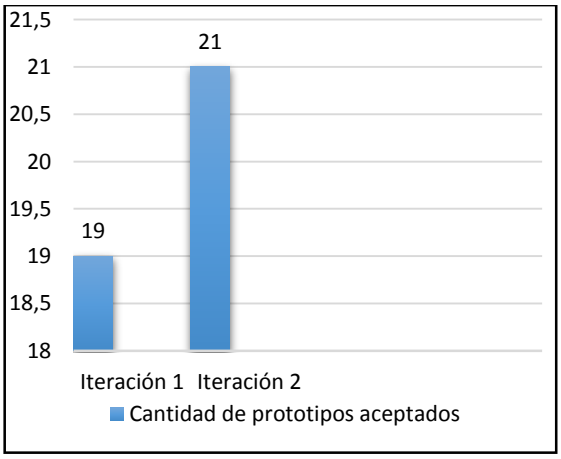

Figura 21 Gráfico de validación mediante prototipos

3.2.2 Modelo de métricas orientadas a objeto aplicadas al diagrama de casos de uso

En correspondencia con lo estudiado en el capítulo 1 se le aplican métricas al diagrama de caso de uso para medir la calidad de la funcionalidad, para ello se tendrán en cuenta cuatro atributos: completitud, consistencia, correctitud y complejidad. Estos aspectos cuentan con un conjunto de

factores que tienen asociadas una o varias métricas que establecen una medida cuantitativa del grado en que los factores indican una mala calidad. En el anexo 3 se muestran los resultados obtenidos al evaluar los factores de la calidad correspondientes al documento de descripción de casos de uso y el diagrama de casos de uso del sistema.

Finalmente el proceso de Casación mostró algunas no conformidades en la primera iteración con un 87,5% en consistencia, 96,43% en completitud y 91,67% en complejidad, evidenciando así los problemas existentes. Después de la realización de la segunda iteración, se obtuvo un diagrama de casos de uso del sistema con 100% de calidad en su funcionalidad, teniendo en cuenta los atributos medibles descritos anteriormente. En la figura 22 se muestran los resultados por atributo.

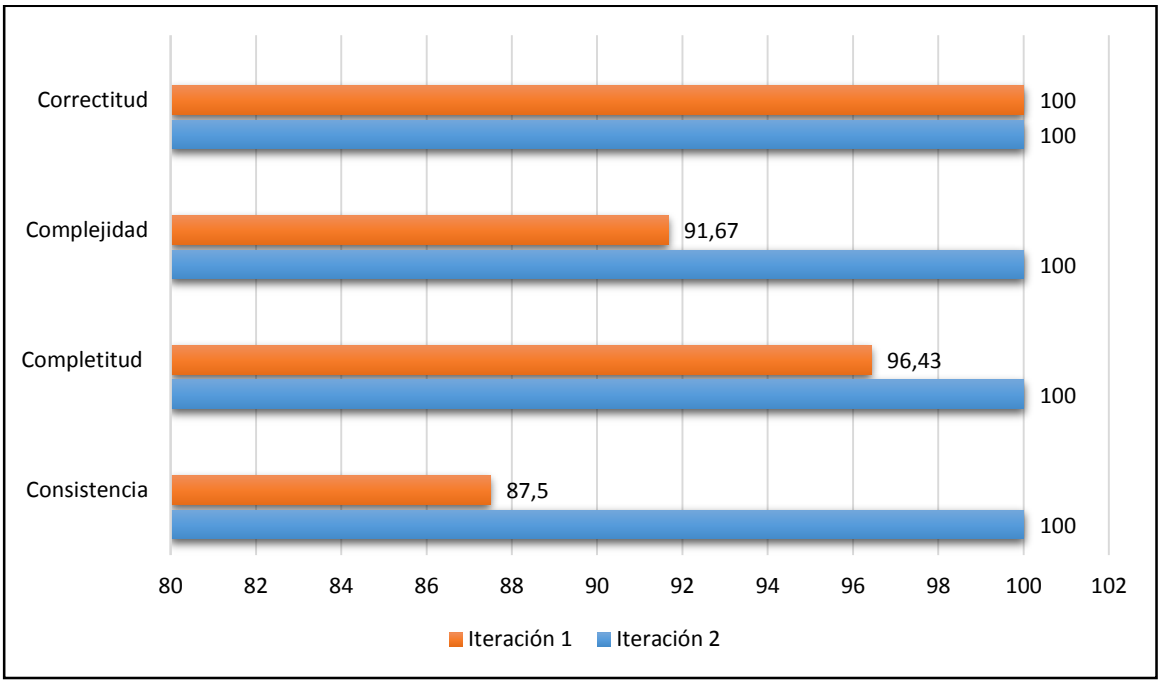

Figura 22 Grado de funcionalidad del diagrama de casos de uso del sistema

3.2.3 Tamaño Operacional de la Clase

La métrica Tamaño Operacional de la Clase (TOC) se emplea con el propósito de mejorar la comprensión de la calidad de la solución, estimar la efectividad del proceso y mejorar la calidad del trabajo. Está diseñada para evaluar los siguientes atributos de calidad:

- Complejidad de las pruebas: Se refiere al grado de dificultad que tiene implementado un diseño de clases determinado, lo que complejiza la aplicación de pruebas al sistema.
- Responsabilidad: Consiste en la responsabilidad asignada a una clase de un dominio.
- Reutilización: Consiste en el grado de reutilización presente en una clase dentro de un diseño de software.

