Universidad de las Ciencias Informáticas

Facultad 3

Título: *Desarrollo del módulo Revisión del subsistema Laboral del proyecto Sistema de Informatización para la Gestión de los Tribunales Populares Cubanos (SITPC).*

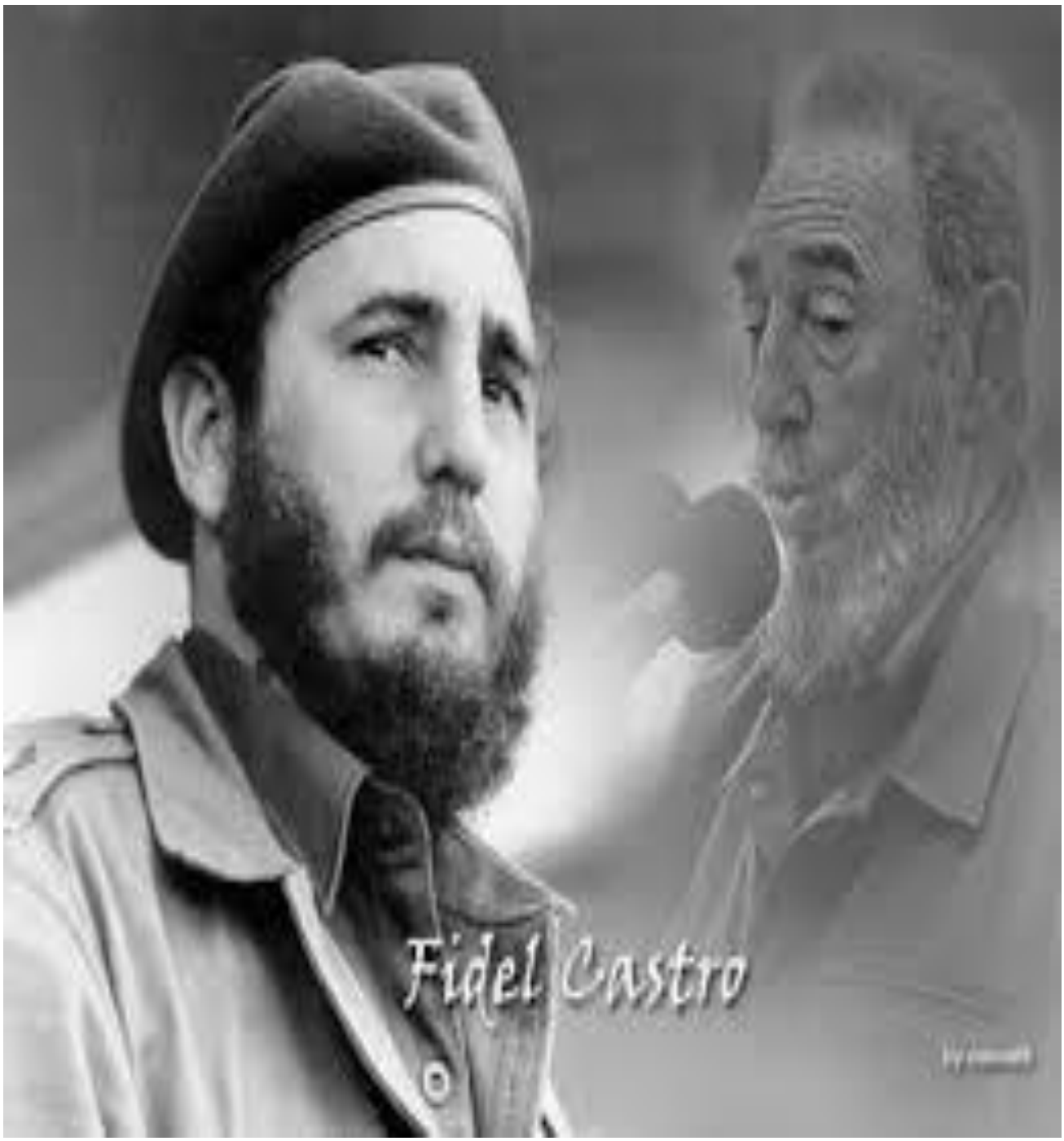
Autores: *Michel L. Frómeta Burey*

Abel Venero Acosta

Tutora: *Ing. Yulia Fustiel Alvarez*

Ciudad de La Habana, __

¨Año 57 de la Revolución¨



“Mi confianza en el triunfo final de lo que creo, es completa”.

Fidel Castro

Agradecimiento

A mi mamá, mi papá y mi hermano por sus años de dedicación y por su apoyo incondicional.

A mi tutora por su esfuerzo y dedicación…GRACIAS!!!

Dedicatoria

A toda mi familia y mis amistades de la universidad.

Declaración de Autoría

Declaración de autoría:

Por este medio declaramos que Michel Lázaro Frómeta Burey y Abel Venero Acosta, son los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) para que haga el uso que estime pertinente con este trabajo. Para que así conste firmamos la presente a los __ días del mes de junio del año 2015**.**

Michel Lázaro Frómeta Burey	Abel Venero Acosta
Firma del Autor	Firma del Autor

Ing. Yulia Fustiel Alvarez

Firma de la Tutora

Resumen

Los Tribunales Populares Cubanos (TPC) se encuentran estructurados en tres instancias: Municipal, Provincial y Suprema, e imparten justicia por las materias: Penal, Civil, Administrativo, Económico y Laboral. En la actualidad los numerosos procedimientos que las componen se realizan ineficientemente, generalmente de forma manual. A raíz de ello surge la necesidad de desarrollar un sistema informático capaz de gestionar la información generada durante el proceso judicial, garantizando la disponibilidad y el control de la misma. Con dicho propósito, los TPC en conjunto con el Centro de Gobierno Electrónico (CEGEL) de la Universidad de las Ciencias Informáticas, desarrollan un sistema informático que en correspondencia con los objetivos de la sociedad cubana, automatiza los procedimientos judiciales, lo cual está acorde con el proceso de actualización del modelo económico y político cubano. En la presente investigación se describe el desarrollo del procedimiento Revisión de la materia Laboral del Sistema de Informatización para la gestión de los TPC, orientado a gestionar los procesos de registro y disposición de escritos, vencimiento de términos, deliberaciones y dictado de sentencias de actos judiciales. La implementación del módulo incorpora los siguientes beneficios para los tribunales: gestión de los términos de los trámites que se realizan en el tribunal alertando a los usuarios cuando un trámite está a punto de declararse extemporáneo, mecanismos de búsqueda de información, así como el ahorro de recursos materiales.

Palabras claves: control de la información, declaración de extemporaneidad, disponibilidad de la información, trámites judiciales, Tribunales Populares Cubanos.

Índice

Índice

Índice de Tablas

Índice de Figuras

Introducción

El desarrollo tecnológico condicionado por la vorágine del mundo actual ha propiciado de forma creciente la evolución de las Tecnologías de la Información y las Comunicaciones (TIC). Esto ha traído consigo que formen parte indisoluble de la cultura tecnológica, en la cual la sociedad asume un rol activo como mera receptora de los beneficios que propicia y además contribuye a consolidarla. Producto de dicha interacción constante e incremental en los últimos años, las TIC han permitido a la humanidad progresar rápidamente en cuanto a ciencia y técnica, además en muchas ocasiones se tienen en cuenta entre los indicadores que se utilizan para medir el desarrollo social.

En Cuba se trabaja en el desarrollo de la informática como una de las ramas fundamentales para el soporte de la economía. Ello ha contribuido a mejorar la organización y agilización de los procesos que realizan las empresas y en consecuencia a elevar la calidad de vida de la sociedad. La posibilidad del empleo de las TIC en varias esferas no exime el ámbito de los tribunales de su presencia, pues aunque son caracterizados por mantener su tradición conservadora, estas permiten la informatización de los procesos que en dicho contexto tienen lugar.

La Universidad de las Ciencias Informáticas (UCI), dedicada a la informatización del país mediante la combinación de estudio-trabajo como modelo de formación y adiestramiento de profesionales calificados en la rama, tiene la importante tarea de contribuir con los Tribunales Populares Cubanos (TPC) para la agilización y organización de los procesos que allí se realizan. Específicamente el Centro de Gobierno Electrónico (CEGEL) de la Facultad 3 de la UCI se encuentra enfrascado en el desarrollo de un producto de software a solicitud de los TPC para mejorar la gestión de la información en cada momento del proceso judicial.

En los TPC existen varias materias: Penal, Administrativo, Civil y Laboral. El subsistema Laboral incluye el procedimiento Revisión, que es una concesión del ordenamiento jurídico a la justicia en detrimento de la seguridad. El carácter excepcional de este procedimiento se evidencia en su propia causa, la que constituye a su vez el requisito de proceder contra las sentencias firmes dictadas en procesos laborales o de seguridad social cuando con posterioridad a su firmeza se conozcan hechos de los que no se tuvo noticias antes, aparezcan nuevas pruebas o se demuestre fehacientemente la improcedencia, ilegalidad, arbitrariedad o injusticia notoria de la misma (1).

Actualmente en los TPC la creación y control de la documentación generada durante el procedimiento Revisión se realiza de forma manual, lo que puede traer consigo la aparición de posibles errores, tachaduras y borrones. Los expedientes se almacenan en formato duro en archivos que se encuentran en la secretaría, provocando el deterioro de los mismos en el transcurso del tiempo y

aunque están ubicados en estantes agrupados por años, el acceso a cualquier expediente se torna complejo debido al gran volumen de información, la búsqueda de estos expedientes se realiza de forma frecuente por parte de los jueces y fiscales que necesitan tener a su alcance la información necesaria para realizar sus consultas, estudios y análisis. En los TPC, así como en muchas empresas y entidades del mundo, el uso y seguridad de la información son de vital importancia para la toma de decisiones. La información obtenida en dicho proceso es diligenciada a entidades geográficamente distantes, por lo que la disponibilidad de esta puede ser afectada en algunos casos.

La consecutividad de los trámites es indispensable para mantener los niveles de transparencia del proceso judicial dentro del sistema de justicia. Una vez vencido el término o plazo para realizar una actividad, la secretaria tiene que dar cuenta de su vencimiento 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. Los problemas antes nombrados influyen negativamente en dicho proceso y en ocasiones el tiempo de tramitación es mayor que el estimado para esta cuestión, debido a que los documentos que se necesitan aún no han sido generados.

Ante los inconvenientes descritos anteriormente se plantea el siguiente **problema a resolver:** ¿Cómo gestionar la información asociada al procedimiento Revisión de la materia Laboral para elevar la disponibilidad y el control de la información?

El **objeto de estudio** engloba el desarrollo de software en la gestión de la informática jurídica, de manera que el **campo de acción** queda enmarcado en el procedimiento Revisión de la materia Laboral del proyecto de informatización para la gestión de los TPC.

Para dar solución al problema planteado se determina como **objetivo general:** desarrollar el módulo Revisión del subsistema Laboral del proyecto SITPC, de manera que se contribuya a elevar la disponibilidad y el control de la información generada durante el proceso. De acuerdo con la propuesta anterior se trazan los siguientes **objetivos específicos:**

- Definir el marco teórico de la investigación mediante el estudio y análisis de los principales referentes en los que se sustenta el desarrollo de software en la informática jurídica de gestión.
- Realizar el estudio del procedimiento Revisión de la materia Laboral mediante el levantamiento de información obtenida de los principales actores que intervienen en el mismo.
- Realizar el diseño de la solución para la definición de los objetos de software y la colaboración para satisfacer los requisitos identificados.

- Realizar la implementación del procedimiento Revisión para obtener un producto funcional.
- Validar los artefactos obtenidos en cada etapa del proceso de desarrollo, así como las variables de la investigación mediante la realización de pruebas de software, la aplicación de métricas para evaluar su calidad y el empleo del método Delphi.

Tareas de la investigación:

1. Análisis del negocio del recurso Revisión de la materia Laboral.
2. Análisis de sistemas de informática jurídica enmarcados en el dominio de las aplicaciones web existentes a nivel nacional e internacional.
3. Fundamentación de la metodología, tecnologías y herramientas a utilizar para el desarrollo de la solución propuesta.
4. Análisis del dominio del problema.
5. Realización del diseño para la agrupación de los requisitos funcionales.
6. Definición del modelo de datos.
7. Elaboración de diagramas de clases del diseño, paquetes y modelo de datos.
8. Aplicación de métricas para la validación del diseño propuesto.
9. Implementación de las funcionalidades.
10. Validación de la implementación utilizando los métodos de prueba de caja blanca y caja negra.
11. Validación de los requisitos funcionales, el diseño de la solución y las variables de la investigación utilizando el método Delphi y métricas de calidad.

Durante la presente investigación surge la siguiente **idea a defender:** si se desarrolla el módulo Revisión del Subsistema Laboral del Proyecto SITPC se contribuirá a elevar la disponibilidad y el control de la información que se genera durante el proceso judicial.

Variables dependientes: disponibilidad y control de la información.

Variable independiente: desarrollo del módulo Revisión del subsistema Laboral.

Con el objetivo de realizar las tareas de la investigación se emplearon métodos científicos de investigación (Teóricos y Empíricos):

Métodos Teóricos:

- **Histórico-Lógico:** empleado en el estudio de la bibliografía relacionada con la informática jurídica, además de adquirir conocimientos respecto a cómo se realizan las actividades jurídicas en los TPC.

- **Analítico-Sintético:** utilizado en el análisis individual de sistemas de gestión de la informática jurídica relacionados o no con el procedimiento Revisión Laboral, para replicar aquellas características aplicables a la nueva solución.

- **Modelación:** empleado para representar las estructuras y comportamiento de las clases diseñadas, mediante los diagramas de clases y de secuencia en la fase de diseño.

Métodos Empíricos:

- **Observación:** es la percepción directa del objeto de investigación. Permitió conocer la realidad mediante la relación directa de entes y procesos, de forma que se pudo lograr un seguimiento en cuanto a la realización del procedimiento de Revisión en los TPC.

- **Entrevista:** permitió el intercambio verbal con el cliente para obtener la mayor cantidad de información posible, entender todo el proceso de negocio, comprender la estructura y funcionamiento de la organización, así como las deficiencias existentes que permitieron definir el problema a resolver y establecer el objeto de estudio. Fue utilizada la entrevista formal con el cliente a través de la formulación de una serie de preguntas. (Anexo 14)

- **Cuestionario:** aplicado a un grupo de expertos del equipo de proyecto para obtener información de cuáles serán los beneficios que se obtendrán con el uso del SITPC. (Anexo 12)

- **Medición:** consiste en observar minuciosamente todo aquello que en el objeto de estudio seleccionado y de acuerdo con la teoría, sea relevante. A partir de la aplicación de las métricas de validación, la información obtenida puede ser de carácter cualitativo y cuantitativo.

La presente investigación se encuentra estructurada de la siguiente forma:

Capítulo 1: Fundamentación Teórica. En este capítulo se encuentra la información que fundamenta el progreso de la investigación, sustentada en el estudio de algunos sistemas de gestión jurídica a nivel nacional e internacional y la definición de los principales conceptos relacionados con el negocio.

Además se realiza el estudio del procedimiento Revisión Laboral, así como el de las herramientas y metodología a emplear para solucionar el problema planteado.

Capítulo 2: Descripción de la solución propuesta. En este capítulo se realiza un análisis de la solución que se propone a través de los diagramas: procesos del negocio, clases del diseño, secuencia, paquetes y el modelo de datos correspondiente.

Capítulo 3: Implementación y pruebas. Se presentan los estándares de codificación empleados en el desarrollo del sistema, la descripción de la seguridad y el tratamiento de errores del mismo. Además se realiza la validación de los requisitos funcionales del sistema y del diseño, mediante el empleo de un conjunto de métricas. También se describe el método Delphi para la validación de las variables de la investigación.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En este capítulo se abordan conceptos y definiciones generales que permitirán comprender el procedimiento Revisión de la materia Laboral de los TPC. Además, serán descritas la metodología propuesta para el desarrollo del sistema, las herramientas empleadas durante el proceso, la arquitectura y patrones de diseño, así como el marco de trabajo empleado para el desarrollo de la aplicación.

1.2 Descripción del procedimiento Revisión Laboral

El procedimiento Revisión de la materia Laboral que se realiza en los TPC, verifica todas las solicitudes y pruebas por parte de una persona jurídica o natural que puedan dar lugar a un cambio en la sentencia dictada. Inicia cuando un promovente[1] presenta el escrito de solicitud de Revisión ante el Tribunal Supremo Popular (TSP), al no estar de acuerdo con la sentencia emitida.

La secretaria del TSP recibe la solicitud de Revisión junto con otros documentos presentados y registra los datos necesarios en el libro de presentación de escritos (LPE), inmediatamente se radica el expediente otorgándole un número consecutivo que empieza en 1 cada año. Luego, el presidente de la sala designa al juez que estará al frente del proceso (Juez ponente) y dicta providencia disponiendo que se entregue el expediente al ponente para su estudio.

Ulteriormente se constituye la sala para deliberar el caso y decidir si se admite o no la solicitud. Se declara inadmisible la solicitud de Revisión en caso de extemporaneidad o que la medida no sea separación definitiva del cargo, se notifica al promovente (mediante oficio) que la solicitud de revisión fue declarada inadmisible, en otro caso las partes solicitan aclaración para esclarecer aspectos oscuros, suplir omisiones o rectificar equivocaciones importantes de las que adolezca la resolución firme. La sala, compuesta por el presidente, juez ponente, jueces profesionales y legos se reúne para decidir sobre la aclaración, mientras la secretaria notifica a las partes sobre el auto aclaratorio.

