Universidad de las Ciencias Informáticas

Facultad 3

CEGEL

Desarrollo del procedimiento Revisión del subsistema Económico del proyecto SITPC

Trabajo final presentado en opción al título de Ingeniero en Ciencias Informáticas

Autor: **Yadieski Sandy Escalona Marín**

Tutor(es): **Ing. Eliober Cleger Despaigne**

Ing. Hernán Antonio Sánchez Guzmán

Ciudad de la Habana, Junio del 2015

DECLARACIÓN DE AUTORÍA

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

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

Ing. Eliober Cleger Despaigne

_____

Firma del Tutor

Ing. Hernán Antonio Sánchez Guzmán

_____

Firma del Tutor

Yadieski Sandy Escalona Marín

_____

Firma del Autor

AGRADECIMIENTOS

A mis padres por estar siempre a mi lado y permitirme ser la persona que soy, por ser las personas que más amo.

A mis profesores y amigos por su apoyo y ayuda en los momentos que pensé que no podía más.

A mis tutores por su sacrificio y dedicación, por las horas de sueños que se perdieron por mí.

DEDICATORIA

A mis abuelos y a todas las personas que confiaron en que si podía cumplir con este sueño.

RESUMEN

Los Tribunales Populares Cubanos están estructurados en tres instancias: Municipal, Provincial y Supremo, en las cuales se tramitan cotidianamente una gran cantidad de datos en dependencia de la materia en que se desarrollen. Estas materias son: Civil, Laboral, Penal, Administrativo y Económico. Actualmente los procesos que se llevan a cabo en la materia Económica se realizan manualmente lo que provoca demoras en la tramitación de los procesos y por tanto el incumplimiento de los términos procesales. Conjuntamente se pueden introducir errores en la creación de las resoluciones y demás documentos judiciales. Dadas las condiciones que anteceden, se evidencia la necesidad de crear un sistema informático que responda a las necesidades de los Tribunales Populares Cubanos. El presente trabajo tiene como objetivo la especificación de un conjunto de artefactos que resultan necesarios para una satisfactoria implementación del subsistema Económico en la instancia Suprema, específicamente para el procedimiento de Revisión. Para ello, se realiza una descripción de la metodología, las herramientas y otras tecnologías que se utilizan en el desarrollo del subsistema. De igual manera se ofrecen los artefactos obtenidos durante su diseño e implementación, los cuales son verificados mediante un conjunto de pruebas y métricas que permiten comprobar la calidad del software. A partir del desarrollo del procedimiento Revisión del subsistema Económico, se espera contribuir a mantener la integridad de la información y disminuir el tiempo de ejecución del proceso.

Palabras claves: informática jurídica de gestión, materia económica, procedimiento de revisión

TABLA DE CONTENIDO

TABLA DE CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

TABLA DE CONTENIDO

INTRODUCCIÓN

Las TIC's (Tecnologías de la Información y las Comunicaciones) constituyen el puente que permite el establecimiento de relaciones jurídicas, ignorando las distancias y desafiando los parámetros tradicionales. El hombre con los sistemas de cómputo ha logrado librarse de tareas manuales, usándolo como recurso de efectividad. El gobierno, las instituciones, empresas, los profesionales y un sin número de personas están motivados a buscar soluciones que informaticen sus tareas, para realizarlas de la forma más sencilla, en el menor tiempo y con el menor costo posible.

El sistema de tribunales en Cuba posee una composición que está dada por tres instancias: el Tribunal Supremo Popular (TSP), los Tribunales Provinciales Populares y los Tribunales Municipales Populares. Su objetivo principal es impartir justicia para contribuir a la seguridad jurídica, a la preservación y al desarrollo de la sociedad socialista. En las diferentes instancias se resuelven asuntos de las materias: Civil, Administrativo, Laboral, Económico y Penal, siguiendo las leyes procesales vigentes. Todas las disposiciones se registran en autos, providencias o sentencias, resoluciones que deben ser notificadas a las partes, lo que genera un sinnúmero de documentación así como desgaste humano. Los reportes estadísticos se realizan manualmente y están sujetos a períodos fijos, dificultando su obtención dinámica. (TSP, 2015)

El centro CEGEL perteneciente a la Universidad de las Ciencias Informáticas (UCI) se encuentra enfrascado en el desarrollo de un producto de software a solicitud de los Tribunales Populares Cubanos (TPC). Este convenio tiene como colofón darle solución a las deficiencias que presenta este organismo en la gestión de la información. En este contexto se crea el proyecto Sistema de Informatización para la Gestión de los Tribunales Populares Cubanos (SITPC), compuesto por los siguientes subsistemas: Administrativo, Laboral, Civil, Económico, Penal y Administración y Gobierno.

Como parte del subsistema Económico se encuentra el procedimiento Revisión. Este procede contra las sentencias dictadas en recursos de casación por la Sala de lo Económico del Tribunal Supremo Popular. Para su tramitación se captan desde la fase inicial grandes cantidades de datos, los que deben ser repetidos en cada documento que lo requiera, se realiza de forma manual y provoca que se cometan errores como: la duplicación, la alteración de datos o la perdida de información debido al gran tamaño de los expedientes compuestos por más de 50 fojas. Son varias las personas implicadas durante el proceso, clasificadas en naturales, fiscales y jurídicas. Todas las notificaciones se realizan vía correo postal, frenando el cumplimiento de los términos de ley, relacionados al tiempo, por parte de los tribunales. Producto de esta tramitación engorrosa de captar la información, almacenarla, custodiarla y notificar a las personas implicadas, el proceso llega a ser tan largo que cuando la sentencia llega a las

partes ya no interesa. Se aprecian además rasgos de desobediencia dentro del personal que participaba en el proceso en cuanto al manejo de la integridad de la información.
A partir de la problemática descrita anteriormente se identificó como **problema a resolver**:

¿Cómo ejecutar el procedimiento Revisión de la materia de lo Económico de manera que contribuya a la integridad de la información y disminuya su tiempo de realización?

Se identificó como **objeto de estudio**: El desarrollo de software en la informática jurídica de gestión, planteándose como **objetivo general**: Desarrollar una herramienta informática para la ejecución del procedimiento Revisión del subsistema Económico de manera que contribuya a la integridad de la información y disminuya el tiempo de realización del proceso. Enmarcado en el **Campo de acción**: Desarrollo del procedimiento Revisión del subsistema Económico del proyecto SITPC.

Objetivos específicos:

- Elaborar el marco teórico de la investigación para obtención de la teoría asociada al campo de acción y las principales tecnologías que dan soporte al desarrollo web.
- Realizar el modelo de procesos de negocios de manera que permita obtener los requisitos de software y realizar el modelo del sistema.
- Obtener el modelo de diseño para transformar los requisitos identificados en una representación técnica del software.
- Realizar la implementación del diseño propuesto para la obtención de un producto funcional.
- Evaluar la solución propuesta para comprobar la calidad del trabajo realizado, corregir errores y asegurar que el producto obtenido satisface los requisitos establecidos.
- Aplicar instrumentos que permitan evaluar el comportamiento de las variables de la investigación antes y después de la solución propuesta.

Formulándose como **idea a defender**: El desarrollo del procedimiento Revisión del subsistema Económico, contribuirá a la integridad de la información y disminuirá el tiempo de ejecución del proceso.

Tareas a desarrollar para dar cumplimiento a los objetivos específicos:

1. Caracterización el sistema de los TPC y el procedimiento Revisión de la materia de lo Económico.
2. Caracterización de soluciones informáticas que gestionen los procedimientos de la materia de lo económico dentro de los procesos judiciales.
3. Fundamentación de la metodología de desarrollo de software, herramienta CASE, lenguaje de modelado y demás tecnologías a utilizar.
4. Modelación del proceso de negocio.
5. Identificación de las reglas del negocio.

6. Identificación y especificación de los Requisitos de software.
7. Descripción de los casos de uso.
8. Confección de los prototipos de interfaz de usuario.
9. Se verifica la calidad de los requisitos obtenidos mediante la aplicación de métricas y los prototipos de interfaz de usuario.
10. Realización del diseño.
11. Aplicación de métricas para la verificar la calidad del diseño.
12. Implementación de los Casos de Usos.
13. Diseño de casos de prueba.
14. Ejecución de pruebas a las funcionalidades implementadas.
15. Solución de las No conformidades detectadas.
16. Verificación de la solución propuesta.

Para el desarrollo de las tareas de investigación serán empleados los métodos de Investigación y Desarrollo (I + D) de la Informática: Teóricos y Empíricos (Métodos “I + D” de la Informática, 2005), que constituyen los procedimientos que se utilizan para estudiar la realidad, la naturaleza, la sociedad y el pensamiento, con el propósito de descubrir su esencia, sus relaciones y llegar al conocimiento científico; por lo que son el instrumento para alcanzar los fines de la investigación.

Métodos de Investigación Científica

Métodos teóricos

- Histórico-lógico: Mediante este método se analiza la trayectoria real para el procedimiento Revisión su evolución y desarrollo, además se utiliza para investigar sobre sistemas de gestión que tengan similitud con el que se va a implementar además de permitir la utilización del pensamiento en sus distintas funciones para alcanzar el conocimiento.
- Analítico-sintético: Facilita el entendimiento del fenómeno en que se trabaja, siendo más útil la división del mismo en diferentes fases y de esta forma descubrir sus características generales, lo que ayuda a seguir una correcta investigación. A partir de un estudio detallado de la información obtenida se hace necesario organizarla y sintetizarla para lograr una estructura adecuada.
- Modelación: El modelo científico es un instrumento de la investigación de carácter material o teórico, creado para reproducir el fenómeno que se está estudiando. Se modelan todos los procesos relacionados con la Revisión Económica para un entendimiento del proceso más detallado que precise el cumplimiento del objetivo.

Métodos empíricos

- Entrevista: Es una conversación planificada entre el investigador y el entrevistado para obtener información. Se realizan entrevistas a los jueces para obtener información y entender mejor los procesos relacionados con el procedimiento de Revisión Económica.
- Observación: Se usó en el diagnóstico del problema a investigar y es de gran utilidad en el diseño de la investigación. Se realizan observaciones sobre la forma en que se desarrolla la tramitación del procedimiento en el Tribunal Supremo Popular, luego son corroboradas las prácticas observadas según la forma en que deban realizarse.
- Encuesta: Se utilizó con el fin de verificar las variables de la investigación. Se efectúan encuestas a los analistas y programadores con vasta experiencia en el proyecto para evaluar el grado de veracidad en el cumplimiento del objetivo de la investigación realizada.

El presente documento se estructura en tres capítulos descritos a continuación:

Capítulo 1: En este capítulo se estudian las principales definiciones que se deben tener en cuenta para desarrollar una solución que informatice los procedimientos de Revisión del subsistema Económico. Se explica el flujo del proceso a implementar. Se realiza un análisis de las soluciones informáticas existentes en el mundo relacionado al campo de acción. Por último, se analizan las plataformas, metodologías, herramientas para el desarrollo del software y los patrones de diseño orientado a objetos.