Dicha métrica consiste en medir 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. Si el resultado

obtenido indica valores grandes, significa que la clase posee un alto grado de responsabilidad, debido a esto se reducirá la reutilización, se hará mucho más difícil la implementación y la realización de pruebas a dicha clase.

A continuación se muestran los umbrales definidos por Lorenz y Kidd (ver tabla 5) que se emplearán para evaluar los resultados de la aplicación de la métrica TOC.

Tabla 4 Umbrales de Lorenz y Kidd

Tamaño Operacional de la Clase	Umbral
Pequeño	<=20
Medio	>20 y <=30
Grande	>30

Primeramente se reflejan en la tabla 6 las medidas de los tres parámetros de calidad obtenidas luego de la aplicación de la métrica TOC a diez de las clases que conforman la propuesta de solución.

Tabla 5 Representación de los resultados obtenidos al aplicar la métrica TOC

Nombre de la clase	Cantidad de procedimientos	Complejidad	Responsabilidad	Reutilización
BuscarCausa Controller	3	Baja	Baja	Alta
EscritoOposicion Controller	8	Media	Media	Media
EscritoOposicion SubsanadoController	7	Baja	Baja	Alta
EscritoRecursoCasacionController	9	Media	Media	Media
EscritoOposicionGtr	17	Alta	Alta	Baja
EscritoOposicion SubsanadoGtr	11	Media	Media	Media
DisponerRecurso CasacionGtr	10	Media	Media	Media
CausaRepository	4	Baja	Baja	Alta
AbogadoRepository	3	Baja	Baja	Alta
AcusadoRepository	5	Baja	Baja	Alta

Luego de realizada la tabla, se obtuvo un promedio de 7.7 procedimientos teniendo en cuenta los 77 existentes. De las clases analizadas, 10 son de tamaño pequeño, cero de tamaño medio y cero

de tamaño grande. Además, al observar los parámetros de calidad podemos comprobar que el 50% de las clases tienen responsabilidad y complejidad bajas, el 40% de las clases poseen responsabilidad y complejidad media y solo un 10% alta. Respecto a la reutilización el 50% de las clases presentan elevados índices de reutilización, el 40% tiene una reutilización media y el 10% presenta una reutilización baja.

Finalmente, se demuestra la presencia de niveles bajos de complejidad y responsabilidad asociados con altos índices de reutilización, lo que facilitará la implementación de la solución. A continuación se pueden observar los resultados obtenidos distribuidos en tres gráficos de pastel.