En el siguiente paso se resuelve sobre las pruebas propuestas por el promovente y las que de oficio dispongan, así como respecto a la suspensión de la ejecución de la sentencia si se ha solicitado o se dispone de oficio. Posteriormente se le notifica a los implicados sobre la admisión del proceso y al fiscal de la Fiscalía General de la República designado para estos casos, sobre la admisión del

[1] Persona natural que presenta una solicitud o petición de Revisión.

procedimiento, entregándosele copia del auto de admisión y de la solicitud de revisión. Dicho fiscal puede considerar que le asiste, no le asiste o le asiste en parte la razón al promovente, emite los argumentos de su criterio y propone pruebas si lo estima necesario.

Realizada esta tarea, el juez ponente se pronuncia sobre el dictamen del fiscal y resuelve en cuanto a las pruebas que proponga, seguidamente lo cita para su práctica en el caso de que se disponga. Se declara el proceso concluso para dictar sentencia, una vez practicadas las pruebas o que haya transcurrido el término prudencial concedido a la contraparte o tercer afectado sin que hayan presentado el escrito de personería, en caso de celebrada la vista, se decide también declarar el proceso concluso para dictar la sentencia; llevándolo nuevamente a deliberar. La sala se reúne y se emite el fallo donde se dicta la sentencia de haber lugar (HL), realizada esta actividad la secretaria se encarga de notificar a las partes sobre el resultado final de la sentencia.

Una vez dictada la resolución definitiva se remite al tribunal de origen el expediente con una copia certificada de la resolución. Para finalizar el procedimiento son notificadas las partes y la secretaria da cuenta al presidente de sala para que proceda a archivar el expediente.

1.2.1 Conceptos fundamentales

A continuación se realiza la descripción de los principales conceptos relacionados con el procedimiento Revisión de la materia Laboral para un mejor entendimiento del mismo. (1)

Revisión Laboral: revisión de una sentencia firme dictada en otro proceso por la existencia de causas que la hacen injusta.

Fiscal: es quien representa y ejerce el ministerio público en los tribunales. Es el encargado de velar por la pureza del procedimiento, es decir, por el cumplimiento de todos los requisitos y actuaciones previstas por la ley.

Dictamen: juicio que se emite sobre una cosa.

Sentencia: aquella en que el juzgador, concluido el juicio, resuelve finalmente sobre el asunto principal.

Auto: forma de resolución judicial, fundada, que decide cuestiones secundarias, previas, incidentales o de ejecución, para las que no se requiere sentencia.

Radicación: acción de enumerar y asentar en el libro correspondiente determinado escrito.

Disponer: acción realizada por el juez en el sistema que permite dar continuidad a los procesos y dentro de la cual generalmente aparecen las opciones (admitir, rechazar y subsanar).

Subsanación: corrección, enmienda, rectificación.

Pasar a definitivo: acción realizada sobre un documento. Después de ejecutada la acción el documento no puede ser modificado.

1.3 Sistemas existentes

1.3.1 Sistemas internacionales

Garantizar la eficiencia en los procesos judiciales constituye un requisito primordial para el cumplimiento de la ley en cualquier país del mundo. Entre los sistemas informáticos implementados con dicho fin, cuyo estudio constituye la base para el desarrollo de una nueva solución, se encuentran los siguientes:

GEDEX:

Software destinado a la gestión de expedientes jurídicos para despachos de abogados y profesionales. Una de sus características más atractivas radica en que puede ser utilizado sin compromiso de compra.

El estudio de dicha versión de pruebas corrobora la calidad de esta herramienta, ya que posee varias funcionalidades, desde la facilidad de compartir documentos en redes locales hasta la gestión de expedientes y contactos asociados. Almacena, recupera y gestiona documentación legal, agiliza operaciones habituales, como la localización rápida de expedientes, acciones pendientes, plazos de entregas y cobros. La herramienta admite un número ilimitado de cuentas de usuarios protegidas por contraseñas. Facilita ayuda para los usuarios, lo que contribuye a una familiarización rápida. (2).

Este software es imposible de utilizar debido a su carácter privativo, posee una orientación al sistema operativo Windows, además de la relativa inseguridad que constituye el uso de la nube para almacenar información sensible; por tanto se hace complejo emplear algunas de sus funcionalidades en la nueva solución.

TANTUM 2.1:

Es un sistema argentino que integra varias herramientas necesarias para la gestión jurídica, entre las que se destacan:

- Agenda completa de actividades del estudio y de los procesos.
- Administración de expedientes.
- Informes y reportes.
- Modelos automáticos de escritos.
- Búsquedas rápidas por múltiples campos.

Tantum 2.1 es una de las soluciones más simples y completas para la gestión de expedientes, opción que brinda esta herramienta para todos aquellos entes, oficinas u organismos que gestionan expedientes. Está desarrollado sobre un sistema de 32 bits para entornos gráficos, íntegramente diseñado y desarrollado utilizando herramientas y lenguajes de última generación. (3)

Lex-Doctor:
Sistema de gestión jurídica difundido en toda América Latina, adaptándose al procedimiento de todas las jurisdicciones. Este sistema permite desde la informatización de una oficina judicial individual hasta la informatización integral del poder judicial, abarcando la gestión de causas, la publicación de expedientes para consulta en línea y en tiempo real, con tecnología de última generación, escalable y adaptable a las necesidades de cada organismo. (4)

Los sistemas anteriormente mencionados constituyen soluciones informáticas eficientes en las áreas para las que fueron desarrollados, pero Cuba no puede proveer su utilidad ya que son privativos, algunos manejan la información y no la gestión propia de los procesos. Además no están implementadas todas las funcionalidades solicitadas por el cliente, como manejar procesos relacionados con las revisiones laborales, buscar documentos por diferentes criterios o generar reportes de los rollos confeccionados de las revisiones realizadas.

1.3.2 Sistemas nacionales

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

SISPROP:

Software que facilita la tramitación de los procesos penales en los tribunales del país. Dos de las características fundamentales del sistema son: la seguridad dada la naturaleza de la información que se maneja y la flexibilidad con respecto a cualquier modificación que pudiera sufrir la Ley de Procedimiento Penal. El sistema presenta limitaciones, tales como:

- No obtiene datos de la fase judicial del proceso.
- El nivel de informatización que supone el sistema es mínimo y en el caso de las apelaciones es prácticamente nulo.
- Está programado en Delphi utilizando como sistema de gestión de bases de datos a SQL Server 2000 por lo que no es compatible con el software libre. (5)

SISECO:

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

Esta aplicación no cumplió las expectativas iniciales de su creación. Prácticamente no informatiza ningún proceso, pues la radicación se realiza de forma manual y las salvas se guardan en disquetes, tecnología que ha quedado obsoleta en la actualidad. La información almacenada es eliminada cada 5 años, algo ineficiente para un sistema judicial para el cual mantener registros es primordial, ya que la información es el mecanismo principal para la justicia, por lo que tampoco es posible integrar la solución al SITPC.

1.4 Conclusiones del estudio

Los sistemas anteriormente descritos apoyan los procesos judiciales, poseen una demanda y aceptación variable, dependientes del contexto en el que se utilicen. El análisis de los mismos arrojó los siguientes resultados:

- Están concebidos para los sistemas judiciales de los países para los cuales fueron desarrollados y resulta difícil adecuarlos a las normas y leyes del sistema judicial cubano.
- El empleo de las funcionalidades relacionadas con la telefonía celular que permiten algunos de estos sistemas exige una inversión en la infraestructura de telecomunicaciones

cubana actual, además del relativo riesgo que conlleva el almacenamiento de información sensible en la nube.

- Son privativos, lo cual impide que se puedan hacer modificaciones al código.
- Las soluciones desarrolladas en Cuba fueron implementadas sobre tecnologías propietarias.
- No implementan la totalidad de las funcionalidades que requiere el procedimiento Revisión Laboral y a mayor escala el sistema jurídico cubano.

La siguiente tabla muestra un resumen por indicadores del estudio realizado.

Tabla 1. Tabla comparativa de sistemas informáticos existentes

Sistemas/Indicadores	Privativo	Compatible con tecnologías de Software Libre	Multiplataforma	Incluye alguna funcionalidad del procedimiento Revisión Laboral
GEDEX	No	Sí	No	No
TANTUM 2.1	Sí	No	No	No
LEX-DOCTOR	Sí	No	No	No
SisProp	Sí	No	No	No
SisEco	Sí	Sí	Sí	No

Dichas afirmaciones permiten concluir que es necesaria una nueva herramienta para informatizar el procedimiento Revisión de la materia Laboral en los TPC, al no existir una solución independiente y compatible con las tecnologías de software libre integrable con el SITPC y aplicable al contexto actual de los TPC.

1.5 Metodología de desarrollo de software

Las metodologías de desarrollo de software tienen como objetivo proporcionarle al equipo de desarrollo de software una guía para ordenar las actividades de un equipo, dirigir las tareas de cada desarrollador por separado y del equipo como un todo, especificar los artefactos que deben desarrollarse y ofrecer criterios para el control y la medición de los productos y actividades del proyecto (6). A continuación se describe la metodología a utilizar para el desarrollo de la solución.

RUP

Posee tres aspectos que la definen como metodología de desarrollo:

- **Dirigido por casos de uso:** un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante y a su vez representan los requisitos funcionales. El conjunto de todos ellos constituye el modelo de casos de uso, el cual describe la funcionalidad total del sistema.
- **Centrado en la arquitectura:** los arquitectos deben diseñar el sistema de forma tal que evolucione, no solo durante la etapa inicial sino también en las generaciones venideras. Para esto deben trabajar sobre los casos de uso significativos, los cuales constituyen las funciones fundamentales del software.
- **Iterativo e incremental:** en los sistemas grandes es práctico dividir el trabajo en partes más pequeñas o miniproyectos, donde cada uno de ellos es una iteración que posteriormente se convierte en un incremento o crecimiento del producto. RUP define nueve disciplinas (flujos) a realizar en cada fase del proyecto: modelado del negocio, análisis de requisitos, análisis y diseño, implementación, pruebas, distribución (despliegue), gestión de configuración y cambios, gestión del proyecto y gestión del entorno.

La metodología que se describe se divide en cuatro fases:

- Inicio
- Elaboración
- Construcción
- Transición

En la figura 1 se muestra la relación entre las fases y las disciplinas mencionadas anteriormente.

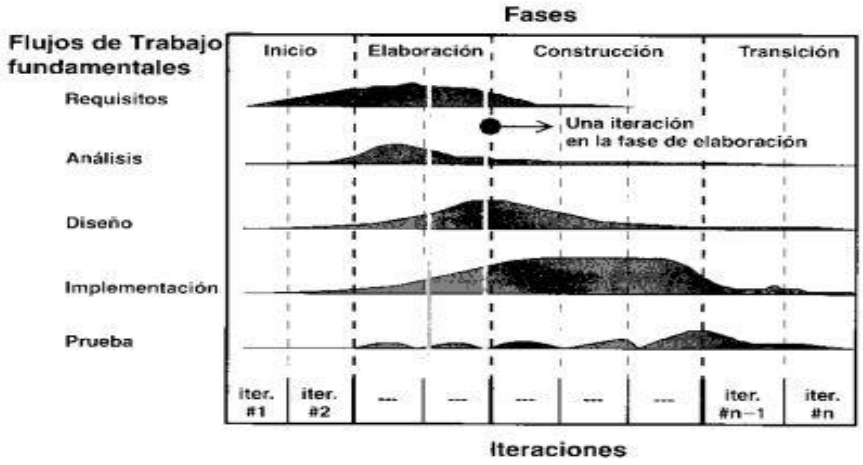


Figura 1: Fases y disciplinas de RUP (Pressman, 2005)

La selección de una determinada metodología en un proyecto depende de la organización y las necesidades del equipo de proyecto, la comunicación existente entre los involucrados, el plazo de entrega del producto, las restricciones y riesgos, entre otros elementos.

En el SITPC se ha seleccionado la metodología RUP debido a que el proyecto presenta características como: equipo con gran cantidad de recursos humanos, el tiempo de duración es largo y los clientes se encuentran lejos del equipo de desarrollo. Esta decisión está sustentada en varios aspectos, siendo el de mayor relevancia que es una metodología robusta que se adapta a las características y complejidad del sistema.

Otras cualidades que dan validez a la selección, son las siguientes: genera gran cantidad de documentación, lo que se ajusta perfectamente al proyecto, el cual requiere una especificación en detalle de todo el trabajo realizado, por su larga duración y el constante cambio del personal de desarrollo; centra el proceso de desarrollo en la arquitectura definiendo para ella actividades y artefactos; provee calidad, reutilización de los componentes, seguridad y mantenimiento del software mediante una gestión sistemática de los riesgos.

1.6 Lenguajes de programación estudiados

Un lenguaje de programación puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos respectivamente. (7)

1.6.1 Lenguajes de programación del lado del cliente

Lenguajes totalmente independientes de un servidor, que pueden ser directamente empleados por el navegador y no necesitan un pre-tratamiento.

1.6.1.1 Lenguaje de marcas de hipertexto (HTML5)

Lenguaje semántico para el marcado de texto, el cual provee una descripción del contenido que los navegadores web usan para construir la página web correspondiente. Facilita la utilización de enlaces que permiten tener referencias de un documento hacia otros sin importar donde estén localizados físicamente, haciendo uso del localizador uniforme de recursos (URL). La web es huésped de muchos protocolos pero HTML es la base, proporcionando el lenguaje básico para el mercado de texto contenido dentro de un documento estructurado que describe los roles y atributos de los elementos que la componen. (8)

1.6.1.2 JavaScript 1.2

Es un lenguaje interpretado, característica que lo hace especialmente idóneo para trabajar en la web, son los navegadores que se utilizan para viajar por ella los que interpretan y, por tanto, ejecutan los programas escritos en *JavaScript*. De esta forma, se envían documentos a través de la web que llevan incorporados el código fuente de programas, convirtiéndose de esta forma en documentos dinámicos, dejando de ser simples fuentes de información estática. Este lenguaje comparte muchos elementos con otros de alto nivel. (9)

JavaScript proporciona los medios para:

- Controlar las ventanas del navegador y el contenido que muestran.
- Evitar depender del servidor web para cálculos.
- Simular el comportamiento de las macros cuando no es posible usarlas.
- Comprobar los datos que el usuario introduce en un formulario antes de enviarlos.
- Comunicarse con el usuario.

1.6.1.3 Hoja de estilo en cascada (CSS3)

Es una tecnología que permite crear páginas web. Gracias a las CSS son mayores los resultados finales de la página, pudiendo hacer cosas que no se podían lograr utilizando solamente HTML, tales como: incluir márgenes, tipos de letra, fondos y colores, además de definir estilos propios en un archivo externo a las páginas; así, si en algún momento se quiere cambiar alguno de ellos, automáticamente se actualizarán todas las páginas vinculadas al sitio. (10)

1.6.2 Lenguajes de programación del lado del servidor

Son lenguajes reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él.

1.6.2.1 PHP 5.3

Lenguaje de código abierto, especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. Es multiplataforma, ya que es posible utilizarlo en la gran mayoría de las versiones existentes de Unix/Linux, Windows y MacOS. Debido a su código abierto los errores que este posee son rápidamente solucionados por la extensa comunidad que lo desarrolla a nivel mundial, además sus actualizaciones son realizadas con frecuencia sin necesidad de pagar por ellas. Es rápido, sin afectar en gran medida el rendimiento del servidor y posee gran interoperabilidad con Apache. Puede utilizar gestores como: Oracle, MySQL y PostgreSQL (11)

1.7 Lenguajes de modelado

1.7.1 Lenguaje unificado de modelado (UML 2.0)

Es un sistema de notación que se ha convertido en estándar en el mundo del desarrollo de sistemas. Está constituido por un conjunto de diagramas y permite al analista de sistemas generar un anteproyecto de varias facetas que sean comprensibles para los clientes, desarrolladores y todos aquellos involucrados en el proceso de desarrollo. Es necesario contar con todos esos diagramas dado que cada uno se dirige a cada tipo de persona implicada en el sistema. Un modelo UML indica qué es lo que supuestamente hará el sistema, no así el cómo. (12)

1.7.2 Notación de modelado de procesos de negocio (BPMN 2.0)

Es un estándar de la Iniciativa de Gestión de Procesos de Negocio (BPMI por sus siglas en inglés), cuyo principal objetivo es proporcionar una notación fácilmente comprensible por todos los usuarios del negocio, desde los analistas, desarrolladores, técnicos, hasta aquellos que monitorizarán y gestionarán los procesos. (13)

1.8 Marco de trabajo

Es una estructura tecnológica de soporte definida normalmente con artefactos o módulos de software, a través de la cual otro proyecto de software puede ser fácilmente organizado y desarrollado. Comúnmente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto. (14)

1.8.1 Symfony 2.1

Framework PHP (*Hypertext Pre-processor*) basado en la arquitectura MVC (Modelo-Vista-Controlador). Fue desarrollado para ser utilizado sobre la versión 5 de PHP ya que hace ampliamente uso de la orientación a objetos que caracteriza a este lenguaje.
La percepción de este marco de trabajo es reutilizar conceptos y desarrollos exitosos de terceros e integrarlos como bibliotecas. Emplea uno de los *frameworks* ORM (Mapeo Objeto-Relación) más importantes dentro de los existentes para PHP llamado Doctrine, el cual es el encargado de la comunicación con la base de datos, permitiendo un control casi total de los datos sin importar que sea de MySQL, PostgreSQL o SQL *server*. (15)

1.8.2 Twig 1.2

Es un motor de plantillas que brinda un amigable ambiente para los diseñadores y desarrolladores apegados a los principios de PHP, añadiendo útiles funcionalidades a los entornos de plantillas. Como características claves se pueden mencionar que es rápido, compila las plantillas hasta código PHP optimizado, posee un modo de recinto de seguridad para evaluar el código de plantilla que no es confiable, permitiendo utilizar este motor como un lenguaje de plantillas para aplicaciones donde los usuarios pueden modificar el diseño de la plantilla; es soportado por flexibles analizadores léxico y sintáctico, brindándole al desarrollador la opción de definir sus propias etiquetas, filtros personalizados y crear su propio lenguaje de dominio específico. (16)

1.8.3 Bootstrap 1.0

Es un *framework* desarrollado por *Twitter* que simplifica el proceso de creación de diseños web combinando CSS y *JavaScript*. La mayor ventaja es que se pueden crear interfaces que se adapten a los distintos navegadores, haciendo uso de un marco de trabajo potente con numerosos componentes web que ahorrarán esfuerzo y tiempo.