Capítulo 2: En este capítulo se realiza un análisis del modelado del negocio, la arquitectura del sistema a implementar y la obtención del modelo de diseño e implementación. Se diseñan los diagramas de clases y de secuencia aplicados al caso de uso seleccionado por su complejidad e importancia para el entendimiento del desarrollo de la solución. Así como el uso de los patrones de diseño evidenciados en la solución y demás artefactos que se generan durante el desarrollo de la solución.

Capítulo 3: En este capítulo se verifica la calidad de los artefactos ingenieriles obtenidos en el desarrollo de la solución propuesta. Se verifica mediante el uso de métricas la calidad de la especificación y el diseño propuesto. Se realizan pruebas de software a la solución desarrollada y se verifica la calidad de las funcionalidades implementadas, atendiendo al correcto comportamiento del sistema ante las disímiles situaciones. Por último, se analizan los resultados obtenidos de la encuesta aplicada a los especialistas del proyecto con el fin de evaluar el comportamiento de las variables de la investigación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se hace un breve resumen sobre los principales artículos que componen el proceso de Revisión Económica, los cuales deben ser tomados en cuenta para el desarrollo de la solución informática que se desea lograr. Se realiza un estado del arte de los sistemas informáticos para la gestión judicial similares al que se desarrollará, además de fundamentar las tecnologías y metodologías que se utilizan durante el desarrollo del sistema.

1.2 Definiciones relacionadas con el objeto de estudio y el campo de acción

Con el objetivo de lograr un entendimiento del contenido a tratar en la investigación, se requiere el conocimiento de los conceptos enunciados a continuación:

Cuando se dice **partes** se refiere a la parte demandante y la parte demandada, que pueden ser personas naturales, personas jurídicas, entre otras. Y cada parte, por otro lado, puede estar constituida por una o más personas.

El **proceso judicial** es un procedimiento en el que se tramitan determinados conflictos que deben ser sometidos a la decisión de jueces en un tribunal, ya sean familiares, patrimoniales, penales, laborales o administrativos.

La **informática jurídica de gestión** está encaminada a organizar y controlar la información jurídica de documentos, expedientes, libros, etc., mediante la aplicación de programas de administración que permitan crear identificadores y descriptores para la clasificación de dicha información. Este tipo de informática es conocida como de administración y/o control, es utilizada en tribunales, estudios jurídicos, notarías, entre otras; se utiliza sobre todo para llevar el seguimiento de trámites y procesos con el objeto de mantener actualizada la información y llevar un buen control de la misma. (Valdes, 2013)

1.3 El procedimiento Revisión del subsistema Económico

El procedimiento Revisión tiene comienzo cuando las partes presentan una demanda y la documentación que la acompaña. La secretaria del TSP registra la nueva demanda de Revisión y delega la misma a un juez según la decisión que tome el jefe de sala. El juez ponente se hará cargo de tomar la decisión si el proceso continúa o no. Una vez que culmine el análisis, se dispone sobre la demanda, la que puede ser rechazada, admitida o declarada para subsanación (en un término de 2 días). En el caso que la demanda sea admitida, la secretaria dará traslado a las partes (darle a conocer al demandante y al demandado que existe un proceso en contra de él). Se dispondrá de un período de 15 días para que el demandado responda

a la demanda. Cuando el término de tiempo concluye, se declara el proceso concluso para sentencia y se realiza el acta de discusión y votación. El procedimiento culmina con el dictamen de la sentencia y la notificación a las partes.

1.4 Soluciones informáticas para la gestión judicial existentes en el mundo

El proyecto del cual es parte esta investigación centra su desarrollo en la rama de la Informática Jurídica de Gestión por las aspiraciones y características del mismo. Su principal objetivo es informatizar los procedimientos judiciales en cada instancia de los tribunales cubanos. La introducción de las TIC's en la administración de justicia permite que la toma de decisiones sea de mayor calidad y, al mismo tiempo, abierta, transparente y próxima al ciudadano. En la actualidad se promueve el uso de la informática en la rama del derecho y existen numerosas soluciones informáticas en el mundo que hacen uso de sistemas informáticos para la gestión procesal. La mayoría de estas aplicaciones están enfocadas en la agilización de la tramitación procesal. Sin embargo, hasta el momento no existe una solución informática que facilite íntegramente el desarrollo de un proceso judicial similar a los que tienen lugar en los TPC debido a las diferencias en las leyes de cada país.

Sistemas estudiados a nivel internacional

- **SARC**

Los SARC (Sistemas Alternativos de Resolución de Conflictos) se establecieron en las Cortes de los Distritos Federales y se utilizan en el arbitraje, la mediación, la evaluación de casos y en los juicios de jurado abreviado. En busca de nuevas y mejores formas de hacer las cosas se incluye el uso de las tecnologías y las mejores prácticas de tramitación para las personas usuarias, con las iniciativas de informatización de Despachos. Utilizada principalmente en **Estados Unidos de América** en la búsqueda de respuestas rápidas a los procesos judiciales aunque España también ha implementado un sitio para arbitraje on-line. (Grupo Especial, 2001)

- **SAGJ**

Sistema Automatizado de Gestión Judicial SAGJ, es un conjunto de elementos orientados al almacenamiento, procesamiento, administración de datos y consulta de información. Fue desarrollado esencialmente para la gestión electrónica y digital de los expedientes que cursan trámite ante los tribunales del Órgano Judicial de la **República de Panamá.**

La plataforma tecnológica de este sistema de información está constituida por un grupo de módulos o aplicaciones informáticas que pueden trabajar de forma independiente y en conjunto, para brindar servicios

integrales a los usuarios de la Administración de Justicia, tales como, el módulo de Reparto del Registro Único de Entrada, la solución informática para la tramitación del Expediente Judicial Electrónico, el Tarjetero Electrónico y las Certificaciones de Depósito Judicial, entre otros. (Autoridad nacinal para la innovasion gubernamental, 2012)

- **SDJ(Costa Rica)**

Sistema de Depósitos y Pagos Judiciales(SDJ): Este sistema fue planificado, desarrollado y plasmado en una realidad por funcionarios del Poder Judicial que tenían todo el conocimiento de la problemática que se vivía día a día y que contaron con todo el apoyo de la Administración Superior. El rediseño de procedimientos se hizo pensando no solo en la reducción de labores para la institución, sino teniendo siempre en mente la solución a un grave problema social que provocaba la tardanza en el pago de las pensiones alimentarias. La individualización del control de los depósitos y pagos a nivel de expediente es uno de los principales aspectos que revolucionaron el proceso. Convertir en una oportunidad las limitaciones de recursos y en ese momento una inadecuada infraestructura de comunicaciones, gracias al establecimiento una alianza estratégica con una institución bancaria especializada en la recepción y pago de dineros, que ya contaba con una infraestructura de comunicación robusta. (Bastos, 2004)

Sistemas estudiados a nivel nacional

- **SisProp**

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

Deficiencias:

- ✓ No capta ningún dato de la fase judicial de la tramitación y decisión del tribunal.
- ✓ No supera la barrera del papel.
- ✓ No aporta estadísticas, ni información alguna.
- ✓ No valida casi ningún dato.

Programado en Delphi, corre sobre SQL Server por lo que no es compatible con el software libre en el que se están programado los sistemas generales de cada materia judicial. (Castro Morell, 2008)

- **Sistema de Procesos Económicos (SISECO)**

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

Deficiencias:

- ✓ Inserta documentos radicados manualmente. La secretaria inserta los datos en la aplicación y cada cinco años borra los datos de los años anteriores quedando solamente la información asentada en los libros.
- ✓ Las salvas se guardan en discos.
- ✓ Posee elevados tiempos de respuesta a los principales procesos, cualidad que no va aparejada con la estadística en tiempo real.

SISECO no informatiza la mayoría de los procedimientos, pues la radicación se realiza de forma manual, las salvas se guardan en discos, que en la actualidad ya no son de uso, la información almacenada es borrada cada cinco años, algo incomprensible para un sistema judicial, ya que la información es el mecanismo principal para la justicia. (Megías, 2010)

1.5 Análisis de las soluciones mostradas

A continuación se muestra la tabla de las diferentes soluciones existentes evaluadas en los requisitos que se consideran más importantes, para para el logro de la solución que se propone.

X – cumple con el criterio	En el Mundo			En Cuba	
Criterios / soluciones	SARC	SAGJ	SDJ	SISPRO	SISECO
Registrar una demanda de revisión.					
Radicar y tunar.					X
Disponer una demanda de revisión.					
Notificar a las partes del proceso.	X	X	X		
Software libre					X
Uso de licencias propietarias	X	X	X	X	

Tabla 1 Evaluación de criterios de informatización en las soluciones estudiadas

Estas soluciones informática estudiadas anteriormente no cumplen con las exigencias necesarias para la informatización de los procesos existentes en los Tribunales Populares Cubanos. Tienen la limitación de que muchos son privativos lo cual va en contra de las políticas de soberanía tecnológica del país. Estas soluciones informáticas están implementados para países que poseen un sistema judicial distinto a Cuba, lo cual es un punto muy importante a tener en cuenta siendo esta la principal limitante. Las soluciones cubanas se implementaron para el sistema judicial de nuestro país pero son obsoletas, sus características no cumplen lo que se busca, no poseen documentación y carecen de funcionalidades en su desarrollo respectivamente. A partir del análisis realizado surge la necesidad de implementar un sistema informático que cumpla con todos los requisitos necesarios para un excelente manejo del Sistema Judicial Cubano.

Como paso previo a la solución del problema fue necesario estudiar las principales características de la metodología, las tecnologías, herramientas y lenguajes con el objetivo de obtener los elementos teóricos necesarios para su aplicación en el desarrollo de la propuesta de solución.

1.6 Fundamentación de las herramientas, metodología y tecnologías a utilizar

Metodología de Desarrollo de Software

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software. Se van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué rol van a desempeñar. Además detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla. (Pressman, 2005)

Proceso Racional Unificado (RUP[1])

Según de define Roger S. Pressman RUP es un modelo de software (técnica para tratar con la complejidad aparejada a un sistema) que permite el desarrollo de software a gran escala, mediante un proceso continuo de pruebas y retroalimentación, garantiza el cumplimiento de ciertos estándares de calidad. Tiene tres características distintivas. Estas características son (Pressman, 2005):

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

RUP identifica a los flujos de trabajo fundamentales que se producen durante el proceso de desarrollo de software. Estos flujos incluyen el Modelado de Negocio, Requisitos, Análisis y Diseño, Implementación, Prueba y Despliegue. Además establece tres etapas para la realización de la documentación necesaria en el desarrollo de un sistema: Configuración y administración del cambio, Administración del proyecto y Ambiente (Guerra, 2010). Debido a la magnitud y complejidad del proyecto de Informatización para la

[1] Siglas del inglés Rational Unified Process

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

A pesar de que RUP define la documentación que es necesaria realizar en la construcción de un software el proyecto de Informatización para la Gestión de los Tribunales Populares Cubanos, centrado en el objetivo de la UCI de mejorar y asegurar la calidad de todos los productos desarrollados, se rige por las plantillas especificadas en el Expediente de Proyecto del Programa de Mejora. Estas plantillas son el resultado de la experiencia práctica y se guían por el modelo CMMI[2] (Capacidad de Madurez para la Integración del Modelado).