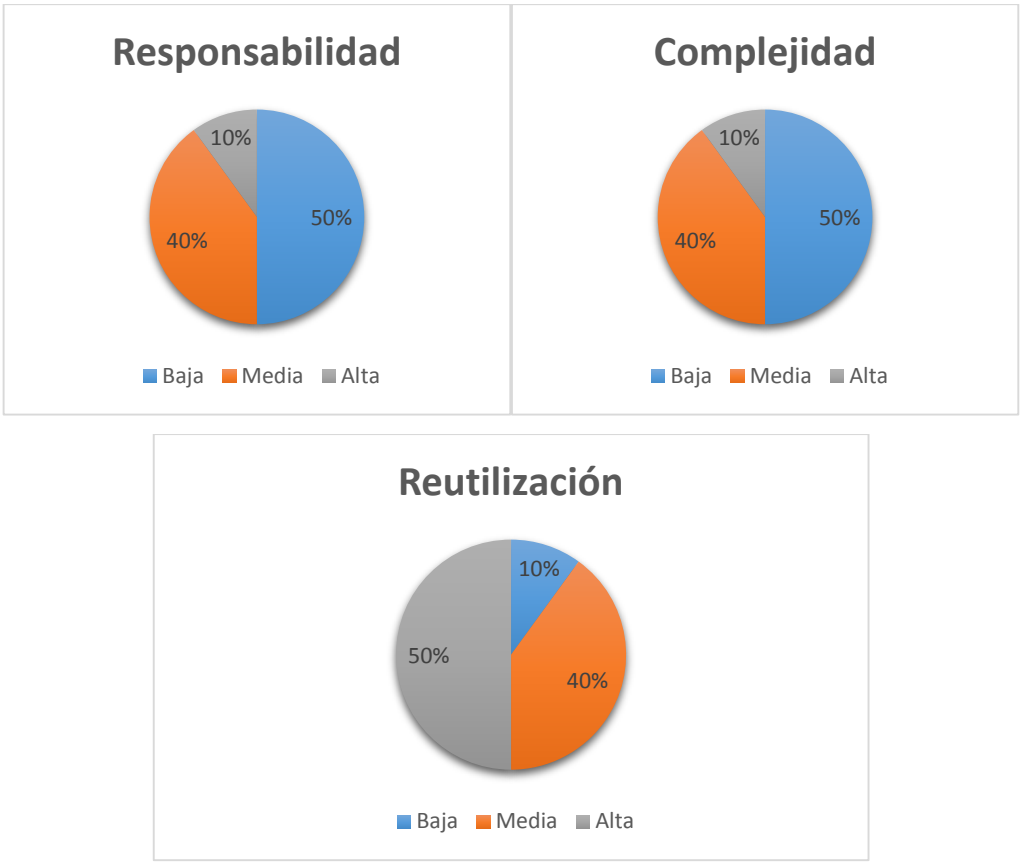

Figura 23 Resultados de la métrica TOC

3.3 Validación mediante pruebas

En correspondencia con el estudio realizado en el capítulo 1 y con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de la solución, se pretende probar el comportamiento del sistema, para ello se describe seguidamente cómo se aplican las pruebas de caja blanca y caja negra en este trabajo.

3.3.1 Pruebas de caja negra

Para probar la solución propuesta se realizaron pruebas funcionales, utilizando el método de pruebas de caja negra mediante la técnica de particiones equivalentes. Se realizaron 3 iteraciones

donde se utilizaron los casos de prueba diseñados para examinar los valores válidos e inválidos de las entradas existentes en el software.

Durante la primera iteración se encontraron un total de 12 no conformidades, de ellas las más representativas estaban relacionadas con reglas del negocio que no habían sido implementadas, lo que impedía la correcta realización de algunos de los procesos de negocio definidos. En la segunda iteración se detectaron 4 no conformidades sobre validaciones que debían realizarse para determinados campos, pero estas fueron corregidas en su totalidad, ya que al efectuar una tercera iteración no se detectó ninguna. En la figura se muestran los resultados obtenidos por iteración.

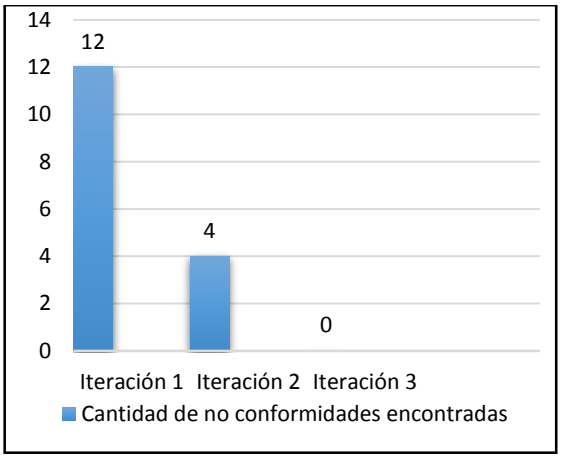

Figura 24 Resultados de las pruebas de caja negra

En la tabla 7 se muestra la descripción de las variables para validar el caso de uso Registrar recurso de Casación y en la tabla 8 los escenarios diseñados para la prueba.

Tabla 6 Descripción de las variables del caso de uso "Registrar recurso de Casación"

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	EFP	Campo de texto	No	Se inserta el número de EFP para filtrar una búsqueda.
2	Causa	Campo de texto	No	Se inserta el número de causa para filtrar una búsqueda.
3	Año	Lista desplegable	No	Se selecciona el año para filtrar una búsqueda.

Tabla 7 Diseño del caso de prueba

Escenario	Descripción	Variable 1	Variable 2	Variable 3	Respuesta del sistema	Flujo central
EC 1.1 Buscar	El sistema permitirá encontrar el	V 187	NA	V 2015	El sistema filtra una búsqueda entre todos los	1. Seleccionar Registrar Escrito en el

EFP o causa	EFP o causa a partir del cual se registrará un recurso de Casación.				números de EFP y muestra si existe alguno igual al escrito en el campo de texto.	menú izquierdo. 2. Seleccionar el botón Buscar.
EC 1.2 Los datos son incorrectos y/o existen campos obligatorios vacíos.	Se introducen datos incorrectos y/o existen campos obligatorios vacíos.	I + NA V 187	NA I + NA 	V 2015 V 2015 I ninguno	El sistema muestra un mensaje alertando que los datos son incorrectos.	1.Seleccionar Registrar Escrito en el menú izquierdo. 2. Seleccionar el botón Buscar.
EC 1.3 La operación es cancelada.	Se cancela la operación.	V 187	NA 	V 2015	Cierra la interfaz y regresa a la pantalla inicial.	1. Seleccionar Registrar Escrito en el menú izquierdo. 2. Seleccionar el botón Cancelar.