Este ofrece una serie de plantillas CSS y ficheros *JavaScript* que permiten integrar el *framework* de forma sencilla y potente en los proyectos web. Entre sus principales características están: (17)

- Ofrece un diseño sólido y estándares como CSS3/HTML5.
- Funciona con todos los navegadores, incluido Internet Explorer.

1.8.4 Biblioteca JQuery 1.8

Es una biblioteca de *JavaScript*, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM (Modelo de Objetos del Documento o Modelo en Objetos para la

Representación de Documentos), manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas web. 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: (18)

- Selección de elementos DOM.
- Interactividad y modificaciones del árbol DOM, incluyendo soporte para CSS 1-3.
- Eventos.
- Manipulación de la hoja de estilos CSS.
- Efectos y animaciones.
- Compatible con los navegadores *Mozilla Firefox 2.0+, Internet Explorer 6+, Safari 3+, Opera 9+ y Google Chrome.*

1.9 ORM

Técnica de programación utilizada para convertir datos entre el lenguaje de programación orientado a objetos empleado y el sistema de base de datos utilizado en la aplicación. Actualmente las bases de datos relacionales solo pueden almacenar datos primitivos y no los objetos que se vayan creando en la aplicación, para esto se crean los ORM, evitándole al desarrollador convertir esos datos primitivos en objetos. (19)

1.9.1 Doctrine

Es un potente y completo sistema ORM para PHP, con una capa de abstracción de la base de datos (DBAL del inglés *Database Abstraction Layer*) incorporada. Entre otras ventajas ofrece la posibilidad de exportar una base de datos existente a sus clases correspondientes y también a la inversa, es decir convertir clases a tablas de una base de datos.

Su principal ventaja radica en poder acceder a la base de datos utilizando la programación orientada a objetos (POO). Doctrine utiliza el patrón *Active Record* para manejar la base de datos, tiene su propio lenguaje de consultas de datos (DQL) y trabaja de manera rápida y eficiente. Es fácilmente integrado a los principales *frameworks* de desarrollo utilizados actualmente. (15)

1.10 Herramientas para el desarrollo del software

1.10.1 Herramientas CASE

Herramientas destinadas a aumentar la productividad en el desarrollo de software. Son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software, reduciendo el coste de las mismas en términos de tiempo y de dinero. (20)

1.10.1.1 Visual Paradigm 8.0

Herramienta para el desarrollo de aplicaciones utilizando modelado UML, la cual es ideal para ingenieros de software, analistas y arquitectos, que están interesados en el desarrollo de sistemas de gran tamaño que necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. Dicha herramienta ofrece ventajas como:

- Navegación intuitiva entre la escritura del código y su visualización.
- Potente generador de informes en formato PDF.
- Ambiente visualmente superior de modelado.
- Sofisticado diagramador automático de diseños.
- Sincronización de código fuente en tiempo real.

Tiene una interfaz intuitiva y es de fácil aprendizaje para los desarrolladores. Permite además hacer descripción de los casos de uso proporcionando una gran variedad de plantillas predeterminadas con la posibilidad de personalizarlas.

1.10.1.2 Axure RX pro 5.0

Es una herramienta que ofrece capacidades que se encuentran típicamente en las herramientas de creación de diagramas, como la colocación de arrastrar y soltar, el cambio de tamaño y formato de los *widgets*[2]. La herramienta Axure, posee todos los elementos necesarios para la creación de prototipos amigables. Los prototipos pueden ser exportados como páginas HTML dinámicas, funcionalidad que posibilita intercambiar de forma más amena con los prototipos, observando el flujo consecutivo de los eventos. Todos los prototipos pueden ser configurados para que sus botones se dirijan a la página hacia donde lo harán en el sistema. En general se puede añadir todo el dinamismo que se desee en los componentes de la herramienta, de forma fácil y rápida. Esta característica posibilita al usuario obtener una idea más clara de cómo será la aplicación en un futuro y de esta forma sentirse identificado y motivado con la solución a elaborar.

1.10.2 Entorno de desarrollo (IDE)

Un IDE es una aplicación de software que ofrece servicios integrales a los programadores de computadoras para el desarrollo de software. Está compuesto por un editor de código, un compilador o intérprete, un depurador y un constructor de interfaz gráfica. (21)

[2] Pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor de *widgets.*

1.10.2.1 NetBeans 7.2

Constituye un IDE especialmente diseñado para el desarrollo de aplicaciones en Java, pero que acepta otros lenguajes de programación. Consta de una gran base de usuarios y una comunidad en constante crecimiento, lo que le ha permitido, al igual que muchos otros sistemas, el progreso paulatino de sus prestaciones y la eliminación de errores de programación (*bugs*[3]) que pudiesen existir. Es libre, de código abierto y multiplataforma con soporte integrado para el lenguaje de programación *Java*. (21)

Ventajas del uso de NetBeans:

- Permite que las aplicaciones se desarrollen a partir de un conjunto de módulos o componentes de software.
- Fácil de instalar y de uso instantáneo, se ejecuta en varias plataformas incluyendo Windows y Linux.
- Gestión de la interfaz de usuario (menús y barras de herramientas).
- Gestión de almacenamiento (guardar o cargar algún tipo de dato).
- Biblioteca visual.
- Herramientas de desarrollo integrado.

1.11 Sistema gestor de base de datos

Un gestor de base de datos o sistema de gestión de base de datos es un software que permite el almacenamiento, manipulación y consulta de la información de una base de datos. (22)

1.11.1 PostgreSQL 9.2

Es un Sistema de Gestión de Base de Datos de Objetos Relacional (ORDBMS). Es ampliamente considerado como una de las alternativas de sistemas de base de datos de código abierto reconocido a nivel internacional por su alta confiabilidad, integridad de datos y corrección. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multi-hilos para garantizar la estabilidad del sistema. Es un sistema seguro y puede soportar grandes volúmenes de datos. (22)

Características fundamentales:

- Transacciones: permiten el paso entre dos estados consistentes manteniendo la integridad de los datos.
- Integridad referencial: es utilizada para garantizar la validez de los datos de la base de datos.

[3] Error o fallo en un programa de computador que desencadena un resultado indeseado.

- Bloqueos de tablas y filas: ofrece varios modos de bloqueo para controlar el acceso concurrente de los datos en tablas.
- Restricciones y disparadores (*triggers*): tienen la función de mantener la integridad y consistencia de la base de datos.
- Soporte de tipos y funciones de usuarios: resiste operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario.
- Incorpora una estructura de datos de arreglo.
- Interfaz con diversos lenguajes, entre ellos PHP.

1.12 Servidor de aplicaciones web

Proporciona servicios que soportan la ejecución y disponibilidad de las aplicaciones desplegadas. Aloja y ejecuta aplicaciones que responden a peticiones que llegan por la red desde equipos clientes o desde otras aplicaciones. (23)

1.12.1 Apache server 2.2

Apache es un servidor web de código libre, flexible, rápido, eficiente y continuamente actualizado, cuya implementación se realiza de forma colaborativa, con altas prestaciones y funcionalidades equivalentes a las de los servidores comerciales. Permite crear un servidor http [4]en un ordenador de una forma relativamente rápida y sencilla. Está estructurado en módulos y la configuración de cada uno de ellos se realiza internamente. Los módulos del Apache se pueden clasificar en tres categorías:

- Módulos base: módulo con las funciones básicas de Apache.
- Módulos multiprocesos: son los responsables de la unión con los puertos de la máquina, aceptando las peticiones y enviando a los hijos a atender las peticiones.
- Módulos adicionales: cualquier otro módulo que le añada una funcionalidad al servidor.

1.13 Sistema de control de versiones

Software que mantiene los registros de todos los cambios que se han realizado a los archivos de un software, lo que entre otras cosas permite el trabajo de distintos desarrolladores en un mismo proyecto.

1.13.1 Subversion 1.5

Es un sistema de control de versiones centralizado, libre y empleado en la administración de ficheros y directorios utilizados generalmente en el desarrollo de programas informáticos. Permite además la

[4] Protocolo usado en cada transacción de la web.

gestión de cambios, admite que varios usuarios clientes obtengan copias del proyecto al mismo tiempo. Posibilita examinar el registro histórico de modificaciones realizadas, deshacer los cambios en un momento dado, comparar diferentes versiones de los archivos, unir trasformaciones realizadas por diferentes usuarios en un mismo fichero, así como otras características propias de un sistema de control de versiones.

1.13.2 RapidSVN 0.12

Es un cliente gráfico de *Subversion*, escrito en C++, multiplataforma, posee una interfaz fácil de utilizar, simple para los usuarios principiantes y flexible para los usuarios más experimentados. Se distribuye bajo licencia GNU-GPL y resultó la opción elegida por el equipo de arquitectura pues resulta viable su utilización sobre Linux.

1.14 Arquitectura de software

La arquitectura de software se refiere a un grupo de abstracciones y patrones que brindan un esquema de referencia útil de guía para el desarrollo de software dentro de un sistema informático. Así, los programadores, diseñadores, ingenieros y analistas pueden trabajar bajo una línea común que les posibilite la compatibilidad necesaria para lograr el objetivo deseado.

Esta familia de estilos enfatiza la facilidad de modificación. Son los estilos más generalizados en sistemas de gran escala. Dentro de esta familia se encuentran el Modelo-Vista-Controlador (MVC), la arquitectura orientada a objetos y la arquitectura por capas.

1.14.1 Modelo-Vista-Controlador (MVC)

Fue diseñada para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Su característica principal es que el modelo, las vistas y los controladores se tratan como entidades separadas; esto hace que cualquier cambio producido en el modelo se refleje automáticamente en cada una de las vistas.

Presenta varias ventajas:

- Existe una clara separación entre los componentes de un programa, lo cual permite implementarlos por separado.
- Cuenta con una API (interfaz de programación de aplicaciones) bien definida; cualquiera que use la API, podrá reemplazar el modelo, la vista o el controlador sin dificultades.
- La conexión entre el modelo y sus vistas es dinámica; se produce en tiempo de ejecución y no en tiempo de compilación.

Al incorporar el modelo de arquitectura MVC a un diseño, las piezas de un programa se pueden construir por separado y luego unirlas en tiempo de ejecución. Si se observa que uno de los componentes funciona mal, puede reemplazarse sin que las otras piezas sean afectadas. Este escenario contrasta con la aproximación monolítica típica de muchos programas. Todos tienen un marco que contiene la totalidad de los elementos, un controlador de eventos y la presentación del resultado.

1.14.2 Arquitectura por capas

La división en capas es una de las técnicas más comunes que utilizan los diseñadores de software para separar un sistema complicado. En este esquema, las capas superiores se sirven de los servicios brindados por las inferiores inmediatas, pero estas desconocen de las capas superiores. Entre los beneficios de esta arquitectura se encuentran:

- Puede comprender una sola capa como un todo coherente sin saber de las demás capas.
- Puede sustituir las capas con implementaciones alternativas de los mismos servicios básicos.
- Minimizar las dependencias entre las capas.
- Una vez que tenga una capa construida, se puede utilizar para muchos servicios de nivel superior.
- Mantenibilidad: provee una organización lógica de aplicación y desarrollo.
- Escalabilidad y rendimiento: permite distribuir una aplicación, cada capa puede residir en un computador distinto, además agregar máquinas mejora el rendimiento.
- Seguridad: permite aislar componentes.

1.15 Conclusiones del Capítulo

El estudio del procedimiento Revisión de la materia Laboral permitió fundamentar la selección de la metodología, las herramientas y lenguajes a emplear para obtener un sistema acorde a las necesidades del cliente final. Los sistemas analizados que modelan procesos similares están desarrollados según las características de cada país, demostrando la necesidad de implementar una aplicación adaptable al sistema judicial.

Capítulo 2: Propuesta de Solución

2.1 Introducción

En el presente capítulo se describe la solución propuesta, donde inicialmente se muestran las reglas de negocio y el diagrama de procesos, así como la especificación de los requisitos funcionales y no funcionales. Se analiza la arquitectura del sistema a desarrollar teniendo en cuenta los patrones de diseño que serán aplicados. A partir de los requisitos identificados en la etapa de levantamiento de requisitos del procedimiento Revisión se obtendrán los diagramas de secuencia, de clases del diseño, de paquetes así como el modelo de datos correspondiente. Además se profundiza en cuanto a la seguridad del SITPC en conjunto con las métricas aplicadas para la validación del diseño y los estándares de codificación en los que se apoya el proyecto para realizar la implementación.

2.2 Objeto de informatización

2.2.1 Diagrama de procesos del negocio (BPMN)

Durante la modelación del negocio se obtuvo el siguiente diagrama BPMN correspondiente al procedimiento Revisión de la materia Laboral. (24)

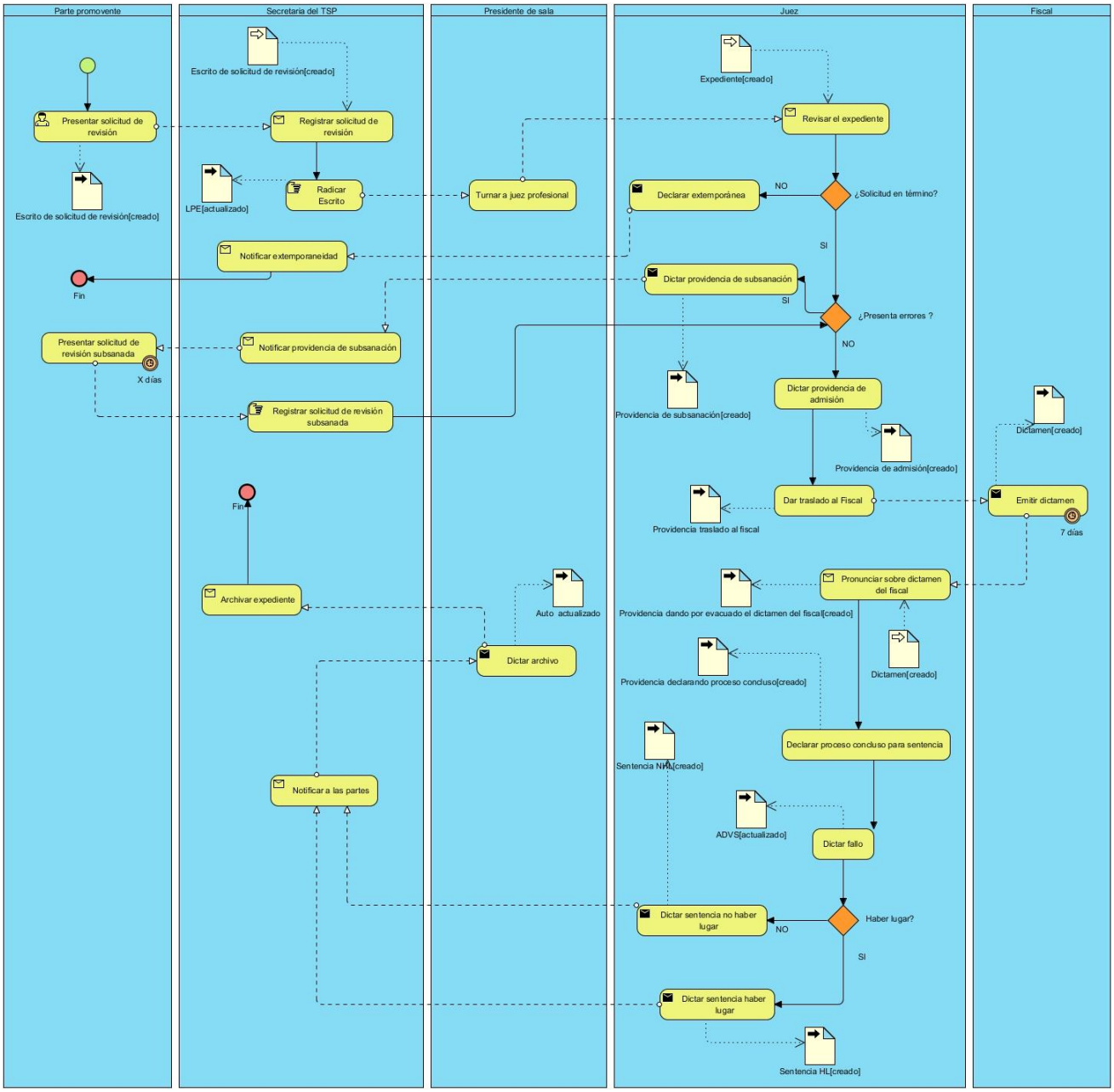


Figura 2. Diagrama de procesos del procedimiento Revisión Laboral

2.2.2 Descripción de las actividades del negocio

El procedimiento Revisión Laboral involucra un conjunto de actividades consecutivas e importantes que se realizan en los TPC para lograr una mejor organización del proceso. A continuación se describen algunas de ellas.

Para obtener más detalles de las actividades del procedimiento Revisión de la materia Laboral consultar el documento “Descripción de procesos de negocio Revisión Laboral” que se encuentra en el repositorio del proyecto.