CMMI es un enfoque de mejora de procesos que provee a las organizaciones de los elementos esenciales para un proceso efectivo mejorando los procesos de desarrollo de productos y servicios, adquisiciones y mantenimiento (BLANCO, 2010). Por lo que en la Universidad de las Ciencias Informáticas se crea el siguiente ciclo de vida básico para los proyectos alcanzados por el Programa de Mejoras pero en el caso del proyecto SITPC del centro CEGEL de la Facultad 3, se pretende llegar en la investigación solo hasta la fase de pruebas internas.

Ciclo básico para los proyectos alcanzados por el Programa de mejoras (CMMI)

Estudio Preliminar: En esta fase se llevan a cabo las actividades relacionadas con la planeación del proyecto a un alto nivel. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto y realizar estimaciones de tiempo, esfuerzo y costo.

Modelado del Negocio: Es la fase destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea automatizar para tener garantías de que el software desarrollado va a cumplir su propósito. Para la descripción y modelado del negocio pueden ser utilizadas diferentes técnicas como el Modelado de casos de uso del Negocio y la notación para el modelado de los procesos de negocio (BPMN por sus siglas en inglés). Artefactos que se generan:

- ✓ Modelo de procesos de negocio.
- ✓ Reglas del negocio.

Requisitos: El esfuerzo principal en esta fase es desarrollar un modelo del sistema que se va a construir. Incluye un conjunto de casos de uso, servicios que describen todas las interacciones que tendrán los usuarios con el software, estos responden a los requisitos funcionales del sistema. Además la especificación

[2] Siglas del inglés Capability Maturity Model Integration

de requisitos incluye requisitos no funcionales. Artefacto que se genera: Especificación de requisitos de software.

Análisis y Diseño: Durante esta fase a través de los modelos de análisis son definidos y refinados los requisitos, para conseguir una comprensión más precisa de los mismos y ayude a estructurar el sistema de una manera más fácil. Además en esta etapa, es modelado el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Esto contribuye a una arquitectura sólida y estable que se convierte en un plano para la implementación. Los modelos desarrollados en esta etapa son más formales y específicos de una implementación. Artefactos que se generan:

- Especificación de casos de uso del sistema.
- Diagramas de secuencia.
- Diagramas de clases del diseño.
- Diagrama de despliegue.

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

Pruebas Internas: Durante esta fase el proyecto verifica el resultado de la implementación probando según sea necesario cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Artefacto que se genera: Diseño de casos de prueba

Pruebas de Liberación: Se ejecutan las pruebas previamente se diseñadas en el laboratorio de calidad del centro antes de ser entregados al cliente para su aceptación.

Por tanto, el equipo de desarrollo de SITPC decidió utilizar la metodología RUP apoyándose en el Programa de Mejoras (CMMI) de la Universidad de las Ciencias Informáticas debido a que el proyecto se caracteriza por tener grandes dimensiones e implica muchos recursos tanto materiales como humanos. Además, se adapta muy bien a proyectos de alta complejidad y larga duración, y facilita el control sobre los cambios. Igualmente, en esta se genera gran cantidad de documentación, requerida por la dinámica del proyecto y necesaria en caso de cambios en el grupo de trabajo.

Herramientas CASE[3]

Para la realización del SITPC, el equipo de arquitectura decidió utilizar como herramienta CASE: Visual Paradigm for UML en su versión 8.0, debido a dos razones fundamentales:

[3] Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora

1. La aplicación a desarrollar será escrita en PHP[4] y esta herramienta se integra fácilmente con este lenguaje.
2. el equipo de desarrollo cuenta con una mayor experiencia en el empleo de esta herramienta en comparación a las otras, posibilitando un considerable ahorro de tiempo en el proceso de desarrollo del software debido a que no hay necesidad de capacitar a los desarrolladores.

Visual Paradigm for UML v8.0

Esta herramienta ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software (análisis y diseño orientados a objetos, construcción, pruebas y despliegue) a través de la representación de todo tipo de diagramas. Constituye una herramienta de software libre, gratuito y de probada utilidad para el analista. Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos. (International)

Esta herramienta se caracteriza por (Alejandre Nuevo, 2013):

- ✓ Disponibilidad en múltiples plataformas (Windows, Linux).
- ✓ Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- ✓ Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- ✓ Capacidades de ingeniería directa e inversa.
- ✓ Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- ✓ Modelado colaborativo con CVS y Subversión (control de versiones).
- ✓ Editor de detalles de casos de uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- ✓ Diagramas de flujo de datos.

Para darle una mejor estructura y organización al SITPC es necesario establecer un marco de trabajo que estandarice todo el proceso de desarrollo, siendo Symfony el seleccionado.

1.7 Framework de desarrollo

Un framework es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, en base a la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, librerías y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa

[4] Del inglés ***H****ypertext* ***P****re-processor*

una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio. (Ecured, 2014)

Symfony 2

Symfony 2 ha sido ideado para explotar todas las nuevas características de PHP 5.3, su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que ya no se deseen en un proyecto.

Symfony 2 también es el framework que más ideas incorpora del resto de los frameworks, incluso de aquellos que no están programados con PHP. (Eguiluz Javier, 2011)
Entre sus características más importantes se tienen:

- Alto rendimiento: Symfony 2 fue concebido desde el principio para ser rápido y para favorecer el rendimiento como mayor prioridad, por lo que es uno de los framework más rápidos. Es aproximadamente 3 veces más rápido que Symfony 1.4 o Zend Framework 1.10 y consume hasta 2 veces menos memoria.
- Flexible: Symfony 2 cuenta con un micro-kernel basado en un contenedor de inyección de dependencia y un manejador de eventos por lo que es muy fácil configurarlo a voluntad.
- Construido para desarrolladores: Symfony 2 proporciona las herramientas que en gran medida mejoran la productividad de los desarrolladores, como la famosa barra de depuración web, soporte nativo de entornos, páginas detalladas de errores y mucho más.
- Usabilidad avanzada: Es un framework fácil de utilizar gracias a que cuenta con una API[5] de desarrollo muy intuitiva.

Para el mapeo de la base dato, el equipo de desarrollo decidió utilizar el Mapeador de objeto relacional Doctrine en su versión 2.0 pues es este ya viene incluido con Symfony.

Mapeador Objeto-Relacional (ORM[6])

Las aplicaciones Symfony 2 no gestionan su información accediendo directamente a la base de datos. Crean, modifican y borran información mediante objetos PHP, en vez de crear y ejecutar sentencias SQL[7]. Esto es posible gracias a unas librerías externas llamadas ORM que son herramientas que permiten acceder, de forma efectiva, a las bases de datos relacionales desde un ambiente orientado a objetos, a

[5] del inglés: Application Programming Interface (interfaz de programación de Aplicaciones)
[6] del inglés: Object-Relational Mapping
[7] del inglés Structured Query Language

través de una interfaz que traduce la lógica de los objetos a la lógica relacional y viceversa. Dicha interfaz permite realizar lo que se conoce como el "mapeo objeto-relacional" y está formada por objetos que contienen el código necesario para acceder a los datos (Eguiluz, Javier, 2011).
Después de un análisis, el equipo de desarrollo del SITPC decidió utilizar Doctrine en su versión 2.1 debido al conjunto de ventajas que este ofrece y también por ser el ORM oficial de Symfony 2.

Doctrine v2.0

Es un ORM que proporciona persistencia transparente para objetos de PHP e incorpora una capa de abstracción de bases de datos de gran alcance. Una de sus principales características es la posibilidad de escribir consultas de base de datos en un dialecto SQL orientado a objetos propio llamado Doctrine Query Language (DQL), inspirado en Hibernates HQL. Esto proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad sin necesidad de duplicación de código innecesario. (Symfony, 2013). Para el trabajo con las vistas se escogió por parte del equipo de arquitectura Bootstrap en su versión 1.0 y jQuery en su versión 1.8 para el trabajo con las validaciones, la gestión de eventos e interacciones Ajax, brindando mucha potencialidad en el trabajo con aplicaciones web.

Bootstrap v1.0

Bootstrap es un framework desarrollado por Twitter que simplifica el proceso de creación de diseños web combinando CSS[8]3 y JavaScript[9]. Su mayor ventaja es que se pueden crear interfaces que se adapten a los distintos navegadores apoyándose en un framework potente con numerosos componentes webs que ahorrarán mucho esfuerzo y tiempo. Es un proyecto de código abierto con licencia Apache v2.0. (Mark Otto, 2013)

Características principales

Bootstrap ofrece una serie de plantillas CSS y ficheros JavaScript que permiten integrar el framework de forma sencilla y potente en proyectos webs.

- Permite crear interfaces que se adapten a los diferentes navegadores, a distintas escalas y resoluciones.
- Se integra perfectamente con las principales librerías JavaScript, por ejemplo jQuery.
- Ofrece un diseño sólido y estándares como CSS3/HTML[10]5.
- Es un framework ligero que se integra de forma limpia en proyectos web.

[8] del inglés cascading style sheets
[9] del inglés scripting Language
[10] del inglés Hypertext Markup Language

JQuery v1.8

JQuery es una biblioteca de JavaScript, que contiene procesos o rutinas ya listos para ser usados, lo que permite simplificar la forma de interactuar con los documentos HTML. Es un software libre y de código abierto debido a que se encuentra registrado bajo la Licencia MIT y la Licencia Pública General de GNU v2, lo que permite su uso en proyectos libres y privativos. (Flanagan, 2011)

Sus características principales son:

- Permite agregar efectos y animaciones personalizadas a las aplicaciones web.
- Soporta extensiones.
- Presenta varias utilidades como obtener información del navegador, operar con objetos y vectores, funciones para rutinas comunes, etc.
- Compatible con la mayoría de los navegadores web más usados.
- Posibilita el trabajo con Ajax.

Con el objetico de ahorrar tiempo y código, el equipo de arquitectura decidió usar el motor de plantillas Twig, ya que, cuando se utiliza el framework Symfony 2, se recomienda su uso para crear todas las plantillas de la aplicación, debido a sus grandes potencialidades y a su vez viene por defecto con Symfony 2.