3.3.2 Pruebas de caja blanca

Durante la fase pruebas se utiliza también el método de caja blanca mediante la técnica del camino básico, ya que permite obtener una medida de la complejidad lógica del diseño y usar la misma como guía para la definición de un conjunto de caminos básicos, garantizando que durante la prueba se ejecute al menos una vez cada sentencia. En la figura 25 se muestra el código del método crearEscrito(), contenido en la clase EscritoOposicionGtr, que se encarga de guardar los datos del escrito creado en la base de datos.

```
public function crearEscrito($html, $idExpediente, $idPersona, $documentosAdjuntos, $user, $estadoDocumento, $actoProcesal,
$estadoTramite){
        $escrito = new EscritoOposicion(); //1
        $documentoGenerado = new DocumentoGenerado(); //1
        $expediente = $this->obtenerExpediente($idExpediente); //1
        $this->publicar('idExpediente', $idExpediente); //1
        $personaNaturalAbogado = $this->obtenerPersonaNatural($idPersona); //1
        $litigantePropio = $this->obtenerLitigantesXExpedienteYPersona($idExpediente, $idPersona); //1
        if ($litigantePropio == null) { //2
            $litigantePropio = new Litigante(); //3
            $litigantePropio->setTipoFormaComparecer($this->obtenerFormaComparecer(EFormaComparecer::Abogado)); //3
            $this->salvarActualizar($litigantePropio); //3
            $this->salvarActualizar($expediente); //3
        }
        $idEscrito = $this->recuperar('idEscrito'); //4
        if (isset($idEscrito)) { //5
            $escrito = $this->obtenerEscrito($idEscrito); //6
            $idDocumento = $escrito->getDocumentoGenerado()->getDocumentoPadre()->first()->getId(); //6
            $tramitarDocumento = $this->obtenerTramitarDocumento($idDocumento); //6
            $this->salvarActualizar($tramitePendiente); //6
            $this->liberar('idEscrito'); //6
        }
        $tramite = $this->tramitar($litigantePropio->getExpediente()->first(), EEvento::Registrar, $user,
        UtilRepository2::getSalaLogged(), UtilRepository2::getTribunalLogged(), UtilRepository2::getProcedimientoLogged(),
        UtilRepository2::getMateriaLogged(), $actoProcesal, $estadoTramite); //7
        $documentoGenerado->setTramite($tramite); //7
        $this->salvarActualizar($documentoGenerado); //7
        $this->publicar('idDocumento', $documentoGenerado->getId()); //7

        foreach ($documentosAdjuntos as $da) { //8
            $documentoAdjunto = $this->obtenerDocumentoAdjunto($da); //9
            $documentoAdjunto->setDocumentoGenerado($documentoGenerado); //9
            $em = $this->getEm(); //9
            $causa = $em->getRepository('ComunBundle:Penal\Causa')->find($idExpediente); //9
            $this->getUtil()->actualizarNumeroPaginaExpediente($documentoAdjunto, $causa->getEfp()->getId()); //9
            $this->salvarActualizar($documentoAdjunto); //9
        }
        $escrito->setDocumentoGenerado($documentoGenerado); //10
        $escrito->setPresentadoPor($personaNaturalAbogado); //10
        $documentoGenerado->setHtmlPdf($htmlPdf); //10
        $this->getEm()->flush(); //10
        $this->getUtil()->pasarDefinitivoDocumento($documentoGenerado); //10
        return $documentoGenerado; //10
    }
```

Figura 25 Método crearEscrito()

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

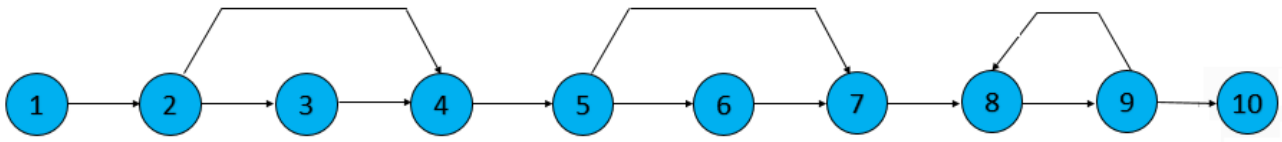

Figura 26 Grafo del flujo correspondiente a la función crearEscrito()

Una vez construido el grafo se determina la complejidad ciclomática, el cálculo es necesario efectuarlo mediante tres fórmulas, de manera tal que sea el mismo resultado en cada caso:
La complejidad ciclomática V(G) de un grafo de flujo G se define como:

- V(G) = cantidad de aristas(A) – cantidad de nodos(N) +2
  V(G) = 12-10+2 = 4

- V(G) = cantidad de nodos predicados[4](P) + 1
  V(G) = 3+1 = 4
- V(G) = número de regiones del grafo de flujo(R)
  V(G) = 4

Luego de efectuar el cálculo se obtuvo una complejidad ciclomática de valor 4, de manera que existen 4 posibles caminos por donde el flujo puede circular, este valor representa el número mínimo de casos de pruebas para el procedimiento analizado. A continuación se especifican los caminos básicos que puede tomar el algoritmo durante su ejecución.