Tabla 2. Descripción de actividades del negocio

Actividades	Descripción
Presentar solicitud de Revisión	El promovente presenta el Escrito de Solicitud de Revisión ante el TSP.
Registrar solicitud de Revisión	La secretaria del TSP recibe el escrito y lo radica. Se registra el escrito en el Libro de Presentación de Escritos (LPE).
Discutir expediente	Se constituye la sala para deliberar el caso y decidir si se admite o no la solicitud. Se declara inadmisible la solicitud en caso de extemporaneidad o que la medida no sea separación definitiva del cargo. Se declara no haber lugar al procedimiento en caso de que la sentencia dictada sea correcta.
Dar traslado al Fiscal	Se le comunica al fiscal sobre la admisión del procedimiento Revisión, entregándole copia del auto donde se admitió la solicitud.
Emitir dictamen del Fiscal	El Fiscal puede considerar que le asiste, no le asiste o le asiste en parte la razón al promovente, emite los argumentos de su criterio y propone pruebas si lo estima necesario.

2.2.3 Reglas del Negocio

Las reglas del negocio permitieron identificar las restricciones de los conceptos del dominio. Como ejemplo de estas reglas se encuentran las siguientes:

Tabla 3: Descripción de las reglas del negocio

No	Nombre	Descripción
1	Promovente	La revisión solo podrá promoverse por la parte afectada por la improcedencia, ilegalidad, o injusticia notoria en que hubiere incurrido la sentencia.
2	Llegada del escrito de solicitud al TSP	Si el escrito de solicitud llega al TSP después del término definido se dicta auto de extemporaneidad.
3	Días para emitir el Dictamen del fiscal	El fiscal tendrá 7 días a partir de que se le notifique sobre la admisión del procedimiento, para emitir su dictamen sobre el proceso.

Para una mejor comprensión de las reglas del negocio consultar el documento “CEGEL-SITPC Reglas de negocio Revisión-Laboral” que se encuentra en el repositorio del proyecto.

2.3 Especificación de los Requisitos de software

2.3.1 Técnicas para la obtención de requisitos

Para la obtención de los requisitos fueron empleadas las técnicas que a continuación se describen:

Tormenta de ideas: fueron realizados talleres y debates entre el equipo de analistas del proyecto, con el objetivo de generar ideas en un ambiente libre de críticas o juicios. Posibilitó generar variedad de vistas del problema y que fuera modelado de diferentes formas, obteniendo de esta manera el modelado del negocio propuesto anteriormente.

Entrevistas: se efectuaron diferentes encuentros comunicativos con jueces y secretarias donde quedaron esclarecidas un conjunto de funcionalidades de importancia para el cliente final.

Observación: se realizaron visitas a diferentes tribunales, observando cada detalle relacionado con el modo de trabajo y las actividades que allí se realizan.

2.3.2 Requisitos funcionales del procedimiento Revisión Laboral

Realizada la modelación del negocio y previo a iniciar el diseño e implementación del procedimiento Revisión Laboral fueron identificados 26 RF descritos en la tabla 4 que a continuación se presenta:

Tabla 4. Descripción de los requisitos funcionales

No.	Nombre	Descripción	Prioridad para el cliente	Complejidad
1	Adicionar documento en formato duro	El sistema permitirá registrar el nombre de un documento en formato duro.	Media	Media
2	Adicionar documento en formato digital	El sistema permitirá registrar un documento en formato digital.	Media	Baja
3	Eliminar documento en formato duro	El sistema permitirá eliminar un documento en formato duro.	Baja	Baja
4	Eliminar documento en formato digital	El sistema permitirá eliminar un documento en formato digital.	Baja	Baja
5	Registrar escrito de solicitud de revisión	El sistema permitirá registrar el escrito de solicitud de revisión presentado.	Alta	Alta
6	Visualizar datos primarios	El sistema permitirá observar los datos más importantes.	Baja	Media
7	Listar documentos necesarios	El sistema permitirá listar los documentos necesarios para el usuario.	Baja	Media
8	Crear auto de extemporaneidad de la solicitud	El sistema permitirá crear el auto definitivo rechazando el escrito subsanado por presentarse fuera de término. Se le notifica al promovente.	Alta	Alta

9	Crear auto de admisión de la solicitud de revisión	El sistema permitirá crear el auto de Admisión en caso de admitirse el proceso de Revisión. Se notifica a las partes.	Alta	Alta
10	Crear providencia de subsanación de la solicitud de revisión	El sistema permitirá crear la providencia ordenando la subsanación del escrito de solicitud. Se notifica al promovente.	Alta	Alta
11	Seleccionar término de subsanación	El sistema permitirá seleccionar el término para entregar el escrito subsanado.	Alta	Media
12	Adicionar motivo de subsanación	El sistema permitirá adicionar nuevos motivos por los cuales realizar una subsanación.	Media	Baja
13	Eliminar motivo de subsanación	El sistema permitirá eliminar algunos de los motivos de subsanación.	Baja	Baja
14	Registrar escrito subsanado	El sistema permitirá registrar el escrito subsanado.	Alta	Alta
15	Crear providencia declarando conclusa la revisión	El sistema permitirá crear la providencia declarando el proceso concluso para sentencia.	Alta	Alta
16	Registrar informe emitido por el fiscal	El sistema permitirá registrar el informe emitido por el fiscal.	Alta	Alta

17	Crear providencia teniendo por recibido el dictamen del fiscal	El sistema permitirá crear la providencia dando por evacuado el trámite del fiscal.	Alta	Alta
18	Crear Acta de Discusión y Votación de Sentencia	El sistema permitirá crear el Acta de Discusión y Votación de Sentencia.	Alta	Alta
19	Declarar no haber lugar al procedimiento Revisión	El sistema permitirá declarar no haber lugar al procedimiento Revisión. Se notifica al promovente.	Alta	Media
20	Crear sentencia	El sistema permitirá crear la sentencia. Se notifica a las partes.	Alta	Alta
21	Crear sentencia de no haber lugar	El sistema permitirá crear la sentencia de no haber lugar si se declaró no haber lugar al procedimiento Revisión. Se notifica al promovente.	Alta	Alta
22	Disponer archivo del expediente	El sistema le permitirá al juez disponer sobre el archivo del expediente.	Alta	Media
23	Crear auto disponiendo el archivo de las actuaciones	El sistema permitirá crear auto disponiendo el archivo de las actuaciones.	Alta	Alta
24	Visualizar expediente	El sistema permitirá visualizar caratula de un expediente.	Baja	Media

25	Visualizar documento	El sistema permitirá mostrar el listado de documentos del expediente.	Baja	Baja
26	Buscar expediente archivado	El sistema permitirá buscar un determinado expediente.	Media	Media

2.3.3 Requisitos no funcionales

El equipo de arquitectura del proyecto SITPC identificó 45 requisitos no funcionales (RNF) agrupados en los siguientes indicadores: usabilidad (7), confiabilidad (7), eficiencia (4), soporte (2), restricciones de diseño (3), documentación (3), interfaz (3), seguridad (14) y despliegue (2). Para obtener más detalles de los RNF consultar documento “CEGEL-SITPC-0113-Requisitos No Funcionales-v1.3” que se encuentra el repositorio del proyecto. (25)

2.4 Modelado del sistema

2.4.1 Definición de los actores del sistema

En la descripción del procedimiento Revisión Laboral intervinieron dos actores (Registrador y el Juez ponente) descritos en la siguiente tabla.

Tabla 5. Definición de actores del sistema

Actor	Objetivo
Registrador	Actor encargado de registrar los distintos escritos que presentan las partes del proceso.
Juez ponente	Actor encargado de disponer sobre la solicitud de revisión, ordenando la admisión, subsanación o rechazo en caso de que se presente fuera del término establecido.

2.4.2 Diagrama de Casos de Uso del Sistema

Un caso de uso es una secuencia de transacciones desarrolladas por un sistema, dando respuesta a un evento que inicia un actor sobre el propio sistema. Los casos de uso comprenden los pasos

necesarios para alcanzar un objetivo propuesto de su actor principal. La figura 3 muestra el diagrama de casos de uso del sistema. (26)

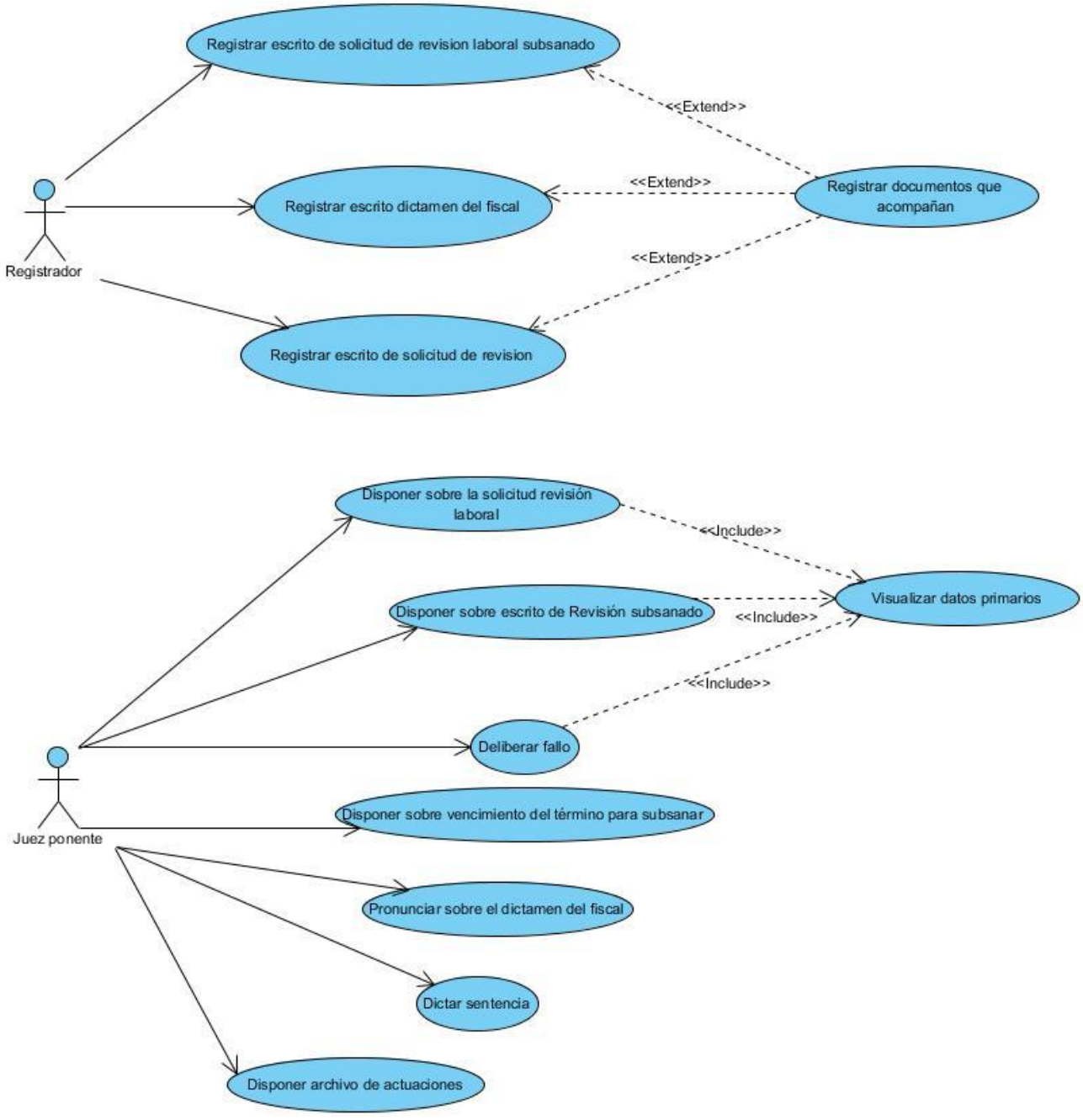


Figura 3: Diagrama de casos de uso

2.4.2.1 Patrones de casos de uso del sistema

Los patrones de casos de uso describen el comportamiento que debe existir en el sistema, ayudan a representar qué es lo que el sistema debe hacer. Además capturan mejores prácticas para modelar casos de uso de manera que sea mantenible, reusable y entendible.

Para el desarrollo de la solución propuesta fueron empleados los siguientes patrones de casos de uso:

Include: cuando se relacionan dos casos de uso con un "include", se refiere a que el primero (el caso de uso base) incluye al segundo (el caso de uso incluído). Es decir, el segundo es parte esencial del primero, de forma que sin el segundo, el primero no podría funcionar bien; pues no podría cumplir su objetivo. En el sistema el caso de uso incluido o caso base es el "Visualizar datos primarios". (28)

Extend: no es indispensable que ocurra el caso de uso extendido, y cuando lo hace ofrece un valor extra (extiende) al objetivo original del caso de uso base. En el sistema se evidencia con el caso de uso "Registrar Documentos que acompañan". (28)

La aplicación de estos patrones proporciona los siguientes beneficios:

- Aumentar la productividad.
- Evitar el doble trabajo por errores.
- No invertir tiempo en resolver problemas ya resueltos.

2.4.2.2 Especificación de los Casos de Uso del Sistema

A continuación se muestra la descripción de los casos de uso "Registrar solicitud de Revisión". El resto de los casos de uso se encuentran descritos en el documento "Descripción de casos de uso del proceso Revisión Laboral" ubicado en el repositorio del proyecto SITPC.

Tabla 6. Especificación del caso de uso Registrar Solicitud de Revisión

Registrar Solicitud de Revisión		
Objetivo	Registrar en el sistema un escrito de solicitud de revisión.	
Actores	Registrador: (Inicia) registra en el sistema los datos de un escrito de solicitud.	
Resumen	Comienza con la solicitud de revisión de la parte afectada por una resolución que se dictó en el expediente. Termina con la creación y registro del escrito de solicitud de revisión.	
Complejidad	Alta	
Prioridad	Alta	
Precondiciones	El usuario debe estar autenticado con los permisos necesarios.	
Postcondiciones	Se registró una solicitud. Se turnó el expediente, se radicó y se actualizó el LPE.	
Flujo de eventos		
Flujo básico Registrar escrito de Revisión		
	Actor	**Sistema**
1	Selecciona en el menú izquierdo la opción “Registrar solicitud”.	
2		Muestra un formulario para buscar si existe el expediente, teniendo como criterios de búsqueda: el número de identificación del mismo y el año.
3	Inserta los datos solicitados.	
4		El sistema procesa los datos para localizar el expediente y muestra los resultados de la búsqueda.
5	Selecciona el expediente al que se le realizará la revisión.	
6		Muestra una interfaz donde se listan los datos primarios del expediente:

		• Número de expediente • Juez ponente • Demandante(s) • Demandado(s)
7	Selecciona “Abogado que se está personando”.	
8		Muestra una interfaz para seleccionar el abogado que presenta el escrito.
9	Selecciona el abogado.	
10		Habilita un componente para seleccionar el tipo de escrito a presentar.
11	Selecciona el tipo de escrito "Solicitud de Revisión"	
12		Presenta las partes del proceso y las resoluciones generadas en el expediente objeto de revisión.
13	Selecciona quién solicita la revisión.	
14	Selecciona la resolución a revisar.	
15	Inserta los datos	
16	Selecciona registrar	
17		Valida que los datos introducidos son correctos, que no existan campos obligatorios vacíos y muestra la interfaz inicial. Termina el caso de uso.
Flujos alternos		
“Selecciona presentado por”		
	Actor	**Sistema**
7.1	Selecciona “Otro”.	

7.2		Muestra una interfaz para seleccionar una persona natural que presenta el escrito.
7.3	Selecciona una persona.	
7.4	Hace clic en “seleccionar”. Continúa el flujo normal de eventos en el paso 10.	
“Selecciona Cancelar”		
	Actor	**Sistema**
16.1	Selecciona la opción “cancelar”.	
16.2		Muestra el mensaje de confirmación ¿Está seguro que desea cancelar la operación? • Aceptar • Cancelar
16.3	Selecciona la opción “Aceptar”.	
16.4		Redirecciona hacia la pantalla inicial. Termina el caso de uso.
“Selecciona Cancelar”		
	Actor	**Sistema**
16.1	Selecciona la opción “Cancelar”.	
16.2		Regresa al paso 16 del flujo básico.
Los datos son incorrectos y/o existen campos obligatorios vacíos.		
	Actor	**Sistema**
1	Introduce datos incorrectos y/o existen campos obligatorios vacíos.	
2		Indica que los datos son incorrectos y/o existen campos obligatorios vacíos, no guarda los cambios.