Twig

Twig es un motor y lenguaje de plantillas para PHP muy rápido ya que compila las plantillas hasta código PHP regular optimizado y el costo general en comparación con código PHP regular se ha reducido al mínimo. Es seguro, pues tiene un modo de recinto de seguridad para evaluar el código de plantilla que no es confiable, lo que permite utilizarlo como un lenguaje de plantillas para aplicaciones donde los usuarios pueden modificar el diseño de la plantilla. Además es flexible debido a que es alimentado por flexibles analizadores léxico y sintáctico, que permiten al desarrollador definir sus propias etiquetas y filtros personalizados. (Pacheco, 2011)

Twig proporciona una serie de beneficios entre los que se encuentran:

- Plantillas fáciles de hacer y resultan muy intuitivas en el caso de contar con maquetadores o diseñadores de interfaces.
- Implementa un novedoso mecanismo de herencia múltiple de plantillas.
- Es extensible ya que permite la creación de extensiones propias.
- Gran rapidez: las plantillas son transformadas en PHP.
- El código se escribe en bloques, facilitando la herencia.

1.8 Lenguajes de programación

Las computadoras necesitan un lenguaje propio para poder interpretar y ejecutar las acciones que los usuarios deseen. El lenguaje que permite esto se conoce como lenguaje de programación, que se define como un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones (Sena, 2010). Seguidamente se describen los lenguajes de programación utilizados en el desarrollo del sistema ya sea para el lado del servidor o el lado del cliente.

PHP v5.3

PHP (Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para desarrollo web y que puede ser incrustado en HTML. (Group The PHP, 2014)

Es un lenguaje de programación interpretado, ideado originalmente para la construcción de páginas web dinámicas, extremadamente simple para el principiante, pero a su vez, ofrece muchas características avanzadas para los programadores profesionales. Su funcionamiento es muy básico, cuando el cliente realiza una petición al servidor para que le envíe una página web, el servidor ejecuta el intérprete de PHP, este procesa el script solicitado que generará el contenido de manera dinámica y el resultado es enviado por el intérprete al servidor, quien a su vez se lo envía al cliente (código HTML). También mediante extensiones es posible la generación de archivos PDF, Flash, así como imágenes en diferentes formatos.

Ventajas de este lenguaje:

- ✓ Mejor soporte para la Programación Orientada a Objetos, que en versiones anteriores era extremadamente rudimentario.
- ✓ Mejoras de rendimiento a versiones anteriores.
- ✓ Mejor soporte para MySQL con extensión completamente reescrita El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador.
- ✓ Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, se destaca su conectividad con MySQL y PostgreSQL.
- ✓ Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.

JavaScript v2.0

Es un lenguaje de programación interpretado (no es necesario compilarlo para que se ejecute), surgido como resultado del estándar ECMAScript y desarrollado por Netscape, es orientado a objetos, imperativo, débilmente tipado y dinámico. Se usa principalmente del lado del cliente, implementado como parte del

navegador web permitiendo mejoras significantes en las interfaces de usuario, puesto que es muy eficiente para la realización de validaciones y el desarrollo de web dinámicas. (Flanagan, 2002)

Sus características más significativas son:

- ✓ Es un lenguaje orientado a eventos. Cuando un usuario presiona un botón, interactúa con un enlace o mueve el puntero sobre algún otro elemento HTML, se pueden producir diferentes eventos y a través de JavaScript se pueden desarrollar scripts que ejecuten funciones en respuesta a estos eventos. Con esto se puede cambiar totalmente el aspecto de la página al gusto del usuario, evitándose tener en el servidor un página para cada gusto.
- ✓ Es un lenguaje interpretado, e sea, no necesita ser compilado. El navegador del usuario se encarga de interpretar las sentencias JavaScript contenidas en una página HTML y ejecutarlas adecuadamente línea por línea en tiempo de ejecución.
- ✓ Es soportado por la mayoría de los navegadores web como Internet Explorer, Netscape Navigator, Google Chrome, Opera, Mozilla Firefox, entre otros.

Para la implementación de la solución propuesta se definió por el equipo de arquitectura del SITPC utilizar como entorno de desarrollo NetBeans en su versión 7.2 ya que presenta soporte para Symfony 2.0 y Doctrine 2.0, igualmente posee soporte integrado para Subversion 1.7 e inferiores posibilitando realizar una administración eficiente del trabajo realizado. Como gestor de base datos se seleccionó PostgreSQL ya que brinda soporte al lenguaje PHP. Por la misma razón se escogió como servidor web el Apache en su versión 2.2 que soporta varios lenguajes de script, entre ellos el leguaje seleccionado para el desarrollo, permitiendo el acoplamiento de varios módulos que facilitan grandemente el trabajo con este lenguaje.

1.9 Entorno Integrado de desarrollo (IDE[11])

NetBeans v7.2

NetBeans es un entorno de programación que contiene un editor de código, un compilador, un depurador, un constructor de interfaz gráfica y otras herramientas, que van a facilitar las tareas de desarrollo y mantenimiento de las aplicaciones desarrolladas. Fue escrito completamente en Java y se encuentra bajo las licencias CDDL[12] y GPL[13], ambas de código abierto y libre, por lo que es un producto libre y gratuito, de código abierto y sin restricciones de uso. Cuenta con una comunidad de usuarios en constante crecimiento y con gran cantidad de socios en todo el mundo.

[11] del inglés: Integrated Development Environment
[12] Common Development and Distribution License
[13] General Public License

Es multiplataforma, soporta múltiples lenguajes, proporciona un conjunto de herramientas que permiten crear aplicaciones profesionales ya sean estas web, de escritorio o de otro tipo, además, cuenta con una interfaz muy amigable e intuitiva que permite facilitar el trabajo de los desarrolladores.
Esta versión presenta mejoras en cuanto al rendimiento sobre todo en la velocidad de escaneo y en la exploración de proyecto en segundo plano, evitando el bloqueo de las funciones de edición y navegación. También el editor de PHP en su versión 5.4 e inferiores es más robusto y ágil, presentando mejoras en el completamiento de código y reconocimiento de sintaxis. (Sitio oficial, 2015)

1.10 Sistemas Gestores de Bases de Datos

Un sistema gestor de bases de datos o SGBD (conocido también por las siglas DBMS procedentes del inglés, Data Base Management System), es un software o conjunto de programas que permite controlar la organización, almacenamiento y recuperación de datos en una base de datos. También controla la seguridad y la integridad de la base de datos. El SGBD es la aplicación que sirve de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan, proporcionando un entorno eficiente a la hora de almacenar y recuperar los datos, garantizando la seguridad de los mismos, incluyendo la especificación de tipos, estructura y sus restricciones, además permitiendo la manipulación mediante actualizaciones, la realización de consultas y generación de informes. Para el desarrollo del SITPC el equipo arquitectónico decidió utilizar PostgreSQL, basándose en sus características y ventajas.

PostgreSQL v9.3

Es el SGBD objeto-relacional de código abierto más potente del mercado, cuenta con una arquitectura suficientemente probada, que se ha ganado una sólida reputación de confiabilidad, integridad de datos y corrección, convirtiéndolo en el líder entre los gestores de bases de datos de código abierto. Sus comunidades son cada vez más grandes y muy activas, brindando documentación muy bien organizada, pública y libre, con comentarios de los propios usuarios. Además, utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Es un sistema muy seguro que funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez a el sistema. El conjunto de funcionalidades de PostgreSQL no sólo es comparable a los mejores sistemas gestores de datos privativos, sino que las superan en características avanzadas, extensibilidad, seguridad y estabilidad. Entre sus características más importantes se pueden mencionar: (PostgreSQL, 2013)

- ✓ Es una base de datos 100% ACID compliant (Atomicity, Consistency, Isolation and Durability por sus siglas en inglés.). Atomicidad, Consistencia, Aislamiento y Durabilidad en español.

- ✓ Es altamente escalable, tanto en la cantidad bruta de datos que puede manejar como en el número de usuarios concurrentes que puede atender, realizando esto de forma más eficiente que muchos otros gestores.
- ✓ Implementa el uso de rollback's, subconsultas y transacciones, haciendo su funcionamiento mucho más eficaz, ofreciendo soluciones en disimiles campos.
- ✓ Permite la declaración de funciones propias, así como la definición de disparadores.
- ✓ Incluye herencia entre tablas.
- ✓ Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

PostgreSQL v9.2 es la versión establecida para usar en el SITPC por las normativas de la dirección del proyecto. Este gestor será administrado por una aplicación gráfica llamada pgAdmin III que es una de las más completas y populares con licencia Open Source. Está escrita en C++ y es compatible para sistemas operativos como: GNU/Linux, FreeBSD, Solaris, Mac OS y Windows. Es capaz de gestionar versiones a partir de PostgreSQL 7.3, ejecutándose en cualquier plataforma. Posee una interfaz gráfica que soporta todas las características de PostgreSQL y facilita enormemente la administración. PgAdmin es desarrollado por una comunidad de expertos de PostgreSQL en todo el mundo y está disponible en más de una docena de idiomas. Es un software libre publicado bajo la licencia PostgreSQL.

1.11 Servidor Web

Un servidor Web es un programa que procesa una aplicación del lado del servidor la cual escucha las peticiones de los clientes (usuarios o navegantes), realizadas a través de un navegador web y las atiende o satisface, proporcionando una respuesta o los recursos que se soliciten usando el protocolo HTTP[14] o el protocolo HTTPS[15] (la versión cifrada y autenticada), pertenecientes a la capa de aplicación del modelo OSI[16]. Esta aplicación se encarga de gestionar cualquier sistema web en el lado del servidor, permitiendo realizar conexiones bidireccionales y/o unidireccionales así como síncronas o asíncronas con el cliente. En otras palabras el servidor web se mantiene a la espera de la solicitud de un navegador, resuelve la petición de este y da respuesta a la misma, generalmente enviando una página HTML que el navegador interpreta y muestra en la pantalla al usuario. Para el desarrollo del SITPC el equipo de arquitectura decidió, de entre los servidores web existentes, trabajar con el servidor web Apache en su versión 2.2.

[14] Hypertext Transfer Protocol (en español protocolo de transferencia de hipertexto).
[15] Secure HTTP.
[16] Open System Interconnection (modelo de referencia de Interconexión de Sistemas Abiertos).

Servidor web Apache v2.2

Apache es un servidor web HTTP muy potente, gratuito, de código abierto y flexible debido a que es altamente configurable y de diseño modular. Evita la sobrecarga del servidor con módulos innecesarios en el momento de ejecutar alguna acción. Proporciona también el poder de implementar otro módulo necesario en algún momento y acoplárselo. Además es muy sencillo de mantener y configurar ya que los archivos de configuración están en ASCII[17], por lo que tienen un formato simple y pueden ser editados sencillamente con un editor de texto.

Apache también permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto, además le va a permitir al administrador tener un gran control de todo lo que sucede en el servidor puesto que es altamente configurable en la creación y gestión de logs, permitiendo la creación de ficheros de log a la medida del administrador. Es un servidor multiplataforma y presenta abundante documentación. En la actualidad es uno de los servidores más utilizados en el mundo por todas las ventajas que brinda. (Sitio oficial, 2015)

1.12 SubVersion (SVN)

Es un sistema de control de versiones de código abierto que maneja ficheros y directorios, así como los cambios introducidos en ellos, con el tiempo. Permite recuperar versiones antiguas de datos o examinar el historial de cómo estos cambiaron. Opera a través de redes, lo que permite que varios desarrolladores puedan trabajar en un mismo proyecto a la vez, en diferentes equipos.