- Camino básico #1: 1 – 2 – 4 – 5 – 7 – 8 – 9 – 10
- Camino básico #2: 1 – 2 – 4 – 5 – 6 – 7 – 8 – 9 – 10
- Camino básico #3: 1 – 2 – 3 – 4 – 5 – 7 – 8 – 9 – 10
- Camino básico #4: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10

Posteriormente se procede a ejecutar un caso de prueba.

- Caso de prueba para el camino básico #1

  -Descripción: Existe litigante teniendo en cuenta el expediente y la persona. Además, el escrito se guarda junto a los documentos adjuntos que lo acompañan.

  -Condición de ejecución: Existe litigante para el escrito, dado un $idExpediente y un $idPersona determinados.

  -Entrada: Un $idExpediente y un $idPersona determinados, que posean litigantes.

  -Resultados esperados: Teniendo en cuenta los datos pasados por parámetro se espera que la variable $litigantePropio no sea *null* y se cree el escrito con los datos adjuntos que lo acompañan.

  El resultado fue el esperado.

3.4 Validación de la investigación

Este epígrafe tiene como objetivo analizar las ventajas que supone la utilización del módulo Recurso de Casación del subsistema Penal del proyecto de Informatización para la Gestión de los Tribunales Populares Cubanos. Se comienza por la definición del conjunto de expertos, seguidamente se presentan las competencias que deben poseer los mismos para evaluar la solución propuesta con respecto a la forma en que se desarrolla actualmente el procedimiento. Por último se exponen los resultados esperados con el uso de la aplicación.

3.4.1 Definición del conjunto de expertos

El cuestionario confeccionado se aplicó a nueve expertos (ver anexo 4) del centro CEGEL y estuvo dirigido a un jefe de departamento, analistas, desarrolladores, arquitectos y al jefe de proyecto. Los

[4] Nodos predicados: Son los nodos de los cuales parten dos o más aristas.

encuestados reúnen un grupo de competencias que les permiten poseer las bases conceptuales para conocer cómo debe funcionar el sistema en el entorno en el que se va a desplegar y cómo se lleva a cabo actualmente el procedimiento de Casación.

3.4.2 Competencias de los expertos

Para realizar el proceso de selección de los expertos mencionados anteriormente, se tuvo en cuenta que reunieran los siguientes aspectos:

- Concepción global de la solución propuesta.
- Años de experiencia que oscilan 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.

Finalmente, luego de definido el conjunto de expertos atendiendo a las competencias citadas, se procedió a la ejecución de la encuesta (ver anexo 5) y al análisis de los resultados obtenidos como se describe seguidamente.

3.4.3 Resultados de la encuesta

Los resultados de esta encuesta (ver anexo 6) arrojaron que la mayoría de los expertos consideraron como “Alto” el rendimiento de los trabajadores y la satisfacción del cliente, dado que se pueden consultar directamente los escritos asociados al expediente utilizado en cada proceso, porque estos se encuentran almacenados en una base de datos indexada teniendo en cuenta diversos criterios. Además, demostraron que la media consideró que el tiempo para realizar la búsqueda de los expedientes, se espera que varíe en el orden de las horas a los segundos con el uso de la aplicación web. Se apreció también, que su empleo facilitará que basándose en lo que dicta la ley para tramitar el proceso, 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.

Con la propuesta del sistema, el expediente judicial digital permite agilizar los trámites judiciales y el control exhaustivo del vencimiento de términos, además de evitar tachaduras, el registro de la misma información en diferentes libros, la pérdida o el deterioro de los expedientes, y por ende su reparación. Se propicia el almacenamiento casi ilimitado y el acceso rápido a la información, en

contraposición a los mecanismos actuales de trabajo en los tribunales, en los cuales se producen cantidades exponenciales de información.

Asimismo la aplicación web posibilitará la realización de la supervisión que podrá ser realizada por los especialistas y supervisores designados, desde su territorio sin necesidad de trasladarse físicamente hacia otras localidades. Por tanto, gracias al uso del sistema desarrollado, se espera que se gestione la información del procedimiento de Casación de forma estándar, ágil y cómoda, elevando la calidad de vida de sus usuarios.