3		Regresa al paso 5 del flujo básico Registrar escrito de solicitud.

2.4.2.3 Prototipo de interfaz del sistema

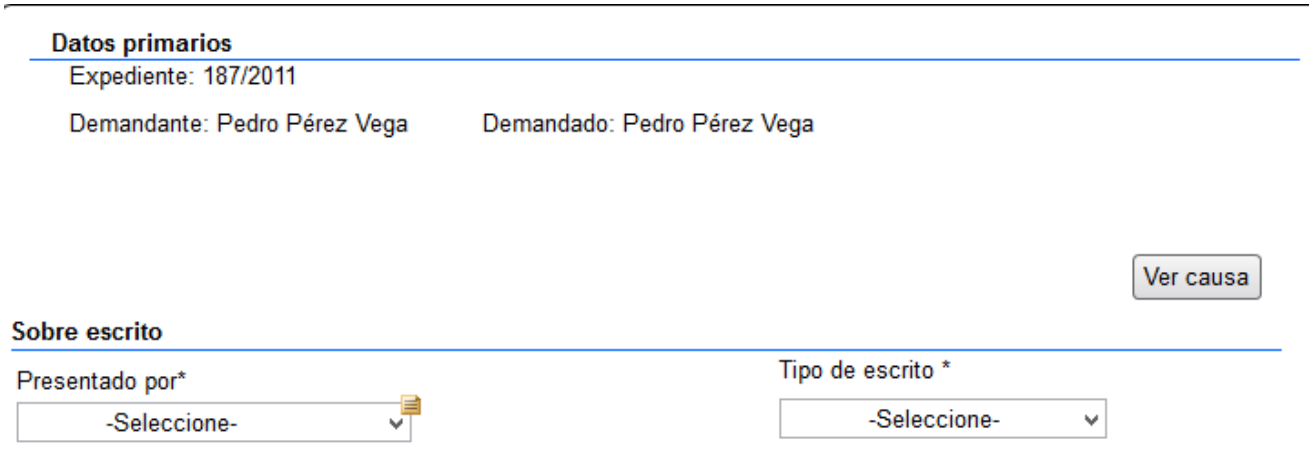


Figura 4: Prototipo del CU Registrar Escrito

2.5 Modelo Conceptual del Módulo

2.5.1 Arquitectura

Durante el desarrollo del sistema propuesto se emplea una arquitectura basada en capas que a su vez hace uso del patrón Modelo-Vista-Controlador, dicho patrón se encuentra incluido dentro de los estilos arquitectónicos de llamada y retorno. Además adquiere cierto aislamiento ya que se pueden realizar actualizaciones en el interior de las capas sin afectar el resto del sistema, de manera que si existe un error o es necesario realizar alguna modificación, solo se realizará el cambio en la capa correspondiente mientras que el resto del sistema se mantiene inmune. Esta arquitectura se encuentra apoyada principalmente en el marco de trabajo Symfony 2, debido a que brinda una estructura de paquetes que define correctamente las clases que forman las partes del modelo, la vista y el controlador respectivamente.

La imagen que se muestra a continuación representa la arquitectura del sistema.

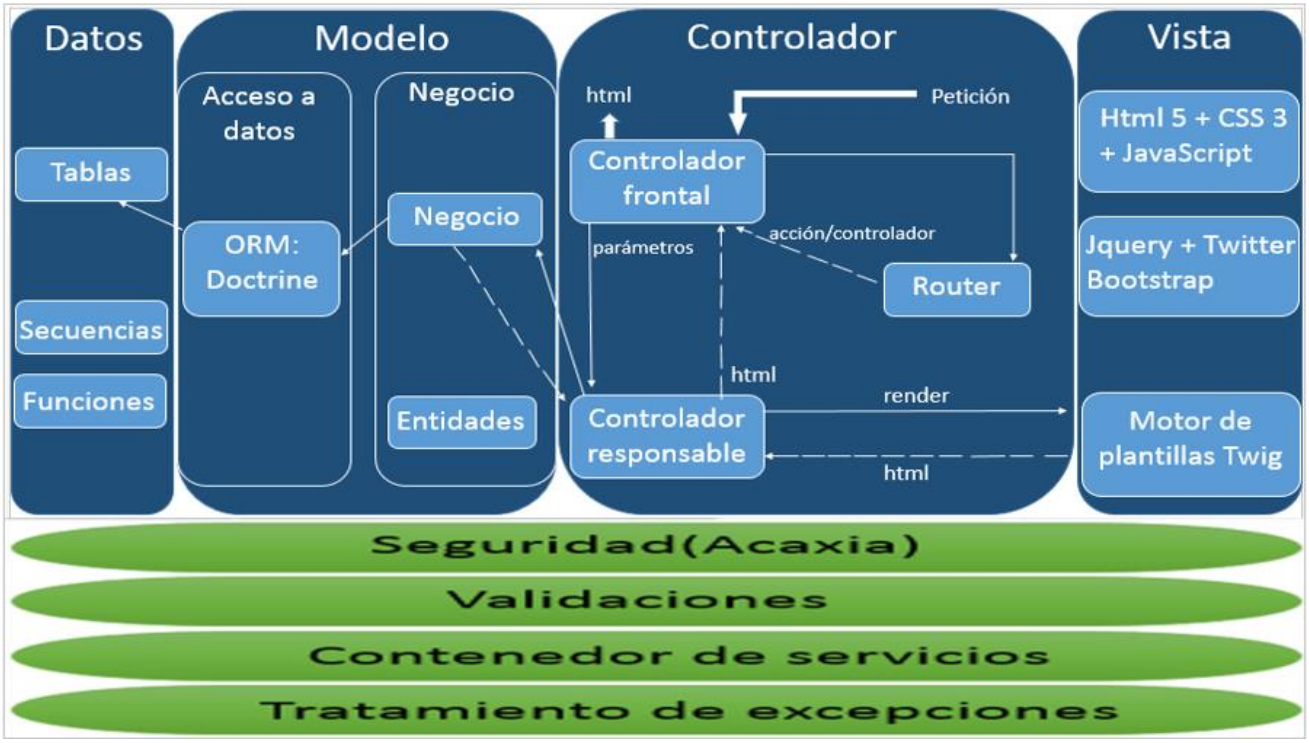


Figura 5: Arquitectura del sistema

Vista

La vista es la intermediaria entre los usuarios y el sistema. Todas las vistas e interfaces del sistema se encuentran en esta capa, deben ser entendibles y amigables, usando para ello imágenes, funciones *JavaScript* y estilos CSS. Las vistas son monitorizadas por el motor de plantillas Twig ubicado en la capa controlador, donde se define una plantilla base ubicada en *"SIT/app/Resources/views"* que en conjunto con otras se encargan de transformar la información del modelo para publicárselo al usuario final. Se muestra en la figura 6 la composición de la capa de las vistas del sistema.

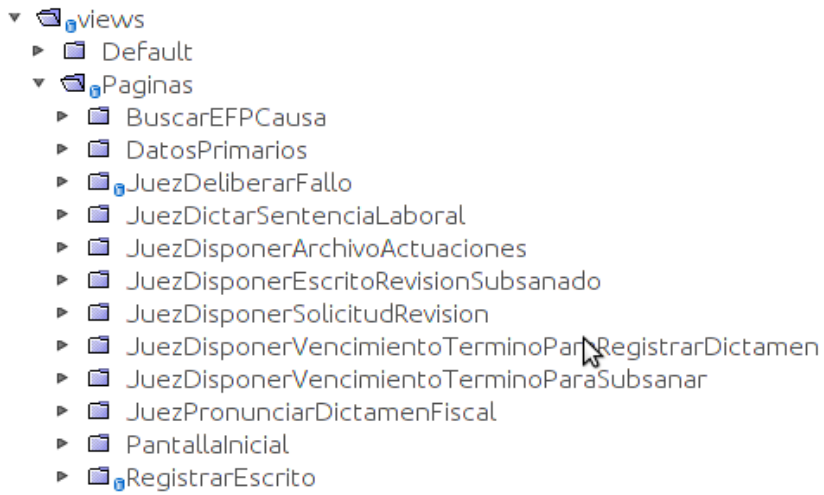


Figura 6: Carpeta contenedora de las Vistas

Controlador

Es el encargado de facilitar respuesta a cada una de las peticiones ejecutadas por el usuario, para mostrar y modificar datos en el modelo. En la estructura de paquetes que propone Symfony 2 existe una dirección donde se deben ubicar todas las clases controladoras. En el SITPC todas las controladoras se encuentran en la siguiente dirección: *"SIT/LABORAL/RevisionLaboralBundle/Controller"*. La figura 7 muestra la estructura de carpetas Controller.

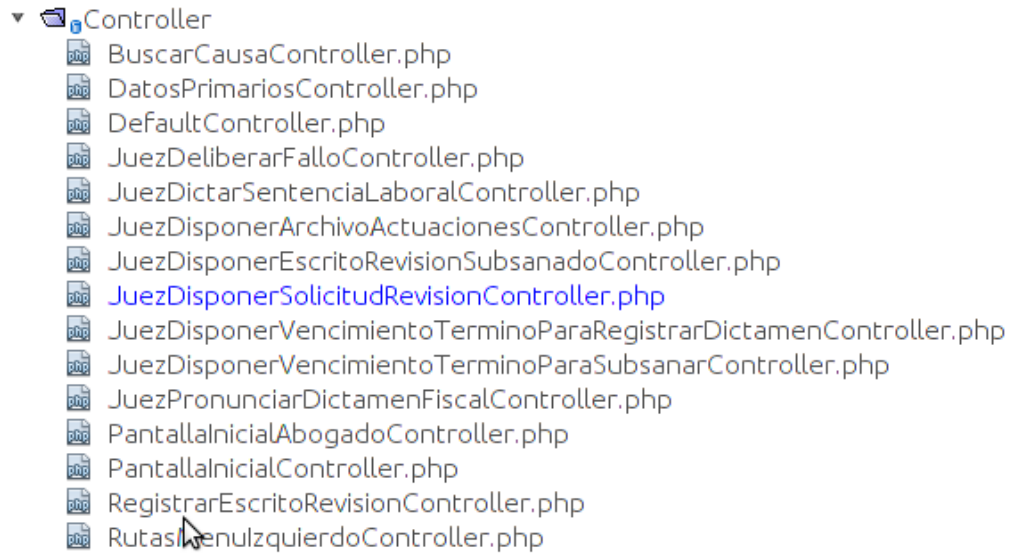


Figura 7: Carpeta de las Controladoras

Modelo

El modelo es quien representa los datos del programa. Maneja los datos y controla todas sus transformaciones. No tiene conocimiento específico de los controladores o de las vistas, tampoco contiene referencias a ellos.

En la capa de negocio se encuentran las entidades las cuales sirven de mediadoras entre las clases controladoras y las clases gestoras. Las clases gestoras están ubicadas en la siguiente dirección: "SIT/src/SIT/SRV/ComunesBundle/Negocio/Gestor". Además se encuentran las entidades del dominio que representan las tablas de la base de datos ya mapeadas por Doctrine, pueden encontrarse en *"SIT\vendor\Base\ComunBundle\Entity".*

La capa de acceso a datos está comprendida por el ORM Doctrine, este tiene la tarea de acceder a los datos almacenados. En esta capa, además, se encuentran las clases *Repository* donde se definen las consultas para acceder a los datos del modelo, dichas clases están ubicadas en "SIT/vendor/Base/ComunBundle/Repository". El controlador se comunica con las clases gestoras las que interactúan con los datos a través del *Entity Manager.* Las gestoras se comunican con las clases *Repository* encargadas de apartar el código de las consultas. En la figura 8 se muestra la estructura de las carpetas Gestora y Resources.

Figura 8: Carpetas Resources y Gestoras

2.5.2 Patrones de Diseño

Proporcionan una estructura conocida para todos los desarrolladores, de forma que la manera de trabajar entre ellos no resulte distinta. La utilización de patrones de diseño permite tener una

estructura de código común para todos los proyectos que implementen una funcionalidad genérica, permiten ahorrar gran cantidad de tiempo en la construcción de software y de esta forma el producto final será más fácil de comprender, mantener y extender. (29)

Experto

Patrón encargado de delegar responsabilidades, suele usarse en el diseño orientado a objetos. Con la utilización de este patrón se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se le pide.

En la solución propuesta se utiliza este patrón principalmente en las clases entidades como *PersonaJuridica* ya que contiene toda la información relacionada con una persona jurídica. En la siguiente imagen se muestra un ejemplo del uso de este patrón en la aplicación.

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

Figura 9: Patrón Experto perteneciente a la clase DatosPrimariosController

Creador

Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento.

En la solución que se propone se manifiesta dicho patrón en la clase controladora JuezDisponersolicitudRevisionController la cual es idónea para asumir la responsabilidad de crear instancias de la clase JuezDisponersolicitudRevisionType. La figura 10 muestra el uso de este patrón en el sistema.

```
public function indexAction() {
    $idTramite= $this->getRequest()->get('idTramite');

        $modelo = $this->get('laboral.rev.tm.MotivosSubsanacionTM');
        $abogado = true;
    $modelo = $this->get('laboral.rev.tm.MotivosSubsanacionTM');
    $form= $this->createForm(new JuezDisponerSolicitudRevisionType());
    return $this->render('RevisionLaboralBundle:Paginas\\JuezDisponerSolicitudRevision:Jue\DisponerSolicitudRevis
                'idTramite' => $this->getRequest()->get('idTramite'),
                'abogado' => $abogado,
```

Figura 10: Patrón Creador perteneciente a la clase JuezDisponerSolicitudController

Controlador

Fue empleado en las clases controladoras. Estas se encargan de recibir los datos introducidos por el usuario en el sistema y enviarlos a las distintas clases en dependencia de lo que se desee realizar. Esto permite lograr una separación entre la capa de presentación y la capa de negocio, aumentando la reutilización de código y brindando un mayor control sobre los cambios. En la solución se emplea un controlador de casos de uso, ya que existe un controlador por cada caso de uso del sistema, brindando soporte al mismo. La figura 11 muestra los controladores del sistema por cada caso de uso.

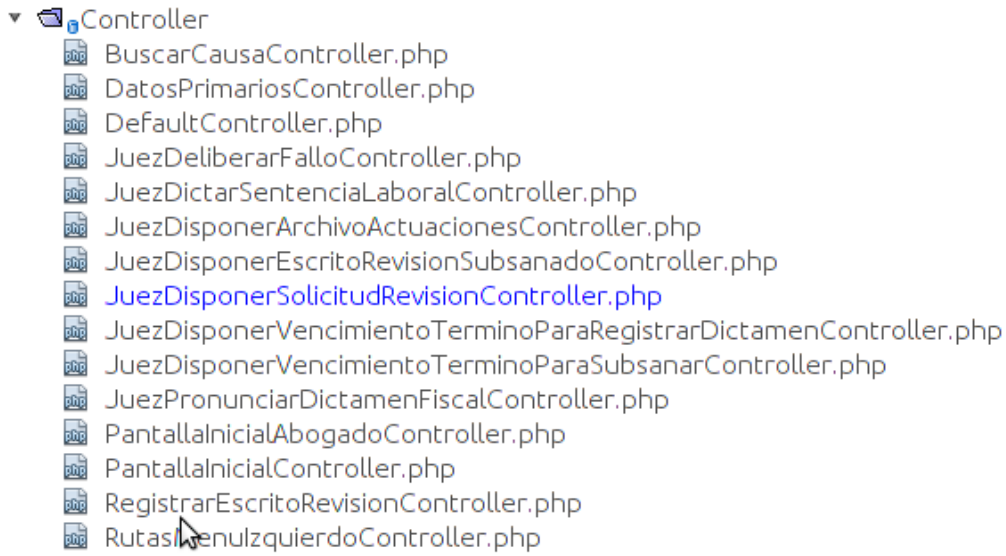


Figura 11: Patrón Controlador

Bajo acoplamiento

Este patrón soporta el diseño de clases más independientes, que reducen el impacto de los cambios y acrecientan la oportunidad de mayor productividad. En el sistema las entidades son las clases más reutilizadas y en la arquitectura se puede apreciar que la vista se comunica con la controladora, esta a su vez con las clases gestoras que se comunican con las entidades, de forma que se puede concluir que las entidades no tienen asociación con la vista ni el controlador. En la siguiente imagen se muestra como la clase controladora asigna responsabilidades, de manera que no tenga que ejecutarlas ella misma y favoreciendo así al bajo acoplamiento.

```
public function guardarVisorAction()
{
    $request = $this->getRequest();
    $fallo = $this->recuperar('fallo');
    $html = $this->getRequest()->get('html');
    $this->getGestor()->visorGuardar($html, $this->recuperar('idT
    return new Response(1);
}
```

Figura 12: Patrón Bajo Acoplamiento

Alta cohesión

El uso de este patrón es importante para la solución que se desea lograr, debido a que es el encargado de asignar a una clase responsabilidades moderadas en un área funcional y colabora con las demás para llevar a cabo las tareas, es decir, se basa en lograr que una clase única solo sea responsable de las funcionalidades que necesite, no así de una gran cumulo de responsabilidades. En la imagen que se presenta a continuación se muestra el uso de este patrón.

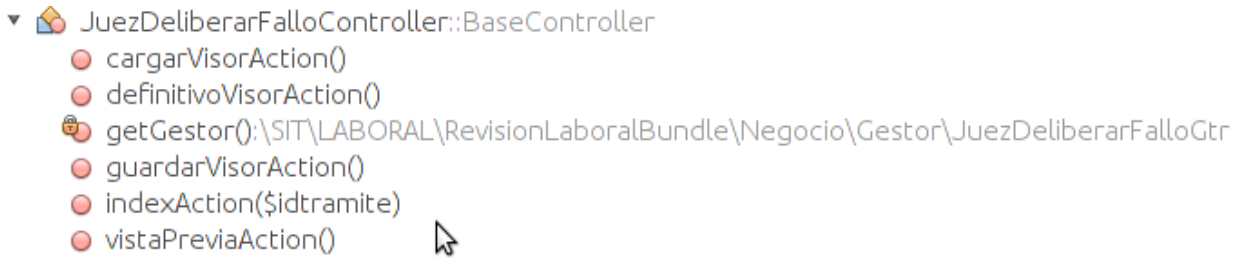


Figura 13: Patrón Alta Cohesión

Decorador

Este patrón es utilizado en el trabajo con plantillas, las cuales son manipuladas utilizando el motor de plantillas Twig e implementando una herencia entre ellas dentro de la aplicación. De tal forma que se puede crear una plantilla base que contiene todos los elementos comunes del sistema y definir los bloques que cada una de las descendientes puede modificar.

En el sistema la plantilla base define un HTML básico que puede ser empleado para una página de dos columnas (menú lateral izquierdo y contenido), además de poseer otros bloques que son comunes para todos los subsistemas y módulos, como son: el banner, el menú superior y el pie de página.