SubVersion v1.5

Subversion es un sistema libre / código abierto de control de versiones (VCS). Es decir, Subversion maneja ficheros y directorios, y los cambios introducidos en ellos, a través del tiempo. Esto le permite recuperar versiones antiguas de sus datos o examinar la historia de cómo cambiaron sus datos. En este sentido, muchas personas piensan de un sistema de control de versiones como una especie de "máquina del tiempo". Subversion puede funcionar a través de redes, lo que permite que pueda ser utilizado por personas en diferentes equipos. En algún nivel, la capacidad para varias personas para modificar y administrar el mismo conjunto de datos de sus respectivas ubicaciones fomenta la colaboración. El progreso puede ocurrir más rápidamente sin un único conducto a través del cual deben producirse todas las modificaciones. Y debido a que el trabajo se versiona, no tienes que temer que la calidad es la compensación por la pérdida

17 ASCII (acrónimo inglés de American Standard Code for Information Interchange — Código Estándar Estadounidense para el Intercambio de Información)

de ese conducto, si se hace algún cambio incorrecto a los datos, simplemente deshaga ese cambio. (Ben Collins-Sussman, 2010)

El equipo de desarrollo del proyecto seleccionó la arquitectura MVC teniendo en cuenta las facilidades de adaptación al cambio que ofrece, además de estar diseñado para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. También al separar la vista de la lógica de negocio y el controlador, permite la reutilización de los mismos y si en algún momento uno de sus componentes funciona mal, puede reemplazarse sin que los demás se vean afectados. Como buenas práctica de programación se hará uso de los patrones de diseños con el objetivo de obtener una solución de alta calidad y que satisfaga las necesidades del cliente.

1.13 Arquitectura de Software

Una arquitectura de software describe los componentes básicos de un sistema de software y su combinación interna. En el marco del desarrollo de software representa la decisión de diseño más temprana. Es determinada básicamente por criterios de calidad como la modificabilidad, mantenibilidad, seguridad y el rendimiento. Una arquitectura de software una vez establecida es modificable más tarde solo con gran esfuerzo. La decisión acerca de su diseño es por eso uno de los puntos más críticos e importantes en el proceso de desarrollo de un software. Para poder funcionar con éxito, la arquitectura de software debe ser sintonizada con los restantes factores del proyecto de software. Una arquitectura de software bien configurada facilita a los usuarios y desarrolladores la comprensión del sistema. Factores importantes que influyen en la aptitud de la arquitectura de software son la planificación de proyectos, el análisis de riesgo, la organización, el proceso de desarrollo, los ciclos de trabajo, el hardware, la garantía de calidad y los requisitos. (Voigtmann, 2013)

Estilos arquitectónicos

Se pueden definir como un “conjunto de reglas de diseño que identifican las clases de componentes y conectores que se pueden utilizar para componer un sistema o subsistema, junto con las restricciones locales o globales de la forma en que se lleva a cabo la composición. Es en gran medida la interacción entre los componentes, mediados por conectores, lo que confiere a los distintos estilos sus características distintivas”. (Reynoso, 2014)

Modelo-Vista-Controlador (MVC)

Es un patrón de arquitectura de software encargado de separar la lógica de negocio de la interfaz del usuario, tratando al modelo, las vistas y los controladores como entidades separadas; lo que hace posible que cualquier cambio producido en el modelo se refleje automáticamente en cada una de las vistas.

Entre sus ventajas se encuentran (Reynoso, 2004):

- Soporte de vistas múltiples: Dado que la vista se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente. Por ejemplo, múltiples páginas de una aplicación web pueden utilizar el mismo modelo de objetos, mostrado de maneras diferentes.
- Adaptación al cambio: Los requisitos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocios. Los usuarios pueden preferir distintas opciones de representación, o requerir soporte para nuevos dispositivos como teléfonos celulares o PDAs18. Dado que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo.

Arquitectura en capas

Se define como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. El objetivo de esta arquitectura es separar la lógica de negocios de la lógica de diseño. Su principal ventaja consiste en que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que suceda algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. (Reynoso, 2014)

1.14 Patrones de diseño orientados a objetos

Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, y describe la esencia de la solución a ese problema, de tal modo que pueda utilizarse esta solución un millón de veces más, sin siquiera hacerlo de la misma manera dos veces. (Alexander, 1997)

Un patrón es:

- Una solución a un problema en un contexto particular.
- Recurrente: hace la solución relevante a otras situaciones.
- Enseña: permite entender cómo adaptarlo a la variante particular del problema donde se quiere aplicar.

[18]Del inglés *personal digital assistant*

- Tiene un nombre para referirse al patrón.

Los patrones facilitan la reutilización de diseños y arquitecturas software que han tenido éxito. (Juan, 2004) Dentro de los patrones de diseño existen dos grupos principales: los patrones GRASP[19] (patrones generales de software para asignación de responsabilidades) y los patrones GOF[20] que es el nombre con el que se conoce por lo general a los autores del libro Design Patterns. (Gamma, 1995)

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Este grupo contiene patrones como:

- Alta Cohesión
- Bajo Acoplamiento
- Controlador
- Experto
- Polimorfismo
- Fabricación Pura

Los patrones de diseño GOF son 23 y se clasifican según su propósito en Creacionales, Estructurales, y de Comportamiento y según su ámbito en Objeto y de Clase. (Gang of Four, 2008)

Para el desarrollo del SITPC los diseñadores deciden utilizar el patrón GOF estructural **Decorador (decorador),** el cual añade funcionalidades a objetos dinámicamente permitiendo no tener que crear sucesivas clases que hereden de la primera incorporando la nueva funcionalidad, sino otras que la implementan y se asocian a la primera y el patrón GOF creacional **Factory.** De los patrones GRASP se decide utilizar el **Controlador**, **Creador**, **Experto**, **Bajo Acoplamiento** y **Alta cohesión**.

Conclusiones

La elaboración el marco teórico de la investigación permitió arribar a las siguientes conclusiones:

- El análisis del estado del arte realizado, evidenció que los sistemas informáticos para la gestión de procesos judiciales desarrollados actualmente no satisfacen las necesidades de negocio y de soberanía tecnológica de los TPC, por lo que se debe informatizar el procedimiento de Revisión de la materia Económica.
- La fundamentación de la metodología, los lenguajes, las herramientas y tecnologías a utilizar, permitieron sentar las bases técnicas, para dar comienzo al desarrollo del procedimiento Revisión del subsistema Económico.

[19] Por sus siglas en inglés General Responsibility Assignment Software Patterns
[20] Por sus siglas en inglés Gang of Four

CAPÍTULO 2: SOLUCIÓN PROPUESTA

2.1 Introducción

En consecuencia al estudio realizado en el capítulo anterior, se presentará una solución desde el punto de vista ingenieril y según plantea la metodología fundamentada en el epígrafe. Se realiza un análisis de la descripción del modelo de negocio y el modelo de diseño e implementación, obteniéndose los siguientes artefactos: Descripción de procesos de negocio, Reglas del negocio, Especificación de requisitos de software, Especificación de casos de uso del sistema y otros artefactos definidos en el Plan de Desarrollo del Proyecto Informatización de los TPC. Se exponen además los patrones de diseño utilizados en la solución propuesta y los estándares de codificación que se tomaron en consideración para el desarrollo del SITPC.

2.2 Modelado del proceso de negocio

Descripción del proceso Revisión de la materia Económico

A continuación se muestra un resumen del procedimiento Revisión Económica en el TSP, con el objetivo de obtener un marco común para el equipo de proyecto, los clientes y los usuarios finales, en la búsqueda de obtener posteriormente los requisitos funcionales y no funcionales del sistema. Para ver el análisis del negocio completo ver los artefactos:

- “CEGEL-SITPC-Descripción_de_procesos_de_negocio_Revisión”.
- “CEGEL-SITPC-Reglas_de_negocio_Revisión_Econ”.

Objetivo	Revisar en el TSP sentencias y autos definitivos que hayan sido declarados firmes.
Evento(s) que lo genera(n)	Presentar la demanda en el TSP.
Pre-condiciones	Haber agotado todas las vías para resolver el conflicto existente entre las partes en las instancias inferiores al TSP.
Marco legal	LPCALE. 1996. Artículo 794.
Reglas de negocio	Regla del negocio 1 – 16.
Responsable	Secretaria, Juez, Abogado, Partes.
Clientes internos	TSP.
Clientes externos	Partes.
Entradas	Escrito de presentación de la demanda. Documentos que acompañan.

Flujo de eventos	
Flujo básico	
1.	Presentar demanda. Se presenta la demanda en el TSP para pedir revisión. Reglas 1, 2, 5, 6, 7.
2.	Dar entrada a la demanda. La secretaria le da entrada a la demanda y la registra.
3.	Disponer sobre la radicación. El presidente de tribunal manda a realizar la radicación y el turnado de la demanda.
4.	Radicar y turnar. Se le asigna una numeración al expediente y la secretaria le asigna un juez ponente para atender el caso según dispuso el presidente del tribunal.
5.	Disponer sobre la demanda. El juez a partir de toda la documentación del caso podrá admitir, rechazar o mandar a subsanar la demanda.
6.	Crear auto de suspensión de la ejecución. El juez admite la demanda e inmediatamente se suspende la ejecución del fallo. Regla 8 y 9. Ver flujos alternos 8.a Se dispuso la subsanación, 8.b Se dispuso que no hay lugar a admitir, 8.c Las partes solicitan la suspensión y el tribunal está de acuerdo y 8.d Las partes solicitan la suspensión y el tribunal no está de acuerdo.
7.	Dar traslado a las partes. Se les hace llegar a las partes una copia del escrito de promoción para informar sobre el proceso de revisión. Además se genera un oficio de comunicación para que el tribunal que atendió el caso en el TPP suspenda la ejecución del fallo.
8.	Realizar contestación. Las partes brindan su opinión y pueden presentar pruebas para el proceso. Regla 10.
9.	Declarar proceso concluso para sentencia. El proceso se declara concluso para sentencia, lo que significa que las partes ya no pueden aportar nada más a la causa. Ver flujos alternos 11.a Se realiza solicitud de práctica de pruebas y 11.b Se realiza solicitud de práctica de pruebas pero se deniegan las pruebas.
10.	Realizar acta de discusión y votación de sentencia. El tribunal realiza una votación para pronunciar la sentencia. Ver flujo alterno 12.a Es necesaria la práctica de PPMP.
11.	Pronunciar sentencia. El juez se pronuncia sobre la sentencia declarándola con o sin lugar. Regla 13, 14 y 15.