3.5 Conclusiones parciales

El estudio realizado en este capítulo permitió determinar lo siguiente:

- La aplicación de las métricas para la calidad de la especificación de los requisitos de software demostró que el grado de ambigüedad en la especificación de los mismos es muy bajo.
- La utilización del modelo de métricas orientadas al diagrama de casos de uso del sistema y la validación mediante prototipos permitió corregir los errores detectados antes de desarrollar el producto.
- La validación del diseño y de la implementación de la solución propuesta demostró que el sistema desarrollado satisface las necesidades del cliente.
- Los resultados de instrumentos de encuesta aplicados a un grupo de expertos del proyecto tribunales permitió demostrar que el uso de la aplicación web contribuirá a la mejora de la gestión de la información y a la estandarización de la tramitación del proceso de Casación.

CONCLUSIONES

Al concluir el presente trabajo se arriban a las siguientes conclusiones:

- El estudio de los conceptos fundamentales asociados al negocio y los sistemas judiciales tanto a nivel mundial como nacional, permitió comprender y demostrar la necesidad de desarrollar un nuevo sistema que se adaptara a la legislación cubana y resolviera los problemas existentes en los Tribunales Populares Cubanos.
- La realización del análisis sobre la metodología de desarrollo de software, los lenguajes de programación y las herramientas, permitió justificar la utilización de las mismas para el desarrollo de la investigación.
- La obtención del modelado del negocio permitió comprender la estructura, dinámica y los problemas existentes en la organización, así como identificar las actividades a informatizar. Además, la construcción de los modelos de diseño e implementación permitió lograr el desarrollo de la solución.
- A partir de la implementación se obtuvo un software funcional capaz de satisfacer las necesidades del cliente y resolver los problemas por los que se decidió comenzar el desarrollo del sistema.
- La aplicación de métricas y pruebas de software permitió evaluar en buena medida el nivel de calidad del diseño y la implementación de la solución.
- El análisis de los resultados de la encuesta demostró que el uso de la aplicación web contribuirá a la mejora de la gestión de la información y a la estandarización de la tramitación del proceso de Casación.

RECOMENDACIONES

Con la realización de este trabajo se recomienda:

- Continuar la implementación del módulo Recurso de Casación del subsistema Penal hasta la sentencia dictada por el Tribunal Superior Popular.

BIBLIOGRAFÍA

BLANCO, K.R. y AGUERO, D.N., 2010. *0505_CICLO DE VIDA DE PROYECTOS PILOTOS DEL PROGRAMA DE MEJORA*. 2010. S.l.: s.n.

BPMNbyExampleSPA [en línea], 2014. 2014. S.l.: s.n. Disponible en: http://www.bizagi.com/docs/BPMNbyExampleSPA.pdf.

CASANOVAS, J. 2004. Usabilidad y arquitectura del software. *Desarrollo Web* [en línea]. [Consulta: 21 mayo 2015]. Disponible en: http://www.desarrolloweb.com/articulos/1622.php.

CHAVES, M.A. 2011. La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. *InterSedes* [en línea], vol. 6, no. 10. [Consulta: 20 mayo 2015]. Disponible en: http://revistas.ucr.ac.cr/index.php/intersedes/article/view/790.

CONSEJERÍA DE PRESIDENCIA, JUSTICIA E IGUALDAD 2012. Colegio de Abogados de las Palmas. *Ilustre Colegio de Abogados de Las Palmas* [en línea]. [Consulta: 20 mayo 2015]. Disponible en: http://www.colegiodeabogadosdelaspalmas.com/2010/noticia.php?id=1412&tipo=1.

CRAIG, L. 2003. *UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado*. Madrid: Pearson Educación. ISBN 8420534382.

DE LA TORRE LLORENTE, C., CASTRO, U.Z., NELSON, J.C., BARROS, M.A.R., PACIOS, C.M., HIERRO, F.C. y MESA, I.G. 2010. *Guía de arquitectura N-Capas orientada al dominio con. NET 4.0*. S.l.: Krasis Press.

ESTEBAN, C.J. y ALEJANDRO, M.C. 2007. Implementación de un Servidor Web Apache. *Talca, Chile*,

FERNÁNDEZ, P.A. 2012. *Utilización de un marco de persistencia objeto relacional en el diseño de aplicaciones web multicapa* [en línea]. S.l.: s.n. [Consulta: 21 mayo 2015]. Disponible en: http://riunet.upv.es/handle/10251/17006.

FERNÁNDEZ, Y.A. 2013. *Género, Código y Juventud, construir sociedades más justas e inclusivas* [en línea]. La Habana: UNIJURIS. ISBN 978-959-7219-14-9. Disponible en: http://www.unjc.co.cu.

FLANAGAN, D. 2002. JavaScript: The Definitive. *Guide (4 ª Edición edición)*,

FRANGANILLO, J. 2011. Html5: el nuevo estándar básico de la Web. *Anuario ThinkEPI*, vol. 5, pp. 261–265.