2.6 Inyección de dependencia

Symfony 2 posee un generador de *bundles* que crea el fichero service.yml. En este fichero se detallan todos los servicios del *bundle* en el que se encuentre y sus dependencias. El fichero se compone de dos secciones, el *parameters* donde se declaran los parámetros de configuración de cada servicio y el *service* donde se declara cada servicio.

La clase Controller *JuezDisponerVencimientoTerminoParaSubsanar* que se encuentra en "SIT\LABORAL\RevisionLaboralBundle\Controller" posee el atributo *container* que es una instancia del contenedor de dependencias. De esta forma desde cualquier clase descendiente de esta controladora se puede instanciar cualquier servicio existente en la aplicación a través del método get () del *container*. La figura 14 muestra un ejemplo de cómo se realiza dicha operación.

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

Figura 14: Método get() de la clase JuezDisponerVencimientoTerminoParaSubsanar

Esto se evidencia en la aplicación cuando se va a acceder a las clases gestoras. Las clases controladoras le piden al *container* [5]un servicio determinado y este accede al archivo de configuración del bundle.

2.7 Modelo de Diseño

2.7.1 Diagrama de Clases del Diseño

Los diagramas de clases del diseño representan las clases que participan en el sistema con sus relaciones estructurales y de herencia. Los diagramas de este tipo contienen las definiciones de las entidades del software en vez de conceptos del mundo real y están compuestos por los siguientes elementos: (30)

Clases: atributos, métodos y visibilidad.

Relaciones: herencia, composición, agregación, asociación y uso.

[5] Contenedor de servicios

La figura 15 muestra el diagrama de clases del diseño correspondiente al caso de uso “registrar escrito de solicitud”, el resto de los diagramas pueden ser observados en los anexos del presente documento.

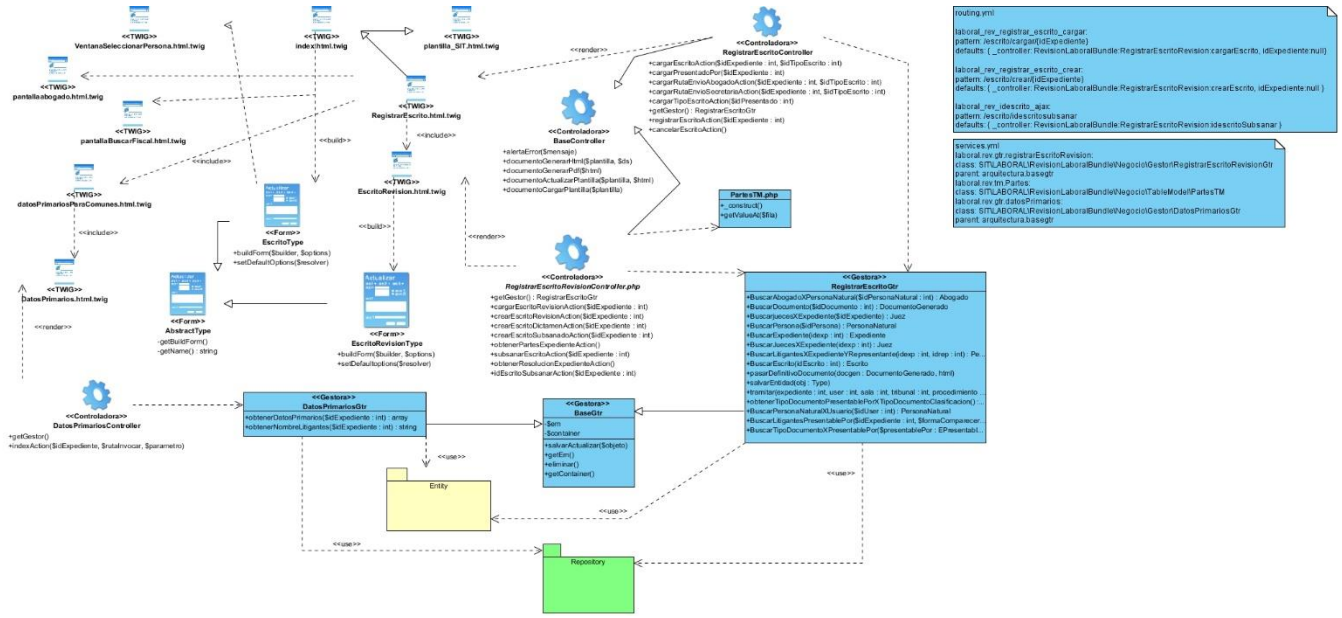


Figura 15.Diagrama de clases del Diseño

2.7.2 Diagrama de Secuencia

Los diagramas de secuencias en UML muestran la forma en que los objetos se comunican entre sí al transcurrir el tiempo. Dichos diagramas manifiestan los objetos con sus “líneas de vida”, participando en la interacción y la secuencia de mensajes intercambiados entre objetos en una secuencia ordenada (31). El diagrama de secuencia correspondiente al caso de uso Registrar Escrito de Revisión se encuentra en el anexo 8.

2.7.3 Diagrama de Paquetes

Los sistemas de gran envergadura deben ser divididos en unidades más pequeñas, de modo que los roles involucrados puedan trabajar con una limitada cantidad de información. Los paquetes contienen elementos del modelo al más alto nivel, como son las clases y sus relaciones. En la figura 16 se muestra el diagrama de paquetes. (32)

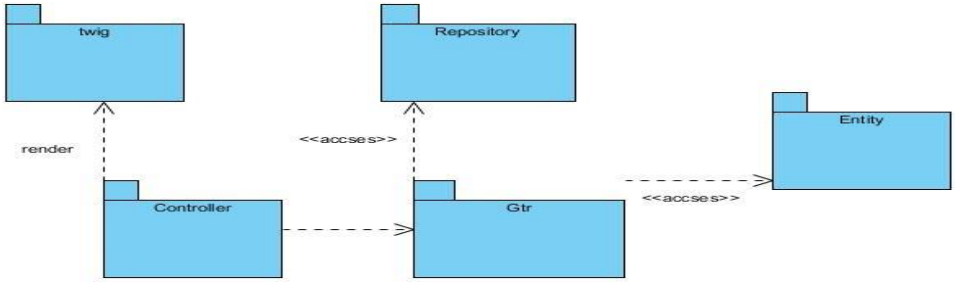


Figura 16: Diagrama de Paquetes

2.7.4 Modelo de Datos

El diagrama de modelo lógico de datos o diagrama de clases persistentes muestra las clases capaces de mantener su valor en el tiempo y el espacio. Para la confección de dicho modelo fueron empleados algunos patrones descritos a continuación: (33)

Llave subrogada: es un campo numérico de una tabla cuyo único requisito es almacenar un valor numérico secuencial e incremental de las tablas.

Modelo entidad-atributo-valor: es un acercamiento al modelo orientado a objetos personificado en el modelo relacional, donde se pueden representar objetos con sus atributos.

La base de datos que se presenta se encuentra en tercera forma normal, aunque hay partes que se dejaron en segunda forma normal por aspectos de rendimiento y negocio, ejemplo de ello son los Escritos y Documentos. En la figura 17 se evidencia el uso de los patrones descritos anteriormente.

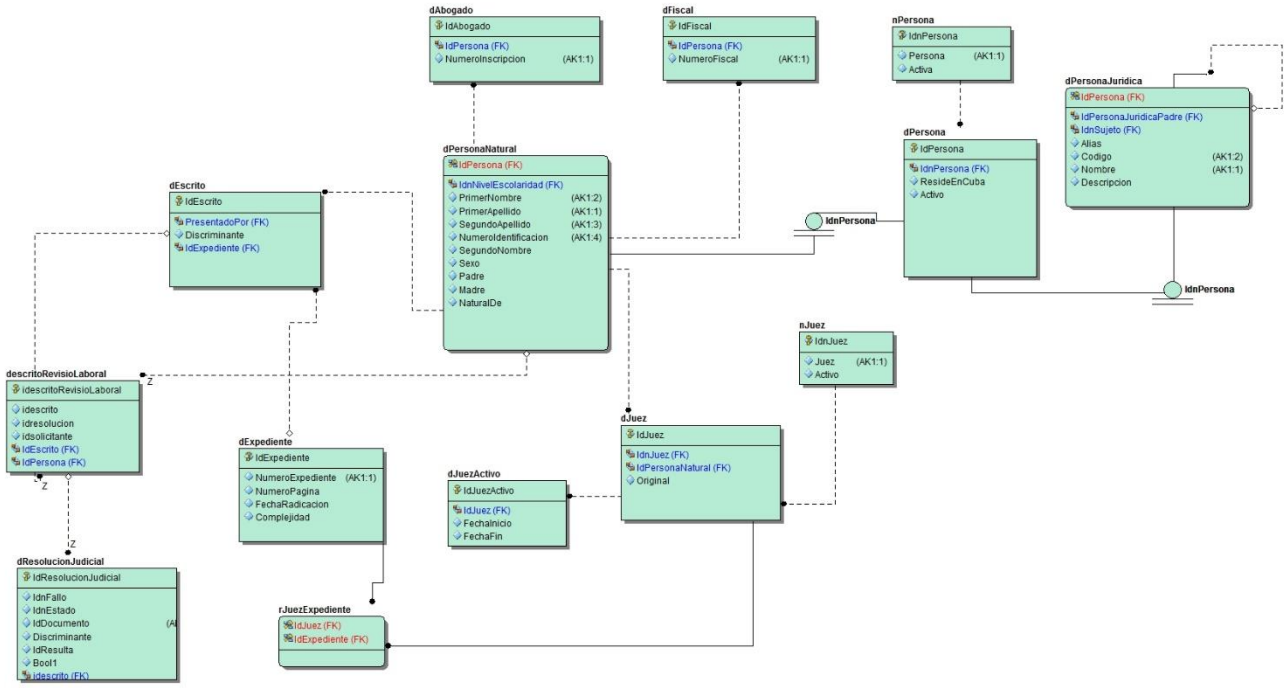


Figura 17: Diagrama de modelo de datos

2.8 Conclusiones del Capítulo

La construcción de la solución propuesta a través de los artefactos descritos: diagrama de casos de uso, diagrama de clases del diseño y diagrama de paquetes, facilitó su interpretación e implementación. El diagrama de modelo de negocio sirvió como guía para las actividades de implementación, al representar las funcionalidades del software a desarrollar. Los patrones de diseño empleados en el desarrollo del sistema proporcionaron una guía para los desarrolladores y la obtención de una estructura de código común.

Capítulo 3: Implementación y pruebas del sistema

3.1 Introducción

En este capítulo se muestra la validación de la solución propuesta, con el fin de comprobar los resultados obtenidos una vez realizado el diseño, además se muestran los artefactos generados durante la implementación del sistema. Se presentan los resultados de las pruebas realizadas y la validación de las variables de la investigación, así como el diagrama de despliegue que refleja el comportamiento del sistema en el TSP.

3.2 Estándares de codificación

Para el desarrollo del sistema se utilizó el estándar de codificación que integra el lenguaje de programación PHP en conjunto con los definidos en los ejemplos que a continuación se presentan. Existen otros estándares evidenciados en el documento CEGEL-CITPC-Estándares de Codificación del expediente de proyecto SITPC.

Llamada a funciones:

Deben llamarse las funciones sin los espacios entre el nombre de la función, el paréntesis de la apertura y el primer parámetro; los espacios entre las comas y cada parámetro, ningún espacio entre el último parámetro, el paréntesis del cierre y el punto y coma. En la figura 18 se muestra un ejemplo.

```
$plantilla = $this->DocumentoCargarPlantilla($nombrePlantilla);
```

Figura 18: Llamada a funciones

Siempre incluir las llaves.

En todo momento a la hora de codificar un bloque de instrucciones debe ir encerrado entre llaves, aun cuando conste de una sola línea. Un correcto ejemplo se muestra en la figura 19.

```
public function getAsunto()
{
    return $this->asunto;
}
```

Figura 19: Incluir Llaves

3.3 Tratamiento de errores

Las situaciones que puedan provocar fallos en los sistemas informáticos se denominan excepciones. El sistema propuesto presenta una interfaz diseñada, implementada y dirigida a evitar tales situaciones y errores. Dicho software tiene la misión de detectar problemas en cuanto a control de acceso (autenticación) por parte de algún usuario, cuenta con las validaciones necesarias para evitar que sean insertados datos erróneos o que no sean insertados datos en campos de carácter obligatorio. Además controla el acceso a páginas restringidas mediante variables de sesión que brinda el lenguaje PHP. Las excepciones se realizan mediante mensajes de error con textos sencillos y de fácil comprensión para el usuario.

3.4 Descripción de la seguridad del sistema

Seguridad: Capacidad de un software para proteger la información y los datos, de tal forma que personas o sistemas no autorizados no puedan leer o modificar los mismos, mientras que personas o sistemas autorizados tengan el acceso a ellos. (34)

En el SITPC mediante la utilización del sistema Acaxia se realiza la gestión de los usuarios, debido a que este permite gestionar el control de acceso de varias aplicaciones de forma simultánea en entornos multidominios. Existen en la arquitectura del sistema nueve módulos fundamentales que permiten garantizar el control de acceso a sí mismo y a otros sistemas que formen parte de su dominio de aplicación.

Entre los principales módulos se encuentran:

- **Gestión de organizaciones y dominios:** tiene la responsabilidad de proveer los mecanismos necesarios para que se establezcan desde el inicio las restricciones y relaciones jerárquicas entre los distintos tipos de organizaciones y los atributos que se van a gestionar por cada una de ellas.
- **Gestión de servidores:** este módulo brinda la posibilidad de configurar los servidores de base de datos y de autenticación que van a utilizar cada uno de los sistemas para la gestión de los datos e identificación y autenticación de los usuarios respectivamente.
- **Gestión de sistemas:** se encarga de gestionar las estructuras de los a los cuales se necesita proveer seguridad.
- **Gestión de roles y usuarios:** tiene la responsabilidad de establecer la mayor parte de las políticas de acceso, apoyándose en las configuraciones realizadas en los demás módulos.

3.5 Diagrama de despliegue del sistema en el TSP

Los diagramas de despliegue muestran las relaciones físicas que existen entre los distintos nodos que componen un sistema, además del reparto de los componentes sobre dichos nodos. A continuación se muestra la figura 20 que describe cómo quedará el sistema una vez desplegado en el TSP. (35)

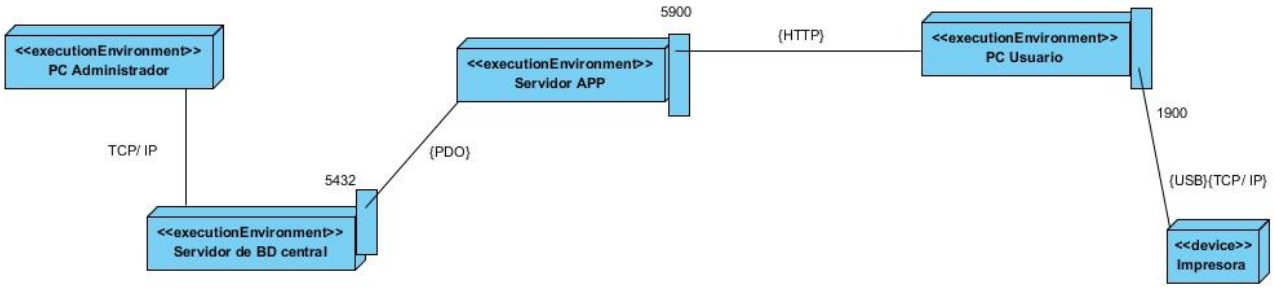


Figura 20: Diagrama de despliegue en el TSP

3.6 Validación de requisitos

Mediante la validación de requisitos se pretende mostrar que estos realmente definen el sistema que el cliente desea. Coincide parcialmente con el análisis ya que este implica encontrar problemas con los requisitos. Este proceso es importante debido a que los errores en el documento de requisitos pueden conducir a importantes costes al repetir el trabajo cuando son descubiertos durante el desarrollo o luego de que el sistema esté en uso. El coste de arreglar un problema en los requerimientos haciendo un cambio en el sistema es mucho mayor que reparar los errores de diseño o los de codificación. La razón de esto es que un cambio en los requisitos normalmente significa que el diseño y la implementación del sistema también deban cambiar y que se proceda nuevamente a la fase de pruebas. (36)

Durante esta etapa el equipo de revisión del proyecto SITPC compuesto por analistas, desarrolladores, clientes y usuarios examinaron la especificación en busca de errores en el contenido o la interpretación, áreas que pudieran requerir de clarificación, información faltante e inconsistencias.

3.6.1 Métricas para requisitos

Los requisitos del software son la base de las medidas de la calidad. En la disciplina Requisitos se tuvo en cuenta la métrica para medir su estabilidad, especificidad y grado de validación.

2 Aplicación de la métrica Estabilidad de requisitos:

Teniendo en cuenta que fueron identificados un total de 26 requisitos funcionales, de los cuales 5 resultaron modificados se calcula:

ETR = [(26– 5) / 26] * 100= 80,76

Como resultado se obtuvo un valor de 80,76. Esta cifra demuestra que no se han realizado cambios significativos sobre los requisitos, son estables y, por tanto, es confiable el análisis y diseño sobre ellos.

3 Aplicación de la métrica Especificidad de los requisitos:

La especificidad de los requisitos se calcula como:

Q**1** = n**ui** /n**r** = 26 /26= 1

Como resultado se obtuvo que la especificación de los requisitos no presenta ambigüedad, asegurando un alto nivel de calidad en el proceso de especificación.

4 Aplicación de la métrica Grado de validación:

El grado de validación de los requisitos se calcula:

Q**3** = n**c** / (n**c** + n**nv**) = 26 /(26+ 0) = 1

Esta métrica arrojó como resultado 1, por lo tanto la definición de los requisitos es correcta.