12.	Dictar nueva sentencia. Si la sentencia es declarada con lugar se dicta una nueva sentencia y se anula la anterior. Ver flujo alterno 14.a Se declara con lugar la demanda y se realiza la retroacción y 14.b Se declara sin lugar y se deja sin efecto la suspensión de la ejecución del fallo.
13.	Notificar. Notificar a las partes sobre el resultado de la sentencia.

Tabla 2 Descripción del procedimiento Revisión del subsistema Económico

2.3 Flujo a informatizar

A continuación se describe el flujo de actividades para el negocio definido como el objeto a informatizar en la investigación, para instancia Suprema de los Tribunales Populares Cubanos. Para consultar el flujo de actividades que se realiza durante todo el proceso de Revisión en el Tribunal Supremo Popular, consultar el (***Anexo 1***).

Actor	**Descripción**
Partes	Actor que dará inicio al proceso al presentar una demanda y la documentación que la acompaña.
Secretaria TSP	Actor encargado de registrar la demanda, radicar, notificar, dar traslado a las partes y registrar el escrito de contestación de la demanda.
Presidente de sala TSP	Actor encargado de turnar la nueva demanda a un juez ponente.
Juez ponente	Actor encargado de disponer sobre una demanda o escrito que se le turno.

Tabla 3 Descripción de los actores del sistema

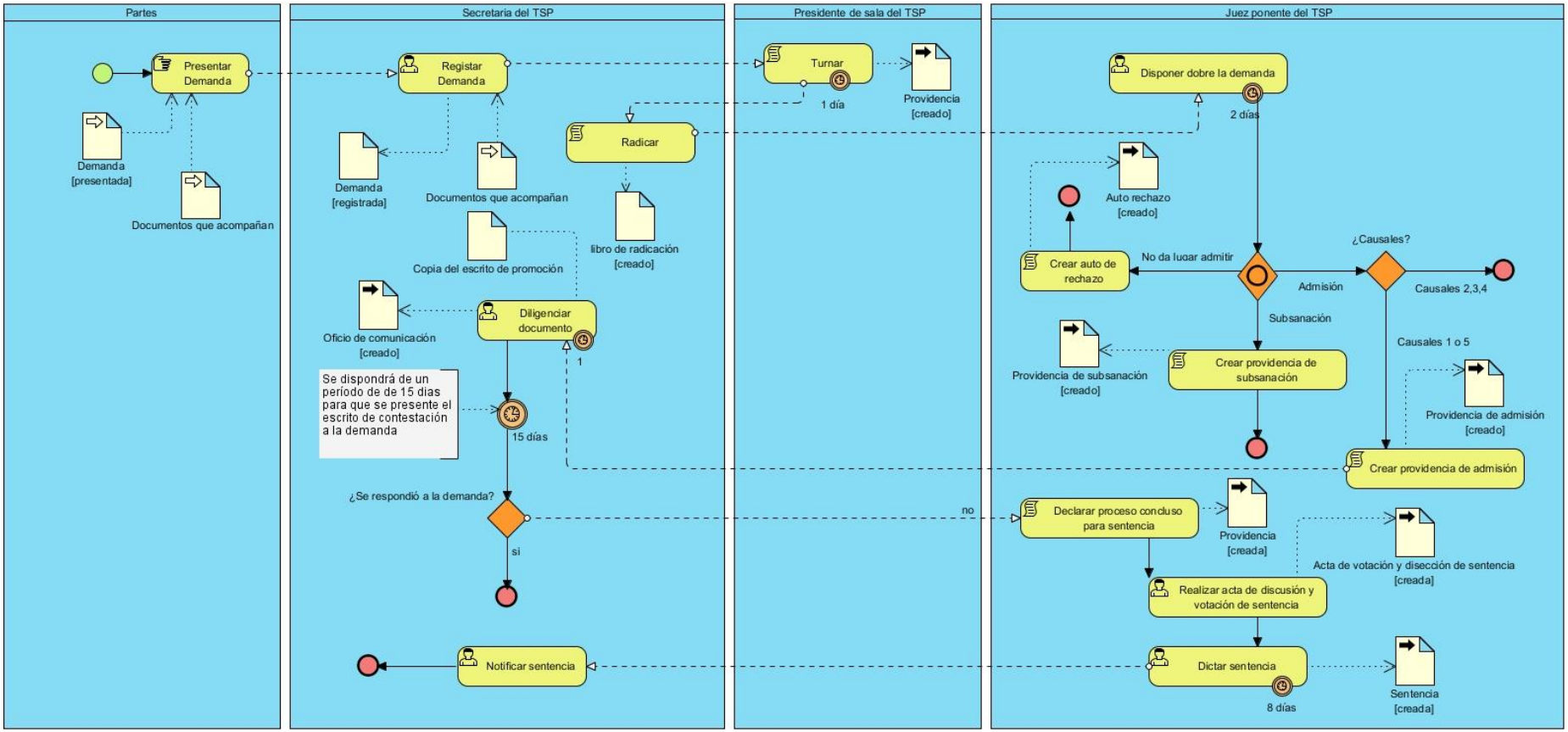


Figura 1 Flujo a informatizar

2.4 Especificación de requisitos de software

Durante el desarrollo del análisis y especificación de requisitos se identificaron 40 requisitos funcionales y 45 requisitos no funcionales. Para mayor información y más detallada ver el documento: "CEGEL-SITPC_Especificación_de_requisitos_de_software_Revisión" adjunto a la investigación y para los requisitos no funcionales ver documento "CEGEL-SITPC-0113-Requisitos No Funcionales-v1.3". A continuación se muestra el análisis y especificación del paquete Presentación de la demanda.

Descripción de actores del paquete Presentación de la demanda

Actor	Objetivos
Secretaria TSP	Actor encargado de Registrar la demanda.
Partes	Actor que dará inicio al proceso al presentar una demanda y la documentación que la acompaña.

Tabla 4 Descripción de actores del paquete Presentación de la demanda

Requisitos funcionales del paquete Presentación de la demanda

Nº	Nombre	Descripción	Prioridad para el cliente	Complejidad
RF.1	Buscar expediente archivado	El sistema permitirá buscar los expedientes archivados y seleccionar al que se le hará una demanda de revisión.	Baja	Baja
RF.2	Buscar un abogado	El sistema permitirá buscar el abogado que representará al demandante.	Baja	Baja
RF.3	Listar abogado	El sistema permitirá listar los abogados para seleccionar el que representa a las partes.	Medio	Medio
RF.4	Listar causal	El sistema permitirá listar las causales y seleccionar por cuál de ella se presenta la demanda.	Medio	Medio
RF.5	Adicionar partes	El sistema permitirá adicionar un una parte ya sea demandante o	Medio	Alta

		demandado		
RF.6	Modificar partes	El sistema permitirá cambiar el demandante o demandado que ya ha sido adicionado.	Medio	Medio
RF.7	Eliminar partes	El sistema permitirá eliminar una parte que ya ha sido adicionada.	Medio	Medio
RF.8	Adicionar persona fiscal	El sistema permitirá adicionar una persona de tipo fiscal.	Medio	Alta
RF.9	Adicionar persona natural	El sistema permitirá adicionar una persona de tipo natural.	Medio	Alta
RF.10	Adicionar persona natural	El sistema permitirá adicionar una persona de tipo natural.	Medio	Alta
RF.11	Modificar datos personas	El sistema permitirá modificar los datos de una persona.	Medio	Alta
RF.12	Buscar persona	El sistema permitirá buscar a la persona ya sea demandante o demandado	Baja	Media
RF.13	Adicionar documento en formato duro	El sistema permitirá adicionar el nombre de un nuevo documento en formato duro.	Medio	Medio
RF.14	Adicionar documento en formato digital	El sistema permitirá adicionar un nuevo documento en formato digital.	Medio	Medio
RF.15	Eliminar documento	El sistema permitirá eliminar un documento que ya ha sido adicionado.	Medio	Medio
RF.16	Registrar demanda	El sistema permitirá registrar una demanda de revisión económica.	Alta	Alta

Tabla 5 Requisitos funcionales del paquete Presentación de la demanda

2.5 Especificación de caso de uso (CU)

En la fase de análisis y diseño se elabora el diagrama de casos de uso del sistema, se emplearon patrones tales como:

- ✓ **Inclusión:** Se evidencia cuando se incluyen otros casos de uso en la descripción del caso de uso en cuestión. Ejemplo: el CU Datos primarios de encuentra incluido en el CU Registrar demanda.
- ✓ **Extensión:** Se evidencia cuando un caso de uso puede o no estar presente en la realización de otro caso de uso. Ejemplo: CU Documentos adjuntos representa las pruebas que se presentan con la demanda (CU Registrar demanda), las cuales pueden ser presentadas o no.

Una vez obtenidos los requisitos y teniendo en cuenta los patrones explicados anteriormente, el diagrama de casos de uso para el paquete Presentación de la demanda quedó elaborado como se muestra en la figura 2. Para obtener información más detallada sobre los paquetes de caso de uso y sus descripciones, consultar el documento “CEGEL-SITPC-Especificación_de_casos_de_uso”.

Objetivo	Registrar una demanda	
Actores	Secretaria TSP, Abogado	
Resumen	El CU se inicia cuando el abogado en representación de la parte demandante presenta una demanda de revisión y la documentación que la acompaña ante la secretaria TSP ante la secretaria TSP encargada de darle registro a la demanda.	
Complejidad	Alta	
Prioridad	Crítica	
Precondiciones	Que se haya dictado sentencia firme o auto definitivo en el TPP.	
Postcondiciones	Queda registrada una nueva demanda de revisión El expediente pasa al estado: pendiente a disposición	
Flujo de eventos		
Flujo básico <Registrar una demanda de revisión económica>		
	Actor	**Sistema**
1	Accede por el menú izquierdo a la opción Registrar demanda	Buscar expediente. Ver CU Buscar expediente
2	Introduce los datos de la demanda	Muestra una interfaz para llenar los datos de la demanda.