GARCÍA RAMÍREZ, F. y PUELLO MARRUGO, P. 2011. Gestión de requisitos en la ingeniería de software. *INGENIATOR* [en línea], vol. 1, no. 1. [Consulta: 20 mayo 2015]. Disponible en: http://letravirtual.usbctg.edu.co/index.php/ingeniator/article/view/137.

GREY, L.W.G. y VILTRES, Y.M. 2014. Proceso de mejora del Sistema de gestión de Proyectos para Cuba y Venezuela. [en línea], [Consulta: 21 mayo 2015]. Disponible en: http://www.uajournals.com/campusvirtuales/journal/5/2.pdf.

GROUP, T. y OTHERS 2011. *PostgreSQL: The world's most advanced open source database*. S.l.: s.n.

GUERRERO, C.A., SUÁREZ, J.M. y GUTIÉRREZ, L.E. 2013. Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web. *Información tecnológica*, vol. 24, no. 3, pp. 103–114.

GUTIÉRREZ, J.J., 2011. *Framework* [en línea]. 2011. S.l.: s.n. Disponible en: http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

HEURTEL, O. 2011. *PHP 5.3: desarrollar un sitio Web dinámico e interactivo* [en línea]. S.l.: Ediciones ENI. [Consulta: 20 mayo 2015]. Disponible en: http://books.google.es/books?hl=es&lr=&id=GcymrdA9lZoC&oi=fnd&pg=PA10&dq=PHP+5.3&ots=8HYyJmcJBo&sig=kkA-vyH-DAlHoTLddBY0EUa4qrM.

IGLESIAS CANLE, I.C. 2010. El diseño legal de la nueva oficina judicial. [en línea], [Consulta: 20 mayo 2015]. Disponible en: https://minerva.usc.es/handle/10347/7958.

JACOBSON, I., BOOCH, G. y RUMBAUGH, J. 2000. *El proceso unificado de desarrollo de software* [en línea]. S.l.: Addison Wesley Reading. [Consulta: 21 mayo 2015]. Disponible en: http://dspace.ucbscz.edu.bo/dspace/handle/123456789/1184. Capítulo 11: Prueba

JUAN, F.J.M. 2000. *Guía de construcción de software en Java con patrones de diseño*. S.l.: Universidad de Oviedo, Escuela Universitaria de Ingeniería Técnica Informática.

MENÉNDEZ, L. 2007. LEXNET: la red al servicio de la justicia. *Escritura pública*, no. 48, pp. 20–22.

MENÉNDEZ, R. y ASENSIO, B. 2013. Apuntes Informática Aplicada a la Gestión Pública. Capitulo 2, Ingenieria del software, metodologias de desarrollo. 2011/12. Universidad de Murcia (España). Rafael Barzanallana. *Universidad de Murcia* [en línea]. [Consulta: 20 mayo 2015]. Disponible en: http://www.um.es/docencia/barzana/IAGP/IAGP2-Metodologias-de-desarrollo.html#BM1.

ORALLO, E.H., 2010. *ActaUML. Universidad Politécnica de Valencia* [en línea]. 31 marzo 2010. S.l.: s.n. [Consulta: 20 mayo 2015]. Disponible en: http://www.disca.upv.es/enheror/pdf/ActaUML.PDF.

OTTO, M., THORNTON, J., REBERT, C., THILO, J. y OTHERS 2011. Twitter Bootstrap. *Dostupné z: http://twitter. github. io/bootstrap/base-css. html*,

ÖVERGAARD, GUNNAR y PALMQVIST, KARIN 2004. *Use Cases Patterns and Blueprints*. S.l.: @TeamLiB. Stockholm, Sweden.

PARADIGM, V. 2010. Visual paradigm for uml. *Visual Paradigm for UML-UML tool for software application development*,

PÉREZ CEBADERA, M.Á. 2011. Situación en el uso de Lexnet como mecanismo de comunicación entre profesionales y órgano judicial. *Práctica de Tribunales.*, no. 80, pp. 8–17.

PÉREZ, D. 2010. Cuenta atrás para la nueva oficina judicial. *Escritura pública*, no. 62, pp. 28–31.

PRESSMAN, R. 1998. Ingenería de software. *Un enfoque práctico*, vol. 1.

RESIG, J. y OTHERS 2009. jquery: The write less, do more, javascript library. *Disponıvel em http://jquery. com/, Acesso em*, vol. 18, no. 04, pp. 2009.

RIESTRA, E.G. 2011. DE LA SOCIEDAD DE LA INFORMACIÓN A LA SOCIEDAD DEL CONOCIMIENTO: EL IMPACTO TECNOLÓGICO EN LA DOCENCIA JURÍDICA. [en línea], [Consulta: 20 mayo 2015]. Disponible en: http://132.248.65.15/libros/1/341/16.pdf.