3.6.2 Aplicación de lista de verificación

Para contribuir al proceso de revisión de los requisitos se aplicó la lista de verificación propuesta por Pressman en su 6ta edición, a continuación se muestra dicha lista: (37)

1. ¿Los requisitos están establecidos de manera clara? ¿Estos pueden malinterpretarse?
2. ¿La fuente del requisito (por ejemplo, una persona, una regulación o un reglamento) está identificada? ¿El enunciado final del requisito ha sido examinado por la fuente original o comparándola con ella?
3. ¿El requisito está restringido en términos cuantitativos?
4. ¿Cuáles otros requisitos están relacionados con este? ¿Están registrados de manera clara por medio de una matriz de referencias cruzadas u otro mecanismo?
5. ¿El requisito viola alguna restricción del dominio del sistema?
6. ¿El requisito se puede probar? Si es así, ¿se pueden especificar las pruebas (algunas veces llamadas criterios de validación) para ejercitar el requisito?
7. ¿El requisito es rastreable por cualquier modelo de sistema que haya sido creado?

8. ¿El requisito es rastreable para los objetivos generales del sistema o producto?
9. ¿La especificación está estructurada de una forma que conduzca a su comprensión, referencia y traducción fácil en productos de trabajo más técnicos?
10. ¿Se ha creado algún índice para la especificación?
11. ¿Los requisitos asociados con el rendimiento, el desempeño y las características operacionales se han establecido de manera clara? ¿Cuáles requisitos parecen ser implícitos?

La realización de estas preguntas sirvió de utilidad para esta etapa del proceso de desarrollo del software, pues se pudo constatar que los requisitos obtenidos se encontraban en correspondencia con las necesidades de clientes y usuarios.

3.7 Validación del Diseño

Para la validación del diseño realizado fueron aplicadas un conjunto de métricas, arrojando los siguientes resultados:

2.8.1 Métrica Tamaño Operacional de Clases (TOC)

Está dirigida a contar la cantidad de atributos y operaciones para una clase individual, además de los valores promedios para el sistema orientado a objetos en su totalidad. TOC puede ser determinada haciendo uso de las siguientes medidas:

- El número de atributos (tanto atributos heredados como propios) encapsulados dentro de la clase.
- El número total de operaciones (las operaciones privadas y heredadas de la clase) que están encapsuladas en ella.

En la siguiente tabla se muestra la definición de los umbrales teniendo en cuenta los indicadores de responsabilidad, complejidad de implementación y reutilización.

Tabla 7: Indicadores de calidad y umbrales

	Categoría	Criterio
Responsabilidad	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Complejidad implementación	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Reutilización	Baja	> 2*Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	<= Prom.

Si el resultado obtenido indica valores grandes, significa que dicha clase posee un alto grado de responsabilidad, por lo que se reducirá la reutilización, se realizará más difícil la implementación y la realización de pruebas a la clase. Para determinar los siguientes valores fueron definidos los umbrales que se muestran en la tabla 9.

Tabla 8: Definición de clases según los umbrales

Categoría	Criterio
Bajo	prom<=15
Medio	15 <prom<=20
Alto	Más de 20

Para la responsabilidad asociada a las clases se obtuvo que el 50% de ellas tienen una baja responsabilidad, el 30% una media responsabilidad y las restantes una responsabilidad alta. Los resultados obtenidos se reflejan en la figura 21.

Figura 21: Resultados del indicador responsabilidad

Con respecto a la reutilización, como se puede observar en la siguiente figura, la métrica aplicada arrojó los siguientes resultados: de las 10 clases se obtuvo que el 23% poseen una alta reutilización, solo un 15% un valor bajo y el 62% con valor medio para el indicador.

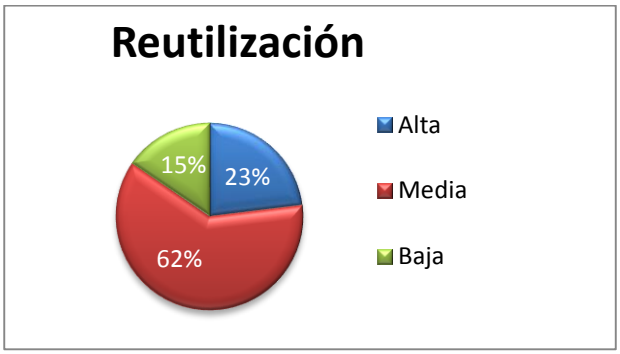


Figura 22: Resultados del indicador reutilización

En la aplicación del indicador asociado a la complejidad de implementación fueron obtenidas el 50% de las clases con baja, el 30% poseen una media complejidad y el resto una complejidad alta. Dichos valores se observan en la siguiente figura 23.

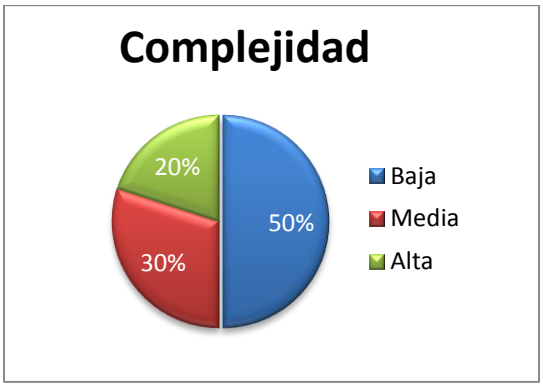


Figura 23: Resultados del indicador complejidad

Los datos representados anteriormente en los gráficos arrojan resultados positivos para la validación del diseño de la solución, ya que al presentar nivel bajo de complejidad y responsabilidad relacionados con altos índices de reutilización se facilitará la implementación de la solución que se propone.

2.8.2 Métrica Árbol de Profundidad de Herencia (APH)

Brinda una medida de cuantas clases padres afectan potencialmente las clases hijas. Esta se mide como el tamaño máximo desde un nodo hasta la raíz del árbol. A medida que crece el APH, es más probable que las clases de niveles inferiores hereden muchos métodos, esto da lugar a posibles

dificultades cuando se intenta predecir el comportamiento de una clase, además una jerarquía de clases profunda (con un valor grande de APH) lleva también a una mayor complejidad de diseño y a la posibilidad de reutilizar mayor cantidad de métodos.
Fueron definidos los siguientes criterios para evaluar dicha métrica:

Tabla 9. Criterios de métrica

Profundidad	Criterio
Mucha	APH>1
Poca	APH<=1

La aplicación de la métrica APH a 10 clases que conforman la propuesta de solución, dado que todas las clases gestoras y controladoras deben heredar de *BaseGtr* y *BaseController* respectivamente, arrojó como resultado que todas las clases poseen poca profundidad de herencia.

3.8 Verificación del sistema

Las pruebas consisten en la dinámica de la verificación del comportamiento de un programa en un conjunto finito de casos de prueba. Según Pressman verificación es el conjunto de actividades que aseguran que el software implemente correctamente una función determinada, y la validación es un conjunto diferente de actividades que aseguran que el software construido corresponde y satisface los requisitos del cliente. A continuación se muestra el nivel de prueba que se tuvo en cuenta:

- Nivel de unidad

3.8.1 Nivel de Unidad

Las pruebas unitarias son aplicadas para verificar que el software cumple los requerimientos funcionales y para asegurar la calidad del código entregado. Además, son la mejor forma de detectar fallas tempranamente en el desarrollo para de esta forma corregirlos cuanto antes. A este nivel se realizan las pruebas de funcionalidad, utilizando los métodos de prueba de Caja Blanca y Caja Negra para comprobar que tanto el código como la interfaz no contengan errores y se ejecutan correctamente. (38)

3.8.1.1 Método de prueba de caja negra

Las pruebas de caja negra están enfocadas en verificar la correcta relación entre las entradas y las salidas de una interfaz de usuario. Además, se limitan a que el *tester* (probador) pruebe con “datos” de entrada y estudie como salen, sin preocuparse de lo que ocurre en el interior. (39)

Para la puesta en práctica de las pruebas de caja negra fue empleada la técnica de la partición de equivalencia, esta divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. El diseño de estos casos de prueba para la partición equivalente se basa en la evaluación de las clases de equivalencia, las cuales representan un conjunto de estados válidos o inválidos para determinadas condiciones de entrada. En la figura 24 se muestran las no conformidades reveladas en cada iteración de las pruebas.

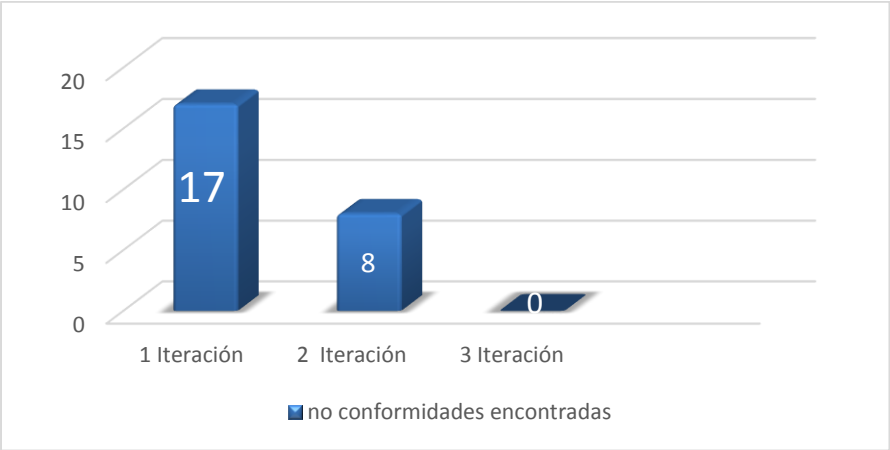


Figura 24: Resultados de las Iteraciones realizadas durante las pruebas de caja negra

3.8.1.2 Método de prueba de caja blanca

La prueba de caja blanca, denominada en ocasiones prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. (39). Este tipo de pruebas cuentan con varias técnicas, dentro de las cuales, para validar la calidad del producto se empleó la de Camino Básico.

El camino básico es una técnica de caja blanca que permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico (diseño de casos de prueba) de caminos de ejecución.

La figura que a continuación se muestra representa el grafo del camino básico perteneciente al método *crearDocumentosGenerado($idExpediente, $documentosAdjuntos, $idTipoEscrito, $tramite, $idTipoEstado)* de la clase *RegistrarEscritoRevisionGtr*.

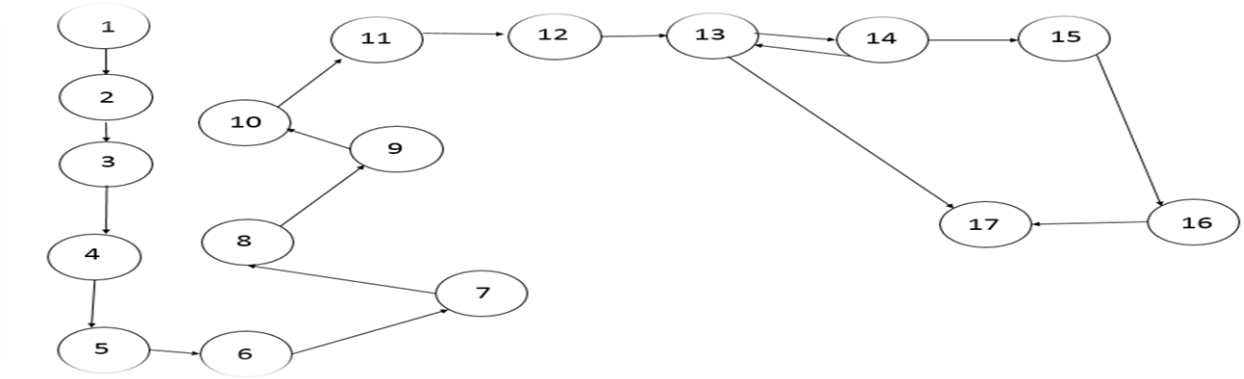


Figura 25: Grafo de camino básico del método *crearDocumentosGenerado*

Posteriormente al análisis de la ruta básica presentada se calcula la complejidad ciclomática de dicho método haciendo uso de la siguiente ecuación:

V(G)=A-N+2 siendo A la cantidad de aristas del grafo y N la cantidad de nodos.

La complejidad del método en cuestión quedaría de la siguiente manera:

V(G)=A-N+2

V(G)=18-17+2=3

El resultado obtenido luego de calculada la complejidad ciclomática indica que existen tres caminos posibles que se describen a continuación:

a) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17.
b) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 17.
c) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 13, 17.

Una vez obtenidos los caminos básicos se realizan por cada uno de ellos los casos de prueba. La siguiente tabla muestra el caso de prueba para la ruta del último inciso definido anteriormente.

Tabla 10: Diseño de caso de prueba del método *crearDocumentosGenerado*

Entrada	Resultados esperados	Condiciones
Es preciso para que el método sea ejecutado satisfactoriamente, la entrada del identificador del expediente en proceso, un arreglo de documentos adjuntos, el identificador del tipo de escrito, el trámite que se está ejecutando y el identificador del tipo de estado.	Se crea un documento generado según los parámetros de entrada.	$tipoEstado = $repo->find($idTipoEstado); Debe encontrarse el identificador del tipo de estado (pasado por parámetro) en registrado en la clase 'ComunBundle:AG\TipoEstado'

3.9 Validación de las variables de la investigación

El método Delphi es realizado por medio de la investigación a expertos con la ayuda de cuestionarios, a fin de poner de manifiesto convergencias de opiniones y deducir eventuales discrepancias entre el grupo de encuestados y los que aplican la misma. Para la aplicación del método es necesario tener en cuenta dos aspectos importantes:

- Selección del panel de expertos.
- Confección del cuestionario.

3.9.1 Proceso de selección de expertos

La selección de los especialistas se realiza a un grupo de personas con conocimientos y capaces de ofrecer un criterio, aportar ideas o valoraciones concluyentes respecto al tema en cuestión. Teniendo en cuenta los conocimientos fueron definidos, por el equipo de proyecto, un total de 5 especialistas:

Ing. Yeleny Almora Galvez (Jefa del proyecto)

Msc. Isabel González Flores (Analista del sistema)

Ing. Yulia Fustiel Alvares (Analista del sistema)

Ing. Yamilka Ríos de la Oz (Analista del sistema)

Ing. Lisandra Santamaria Pérez (Analista del sistema)

3.9.2 Confección del cuestionario

El cuestionario es la forma a través de la cual se interactúa con los especialistas, donde exponen sus criterios y valoraciones. El cuestionario se centra en el cumplimiento de los indicadores de disponibilidad y control de la información, de manera que se logre la validez de la propuesta de solución, son definidas un conjunto de preguntas para facilitar la interacción con los especialistas y se realiza un análisis de los resultados. El cuestionario se encuentra en el anexo 12.

3.9.3 Aplicación del método Delphi

Para la validación de las variables identificadas durante el procedimiento Revisión de la materia Laboral, los expertos seleccionados realizaron un análisis de las mismas, emitiendo respuestas a las preguntas realizadas a través del cuestionario confeccionado. A continuación se muestran los resultados obtenidos:

El 100% de los expertos encuestados coincidieron en el cumplimiento de los indicadores definidos para las variables disponibilidad y control de la información.

3.9.4 Opiniones de los expertos

A continuación se muestran algunas opiniones de los expertos referentes a los beneficios que proporcionará el sistema para la realización del procedimiento Revisión, las restantes opiniones se pueden consultar en el anexo 13:

- Los expedientes están almacenados de forma digital en la herramienta. La información guardada persiste en la base de datos, a la cual se le realizan copias de seguridad, dificultándose la pérdida de información.
- Los expedientes se encuentran disponibles en el sistema, tanto durante el proceso como una vez terminado el mismo.
- El acceso a los expedientes, según el permiso que se tenga en el sistema, puede ser realizado por los usuarios en cualquier momento.
- En el sistema igualmente existe un único expediente, pero debido a que está digital puede ser consultado simultáneamente por el personal autorizado.

3.10 Conclusiones del Capítulo

El presente capítulo permitió demostrar lo siguiente:

La aplicación de las métricas arrojó resultados satisfactorios, demostrando que los requisitos poseen un correcto nivel de especificidad, además existe bajo grado de responsabilidad y complejidad, mientras que la reutilización de código de las clases representa un valor medio. Se demostró que no existe código innecesario en la aplicación a través de los resultados obtenidos en las pruebas de caja blanca, además, la aplicación de las pruebas de caja negra permitió corregir la totalidad de las no conformidades del sistema, validando las funcionalidades del mismo. El diagrama de despliegue permite un mejor entendimiento de cómo quedará el sistema una vez desplegado en el TSP.

Conclusiones Generales

Culminado el desarrollo del presente trabajo se concluye que:

- La definición del marco teórico de la investigación facilitó determinar y detallar las tecnologías, herramientas y lenguajes adecuados para el desarrollo del sistema.
- El estudio de los conceptos relacionados con el procedimiento Revisión de la materia Laboral permitió una mejor visión del contexto donde se desarrolla el mismo.
- El diseño realizado mediante el diagrama de casos de uso, diagrama de clases del diseño y el modelo de datos permitió materializar los requisitos, proporcionando una guía para la agilización de la implementación de las funcionalidades.
- La implementación posibilitó obtener un software funcional capaz de satisfacer las necesidades del cliente y resolver los problemas existentes en el TSP por los cuales se decidió iniciar el desarrollo de la aplicación.
- La validación del diseño e implementación haciendo uso de las métricas de calidad y pruebas de software permitieron demostrar la eficacia de los artefactos obtenidos durante el desarrollo del sistema.

Recomendaciones

Al concluir la investigación se recomienda:

- Concluir la implementación del procedimiento Revisión de la materia Laboral del SITPC de manera que se logre una realización integra del mismo.
- Realizar la integración del módulo Revisión con el resto del sistema SITPC.