		Valida que los datos sean correctos. Para datos incorrectos ver evento 1. Gestionar abogado. Ver CU Gestionar abogado. Adicionar una persona. Ver CU Gestionar persona Eliminar una persona. Ver Sección 1: Eliminar persona Modificar una persona. Ver CU Gestionar persona
3	Selecciona dar siguiente	Se guardan los datos en la entidad de presentación y se muestra la segunda pantalla del registrar demanda.
4	Introduce los datos del documento que se va adjuntar	Adjuntar documentos. Ver CU adjuntar documentos. Eliminar documento. Ver Sección 2: Eliminar persona
5	Selecciona finalizar	Se guardan los datos, muestra un mensaje de confirmación, se realizar Radicar y turnar. Ver CU radicar y turnar. Termina el caso de uso.
Flujos alternos		
evento	**< Registrar una demanda de revisión económica >**	
1	Actor	Sistema
2	Datos incorrectos	Muestra un mensaje de datos incorrectos señalando los campos que no cumplen con la validación.
3	Selecciona botón cancelar de la primera pantalla del registrar	Cancela el registro de la demanda y redirecciona a la pantalla principal de la secretaria.
4	Selecciona botón cancelar de la segunda pantalla del registrar	Cancela el registro de la demanda y redirecciona a la pantalla principal de la secretaria.
5	Selecciona el botón atrás en la segunda pantalla	Carga la primera pantalla del registrar con los datos que fueron introducidos.
Secciones		
Flujo básico < Registrar una demanda de revisión económica >		
	Actor	Sistema
1	Selecciona el botón Eliminar persona	Se muestra un mensaje de confirmación del borrado. Se eliminan los datos del campo.
2	Selecciona el botón Eliminar	Se muestra un mensaje de confirmación del borrado. Se

	documento	eliminan los datos del campo.
Relaciones	CU incluidos	Buscar expediente. Ver CU Buscar expediente Gestionar abogado. Ver CU Gestionar abogado Gestionar persona. Ver CU Gestionar persona Radicar y turnar. Ver CU Radicar y turnar Datos primarios. Ver CU Datos primarios
	CU extendidos	Adjuntar documento (prueba): Ver CU Documento adjuntos

Tabla 6 Especificación de caso de uso del paquete Presentación de la demanda

Diagrama de caso de uso del paquete Presentación de la demanda

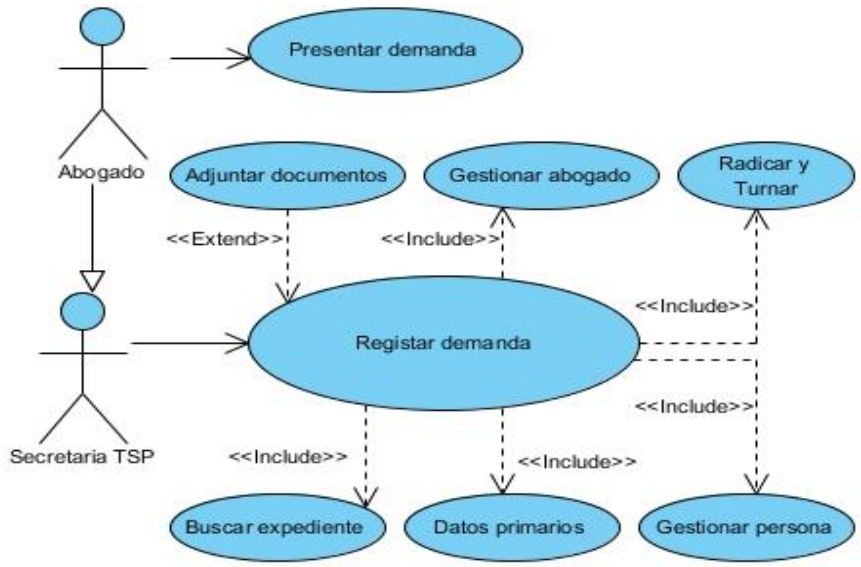


Figura 2 Diagrama de caso de uso del paquete Presentación de la demanda

2.6 Arquitectura del sistema

El desarrollo de SITPC está basado en una arquitectura en capa permitiendo dividir los problemas a resolver en subproblemas y que cada capa contenga solo las funcionalidades relacionadas con sus tareas, proporcionando una alta reutilización del código y un bajo acoplamiento. Se hace uso del patrón arquitectónico Modelo–Vista–Controlador (MVC), el cual permite la reutilización e independencia entre las capas, además permite que se puedan realizar cambios en las mismas sin tener que modificar las otras capas, facilita la estandarización, la utilización de los recursos y la administración. La estructura que define el marco de trabajo Symfony2 para la separación de estas capas deja bien delimitado dónde quedan las

clases del modelo, las de la vista y las del controlador. En el caso de la capa Modelo está dividida en: capa de acceso a datos y capa de abstracción de base de datos. A modo de ilustrar lo explicado se muestra la siguiente imagen:

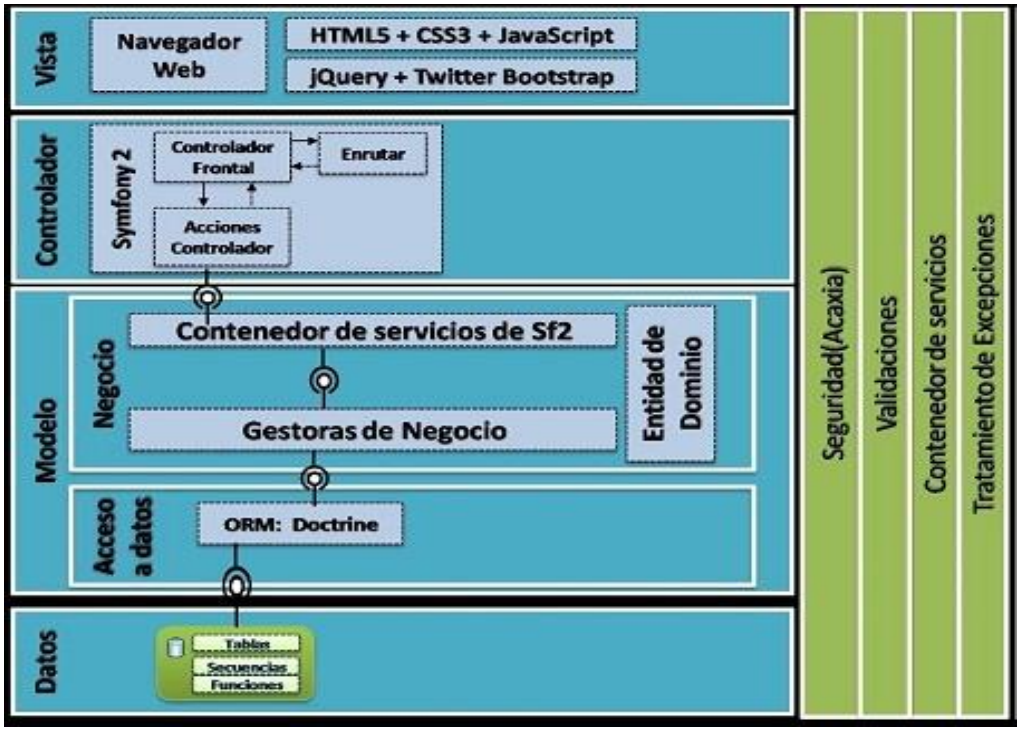


Figura 3 Arquitectura base

Para un mejor entendimiento de la arquitectura se realiza un desglose de la misma con el fin de demostrar su aplicación durante todo el proceso de desarrollo de la solución propuesta.

Capa Vista

En esta capa se encuentra todo lo que se refiere a la visualización de la información, el diseño, colores, estilos y la estructura visual de las páginas con que interactúan directamente los usuarios finales. En Symfony 2, el framework utilizado para desarrollar la solución, queda evidenciada esta capa a través del uso del motor de plantillas Twig y las librerías jQuery y Bootstrap, que facilitan la construcción de interfaces bien acabadas y agradables a la vista del usuario.

Asimismo, se aprovecha la herencia de plantillas a tres niveles que permite Twig pues la plantilla_SIT.html.twig que hereda de base.html.twig, incluye el menú y las funcionalidades dependiendo de la materia donde se encuentre y las del tercer nivel que son las de cada módulo que permiten implementar el contenido del área de trabajo.

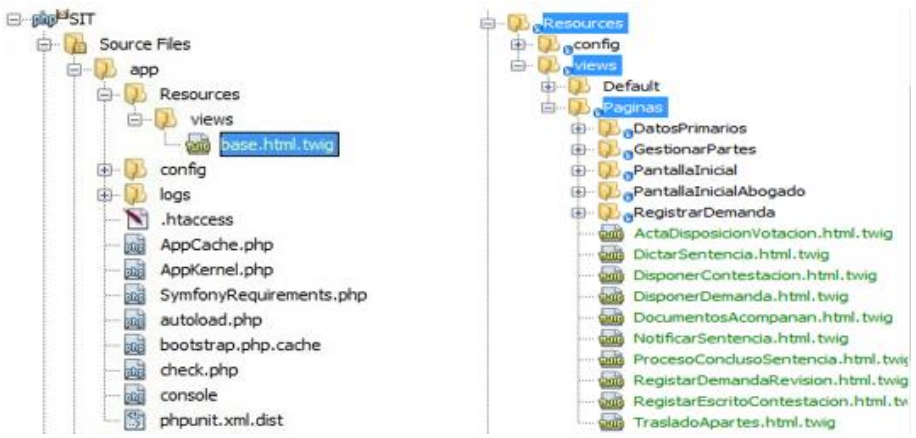


Figura 4 Contenido de la carpeta "Paginas" y ubicación de base.html.twig

Capa Controladora

La responsabilidad de esta capa es procesar y mandar a mostrar los datos obtenidos por la capa de acceso a datos. Es decir, trabaja de intermediario entre la vista y el modelo. Este recibe las solicitudes o peticiones realizadas por el usuario desde la vista, ejecuta las acciones pertinentes y devuelve una respuesta a la interfaz con los resultados de las operaciones realizadas. En el SITPC el uso de esta capa se observa a través de las clases Controller que se encargan de realizar una funcionalidad completa y específica, además en ellas se establecen las reglas que deben cumplirse.

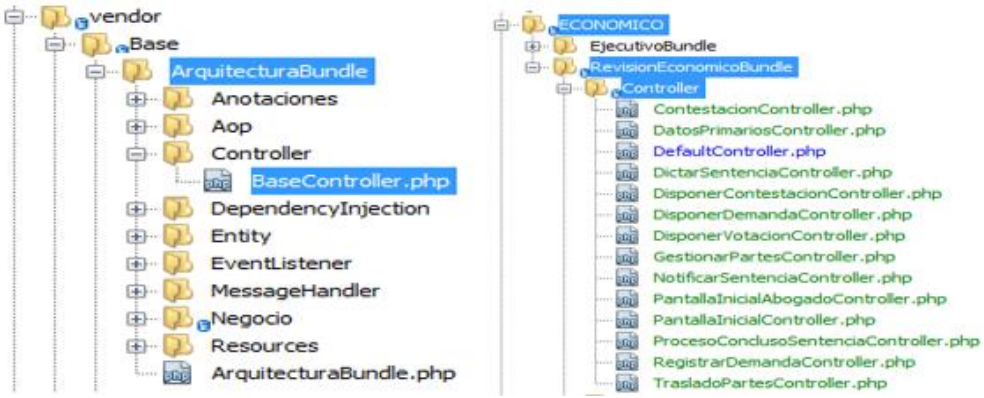


Figura 1 Contenido de la carpeta Controller y ubicación de BaseController

Capa de negocio

- ✓ **Capa de acceso a datos**

Esta capa tiene la responsabilidad de conectar el gestor de base de datos y el manejo de la lógica del negocio. El controlador se conecta con esta capa a través de las clases gestoras (Gtr) que son las encargadas de manejar la lógica del negocio, estas recibirán a través del controlador la información enviada

desde la vista e interactuará con los datos mediante el ORM Doctrine haciendo uso del Entity Manager[21], este a su vez, se comunican con las clases Repository donde se encuentran las consultas más complejas a la base de datos, aislándolas del modelo.