RIVERO, D.G., 2008. *LEY DE PROCEDIMIENTO PENAL, Disposiciones del CGTSP*. 2008. S.l.: ONBC. ISBN 959-7066-43-9

RODRÍGUEZ, C.A.M. 2010. ÁMBITO PROCESAL DEL DERECHO PENAL ECONÓMICO EN CUBA. *3• Editorial En torno al Acuerdo No. 172 del Consejo de Gobierno del TSP Dr. Mario Ugido Rivero 10• La integración del Derecho ante las lagunas de la ley* [en línea], [Consulta: 20 mayo 2015]. Disponible en: http://www.ciidpe.com.ar/area5/dpre%20rodriguez%20mejias.pdf.

RODRÍGUEZ, M.G. 2008. El derecho penal desde una evaluación crítica. *Revista electrónica de ciencia Penal y criminología*, no. 10, pp. 12.

RUIZ-BERTOL, F.J. y ZARAZAGA-SORIA, F.J. 2007. El Control de Versiones en el aprendizaje de la Ingeniería Informática: Un enfoque práctico. *Actas de las XIII Jornadas de Enseñanza Universitaria de Informática* [en línea], [Consulta: 21 mayo 2015]. Disponible en: http://bioinfo.uib.es/~joemiro/aenui/procJenui/Jen2007/ruelco.pdf.

RUIZ TENORIO, R. 2010. *Las pruebas de software y su importancia en las organizaciones* [en línea]. S.l.: s.n. [Consulta: 21 mayo 2015]. Disponible en: http://cdigital.uv.mx/handle/123456789/28540.

RUP, C.I.E.P.U.R. 2011. INGENIERIA DE PROCESOS DE SOFTWARE. [en línea], [Consulta: 20 mayo 2015]. Disponible en: http://www.oocities.org/hwfiestasb/CatedrasUNP/IngenieriaProcesosSoftware/Syllabus_IPS.doc.

SAKIPOVA, D.Y.T. 2011. Modelado del Negocio, Sistema y Análisis de los Requisitos del Módulo Inventario del Proyecto de Software ERP. [en línea], [Consulta: 21 mayo 2015]. Disponible en: http://www.laccei.org/LACCEI2011-Medellin/published/IT246_Torres.pdf.

SOMMERVILLE, I. 2005. *Ingeniería del software* [en línea]. S.l.: Pearson Educación. [Consulta: 21 mayo 2015]. Disponible en: http://books.google.es/books?hl=es&lr=&id=gQWd49zSut4C&oi=fnd&pg=PA1&dq=Los+requisitos+funcionales+Sommerville+&ots=s623smAtvd&sig=KGySXA1Xvs-2FtksuxMAVTjTSiU.

TORRES, A.C., PREVAL, K.S. y RIQUENES, R.E.S. 2012. ARQUITECTURA DE REFERENCIA PARA PHP. [en línea], [Consulta: 20 mayo 2015]. Disponible en: http://www.researchgate.net/profile/Abraham_Calas/publication/275275693_ARQUITECTURA_DE_REFERENCIA_PARA_PHP/links/5536eb4d0cf2058efdeaa148.pdf.

Tribunal Supremo Popular de la República de Cuba. *Tribunal Supremo Popular* [en línea] 2012. [Consulta: 20 mayo 2015]. Disponible en: http://www.tsp.cu/tribunal_supremo_popular_cuba.

VALDÉZ, J.L.C., LEBRÚN, C.A.V., ISORDIA, A.C., SÁNCHEZ, O.G. y MEDINA, H.F. 2014. Diseño del modelo integral colaborativo para el desarrollo ágil de software en las empresas de la zona centro-occidente en México. *Nova Scientia*, vol. 7, no. 13, pp. 133–148.

VALLEJO, C. 2008. Sistema de control de versiones: subversion | Observatorio Tecnológico. *Observatorio Tecnológico* [en línea]. [Consulta: 21 mayo 2015]. Disponible en: http://recursostic.educacion.es/observatorio/web/ca/software/softwaregeneral/548-luis-garcia.

VARGAS, M.P. y GUEVARA, N.E.O. 2013. Bondades de la Programación en la Web como eje fundamental en el desarrollo de Software. *Revista de Innovación e Investigación Ingenieríl*, vol. 2, no. 3, pp. 39–49.

VERA, I.A. y SÁNCHEZ, I.A. 2004. Los gestores personales de bases de datos bibliográficas: conoce usted qué es y cómo se maneja el Procite. *Acimed*, vol. 12, no. 2, pp. 1–1.

VILLAMIZAR, J.A.P. 2012. Ingeniería de Software III: Métricas del Modelo de Análisis. *Ingeniería de Software III* [en línea]. [Consulta: 21 mayo 2015]. Disponible en: http://ing-software3.blogspot.com/2012/11/metricas-del-modelo-de-analisis.html.