Referencias Bibliográficas

Bibliografía Referenciada

1. **Calderío, Blas Roca.** *Ley de Procedimiento Civil, Administrativo y Penal.* 1977.

2. **Manuel González Martínez.** Software para Despachos de Abogados y Profesionales para Gestión de Expedientes Jurídicos y Contactos. [En línea] 2014. [Citado el: 2015 de Febrero de 7.] http://www.brindys.com/brindys/cas.htm.

3. **Tantum_Sistema de Gestión Jurídica.** Gestion Jurídica Simple y Ágil. *Gestion Jurídica Simple y Ágil.* [En línea] 2014. [Citado el: 10 de Marzo de 2015.] http://www.tantum.com.ar/.

4. **Sistemas Jurídicos SRL.** Lex-Doctor. Tu estudio Jurídico en línea. *Lex-Doctor. Tu estudio Jurídico en línea.* [En línea] [Citado el: 7 de Febrero de 2015.] http://www.lex-doctor.com.

5. **Daniel, MSc. y González Guardarramas, MSc. José Ramón.** Sistema para la Tramitación de Procesos Penales. Santa Clara : Universidad Central Marta Abreu, 2008.

6. **Ivar Jacobson, Grady Booch, James Rumbaugh.** *El Proceso Unificado de Desarrollo de Software.* A. Madrid : OTERO, 2000.

7. **Alegsa.com.ar.** Alegsa.com.ar. *Alegsa.com.ar.* [En línea] 1998-2015. http://www.alegsa.com.ar/Dic/lenguajes%20de%20programacion.php.

8. **Aronson., Larry.** *HTML manual of style.* EEUU, Weasley : s.n., 2011.

9. **McFarlan, David Sawyer.** *The Missing Manual.* s.l. : First Edition, 2008.

10. **Desarrolloweb.com.** *Manual de CSS 3.* 2011.

11. **The PHP Group.** PHP. *PHP.* [En línea] 2001-2015. [Citado el: 8 de Febrero de 2015.] http://php.net/.

12. **Schmuller, Joseph.** *Aprendiendo UML en 24h.* Mexico : Pearson educacion, 2000.

13. **Pérez, Juan Diego.** *Notaciones y lenguajes de procesos.* Sevilla : s.n.

14. *Marco de trabajo para el desarrollo de herramientas orientadas a la gestión e integración de servicios telemáticos de infraestructura en GNU/Linux.* **Ramón Alexander Anglada Martínez, Alain Abel Garófalo Hernández.** La Habana : s.n., 2013, Vol. 7.

15. **Fabien Potencier, Ryan Weaver.** *Symfony 2.4.*

16. **Librosweb.es.** Librosweb.es. *Librosweb.es.* [En línea] [Citado el: 13 de Abril de 2015.] http://librosweb.es/libro/symfony_2_x/capitulo_7/plantillas.html.

17. **Mark Otto, Jacob Thornton.** *Bootstrap 3, el manual oficial.* 2015.

18. **Desarrolloweb.com.** Manual de jQuery. *Manual de jQuery.* [En línea] [Citado el: 15 de Febrero de 2015.] http://www.desarrolloweb.com/manuales/manual-jquery.html.

Referencias Bibliográficas

19. **Carrero, Angel.** Conceptos básicos de ORM (Object Relational Mapping). 2013.

20. **Ecured Enciclopedia colaborativa Cubana.** Ecured Enciclopedia colaborativa Cubana. [En línea] [Citado el: 11 de marzo de 205.] http://www.ecured.cu/index.php/Heramientas_CASE.

21. **Oracle Corporation and/or its affiliates.** NetBeans IDE. *NetBeans IDE.* [En línea] 2015. [Citado el: 7 de Febrero de 2015.] https://netbeans.org/community/releases/roadmap.html.

22. **PostgreSQL, Global Development Group .** Sitio oficial de PostgreSQL. *Sitio oficial de PostgreSQL.* [En línea] 2014. http://postgresql.org/about.

23. **Asenjo, Jorge Sanchez.** *Servidores de Aplicaciones Web.* 2012.

24. **Yamilka Rios, Lisandra Santamaria Pérez.** *Descripción de procesos de negocio.* La Habana. Cuba : s.n.

25. **Yoslenys Roque Hernandez, Maurys I Brito.** *Especificación de requisitos no funcionales de software.* La Habana : s.n.

26. **Microsoft_Developer Network.** Diagramas de casos de uso de UML. *Diagramas de casos de uso de UML.* [En línea] 2013. [Citado el: 10 de Mayo de 2015.] https://msdn.microsoft.com/es-es/library/dd409432.aspx.

27. **SG Buzz.** Conocimiento àra crear Software grandioso. *Conocimiento àra crear Software grandioso.* [En línea] [Citado el: 18 de Mayo de 2015.] http://sg.com.mx/content/view/510.

28. **Abiztar Learning Technologies.** Abiztar Learning Technologies. *Abiztar Learning Technologies.* [En línea] 2014. [Citado el: 15 de Mayo de 2015.] http://www.milestone.com.mx/articulos/casos_a_incluir_casos_a_extender.htm.

29. **Larman, Craig.** *UML y Patrones-Face del diseño I.*

30. **Universidad Nacional Abierta y a distancia.** Universidad Nacional Abierta y a distancia. *Universidad Nacional Abierta y a distancia.* [En línea] [Citado el: 4 de Mayo de 2015.] http://datateca.unad.edu.co/contenidos/200609/exeuml/leccin_39_diagrama_de_clases_de_diseo.html.

31. *Diagramas de secuencia.* **Fabián Garcia, Paola Parra, Andres Parrado.** 2006.

32. **Sparx Systems.** Sparx Systems. *Sparx Systems.* [En línea] 2017. [Citado el: 2 de Junio de 2015.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_packagediagram.html.

33. **Carlos E. Barrios, José Leon, Yenni Romero.** *Metodologia de Pressman 2da Parte.* 2015.

34. **Oficina Nacional de Normalización.** *Ingeniería de software-Calidad del Producto.* Ciudad de La Habana. : s.n., 2001.

35. **Spark Systems.** *Spark Systems.* 2007.

36. **Sommerville, Ian.** *Ingenieria de software 7ma edicion.* Madrd : s.n., 2005.

Referencias Bibliográficas

37. **Pressman, Roger S.** *Software Engineering.* 2001.

38. **Pruebas de Software .** Gestión de Calidad y Pruebas de Software. *Gestión de Calidad y Pruebas de Software.* [En línea] 2015. [Citado el: 11 de Abril de 2015.] http://pruebasdesoftware.com/index.htm.

39. **Tenorio, Roberto Ruiz.** *Las pruebas de software y su importancia en las organizaciones.* agosto, 2010.

40. **Rios, Yamilka.** *Reglas de Negocio.* La habana. Cuba : s.n.

41. **Pressman, Roger S.** *Ingeniería del Software: Un enfoque práctico.* México DF : McGraw Hill, 2005.

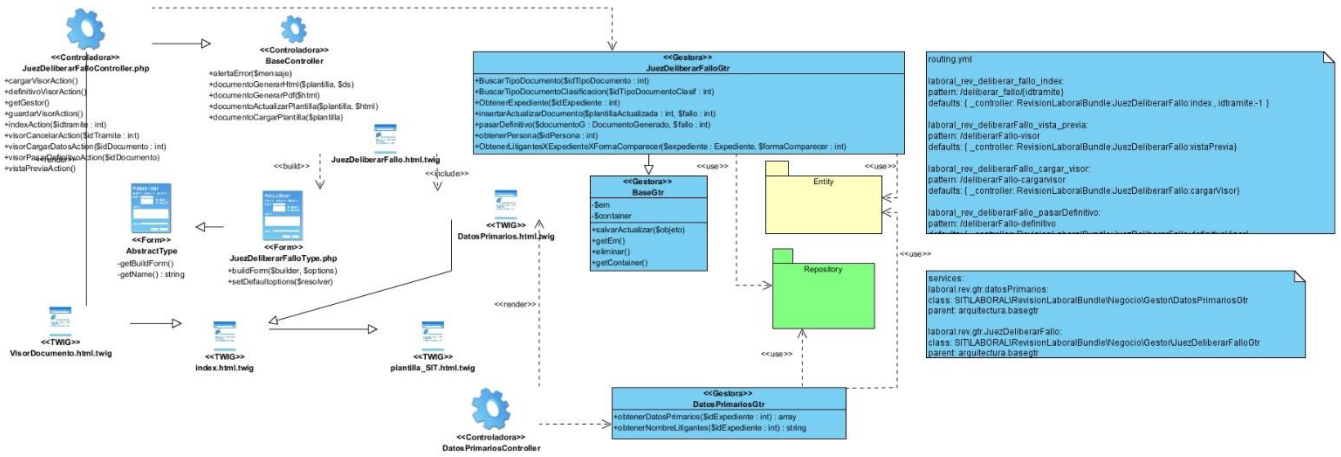
42. **SitePoint Pty.** PHP Libraries. *PHP Libraries.* [En línea] 20 de Mayo de 2015. http://www.sitepoint.com/extending-twig-templates-inheritance-filters-and-functions/.

43. **Departamento de lenguajes y sistemas informáticos.** Patrones de asignación de responsabilidades. *Patrones de asignación de responsabilidades.* Sevilla : s.n.

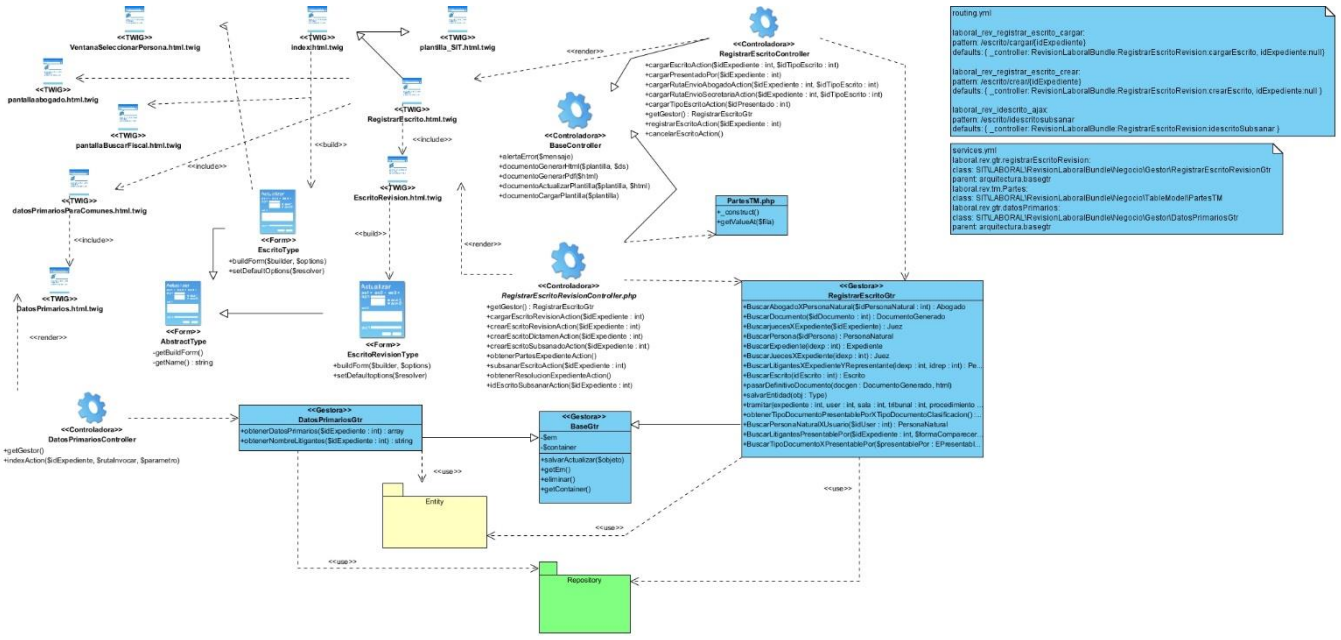
44. **Pressman.** *Ingeniería de Software. Un enfoque práctico. Pág. 384.*

Anexos

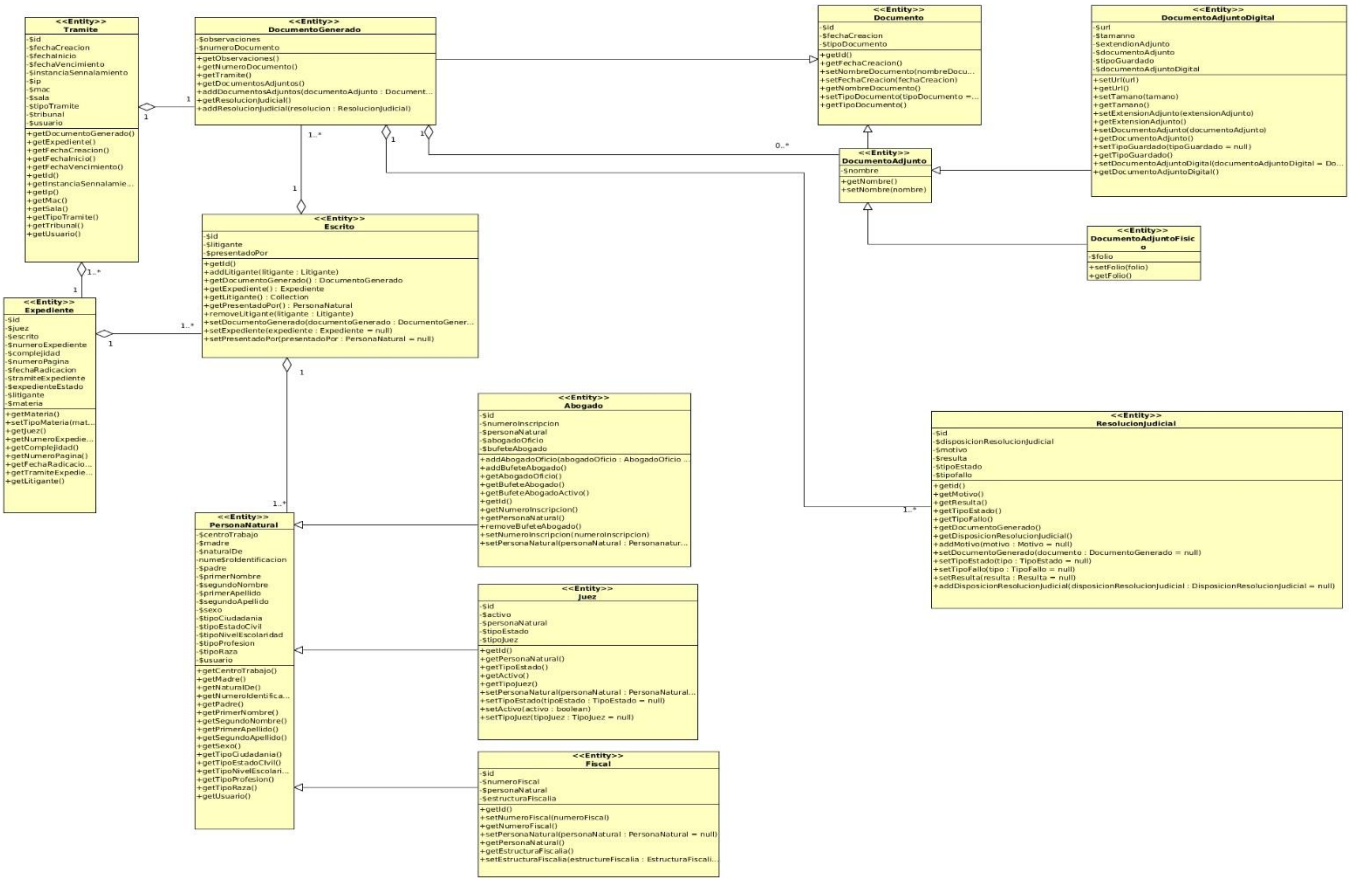
Anexo 1: Diagrama de clases del CU DeliberarFallo

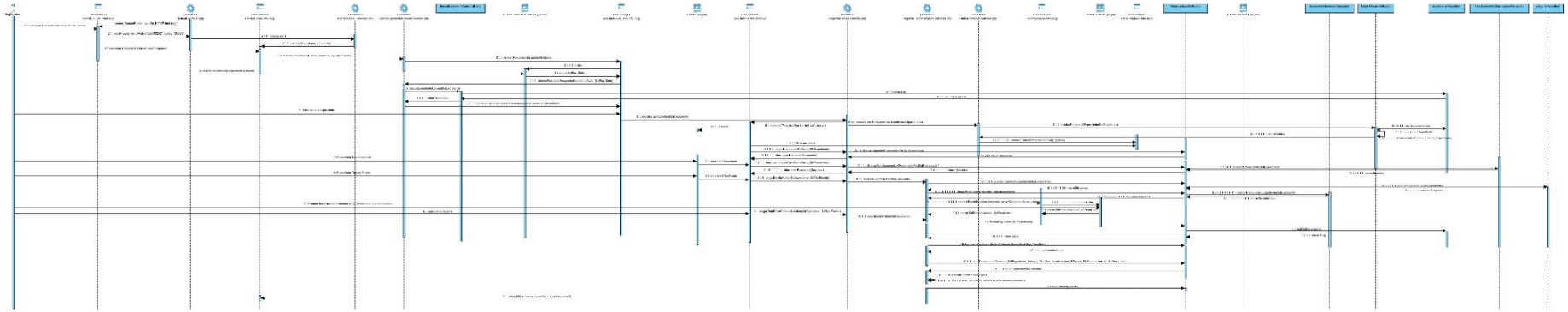


Anexo 2: Diagrama de clases del caso de uso RegistrarEscrito



Anexo 3: Diagrama de clases del paquete Emtity





Anexo 4: Diagrama de secuencia del CU RegistrarEscrito