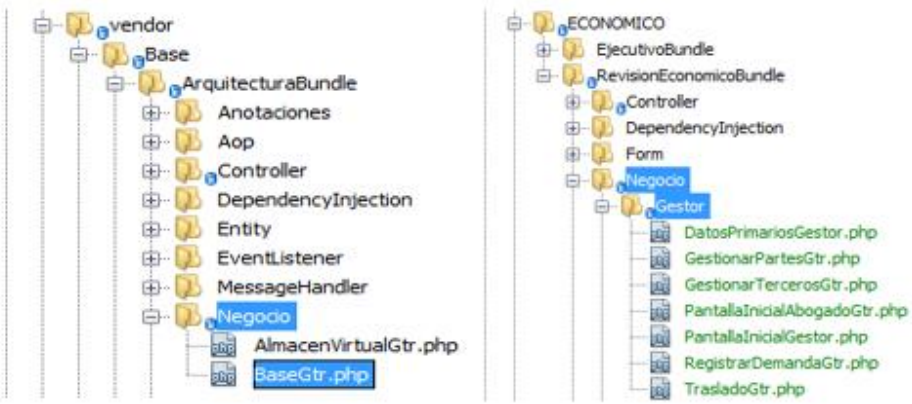


Figura 5 Contenido de la carpeta Gestor y ubicación de BaseGtr

- ✓ **Capa de datos**

Esta capa está formada por las clases entidades y contiene los esquemas, las tablas, las vistas y los procedimientos necesarios para garantizar el almacenamiento y persistencia de la información utilizada por la aplicación.

Componentes Verticales:

Validaciones: Las validaciones se realizan tanto del lado del cliente como del lado del servidor, para asegurar la entrada de valores correctos a la aplicación, garantizando un correcto funcionamiento de la misma.

```
if ($form->isValid()) {
if ($idDemanda == -1) {
$idDemanda = $this->getGestor()->RegistrarDatosDemanda($request, $this->getUser(), $rol, $idExpediente);
} else {
$this->getGestor()->ActualizarDatosDemanda($idDemanda, $request, $this->getUser(), $rol, $idExpediente);
}
return $this->redirect($this->generateUrl('economico_rev_registrar_demanda_docAdjuntos',
array('idDemanda' => $idDemanda, 'idExpediente' => $idExpediente)));
}
```

Figura 6 Validaciones en el servidor

[21] Interfaz pública de Doctrine 2.

Contenedor de servicios: Permite que las respuestas de la aplicación sean más rápidas, potencia la reutilización de código, además estandariza y centraliza la forma de construir objetos en toda la aplicación. En la solución propuesta los servicios se encuentran declarados y publicados en el archivo services.yml, los cuales pueden ser instanciados desde cualquier clase controladora haciendo uso del atributo container. En la figura 8 se pueden observar los servicios que se emplean en las clases gestoras.

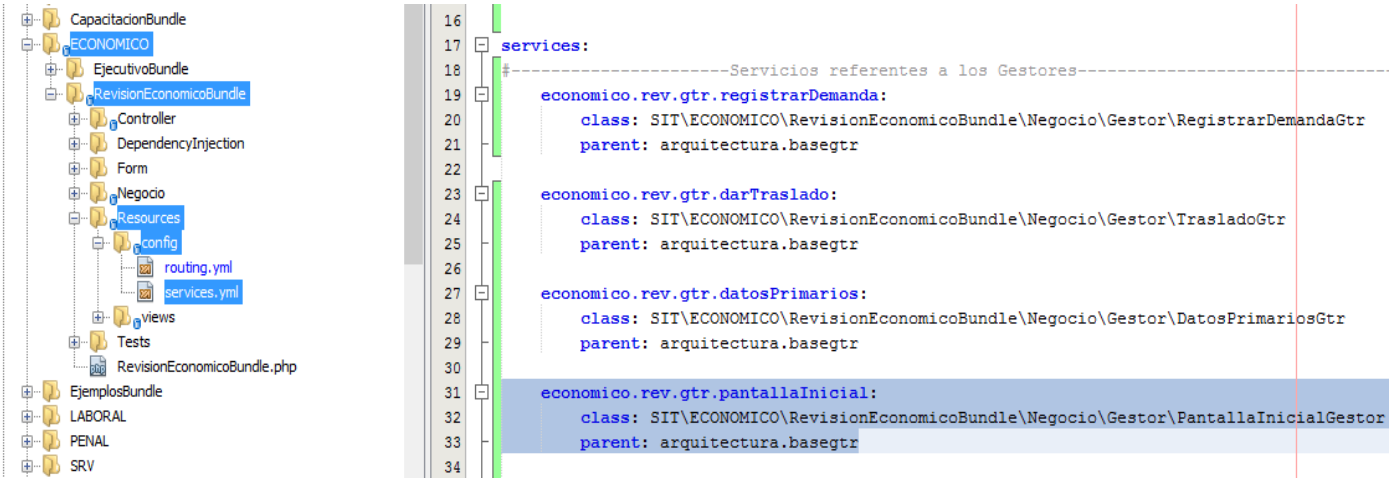


Figura 7 Contenedor de servicios

Tratamiento de excepciones: El tratamiento de excepciones lo brinda Symfony2 de forma interna, se encuentra de forma transversal en toda la aplicación pues en cualquiera de las capas se puede lanzar una excepción.

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

    return $this->container->get('economico.rev.gtr.registrarDemanda');
}
```

Figura 8 Tratamiento de excepciones

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

administración de perfil permite al usuario la personalización de su perfil en el sistema, posibilitándole modificar datos personales, así como el cambio de su contraseña cada cierto tiempo.

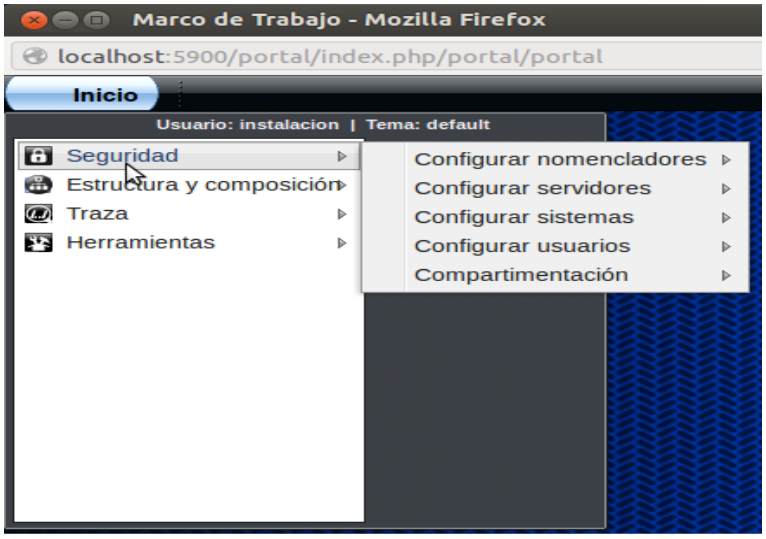


Figura 9 Acaxia sistema encargado de la seguridad del sistema

2.7 Modelo de Diseño

El diseño es el primer paso en la fase de desarrollo de cualquier producto o sistema de ingeniería. El objetivo del diseño es producir un modelo o representación de una entidad que se va a construir posteriormente (Pressman Roger, 1998). La finalidad del Diseño es tener una idea, al menos visual y genérica de todo el sistema. Un buen análisis de requisitos, restricciones y otros factores que uno considere oportuno considerar puede llevar a un diseño bastante maduro y aproximado de como concebir al software.

2.7.1 Patrones de diseño manejados

- **Patrones GRASP**

Patrón Controlador: Una vez que este recibe la petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL introducida por el usuario. Después de consumada la ejecución, le devuelve al usuario la página creada a partir de la plantilla correspondiente. Este patrón se evidencia en la carpeta Controller, dentro de la cual se encuentran todas las clases controladoras del módulo, las mismas se encargan de implementar las funcionalidades pertenecientes a las interfaces necesarias. Ubicación “SIT/src/SIT/ECONOMICO/RevisionEconomicoBundle/Controller”.

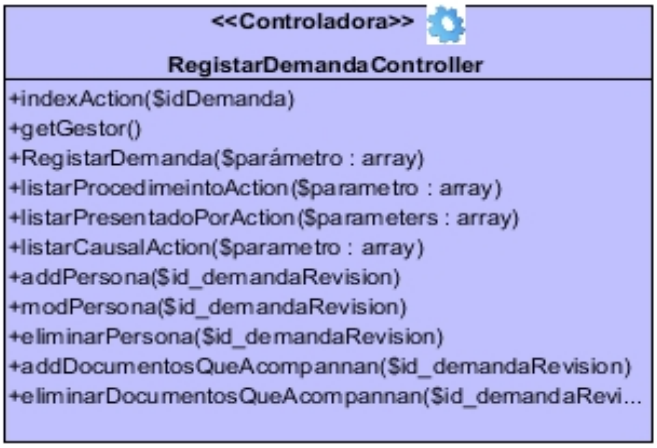


Figura 10 Controladora Registrar Demanda

Patrón Experto: Experto es un patrón para la asignación de responsabilidades; es un principio básico que suele utilizarse 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 les pide. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clases sencillas y más cohesivas que son más fáciles de comprender y mantener. (Saavedra, 2011)

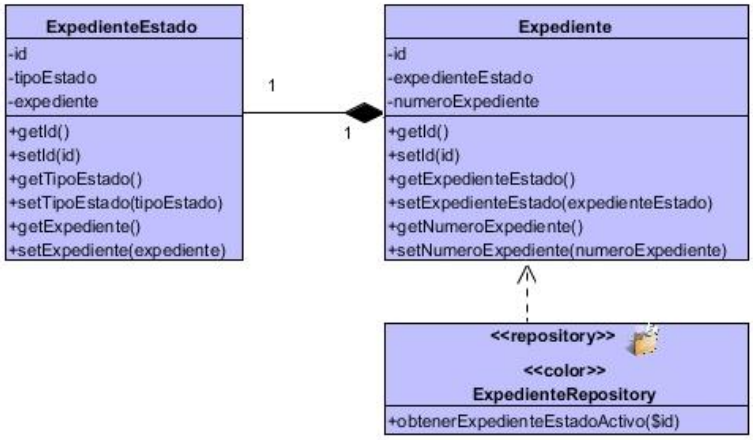


Figura 11 Patrón experto

Patrón Creador: El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Brinda un soporte a un bajo acoplamiento –patrón que será descrito más adelante– lo que supone menos dependencias respecto al mantenimiento y mejores oportunidades de

reutilización. La utilización de este patrón se evidencia en las clases gestoras y en las clases controladoras a la hora de crear los formularios. (Saavedra, 2011)

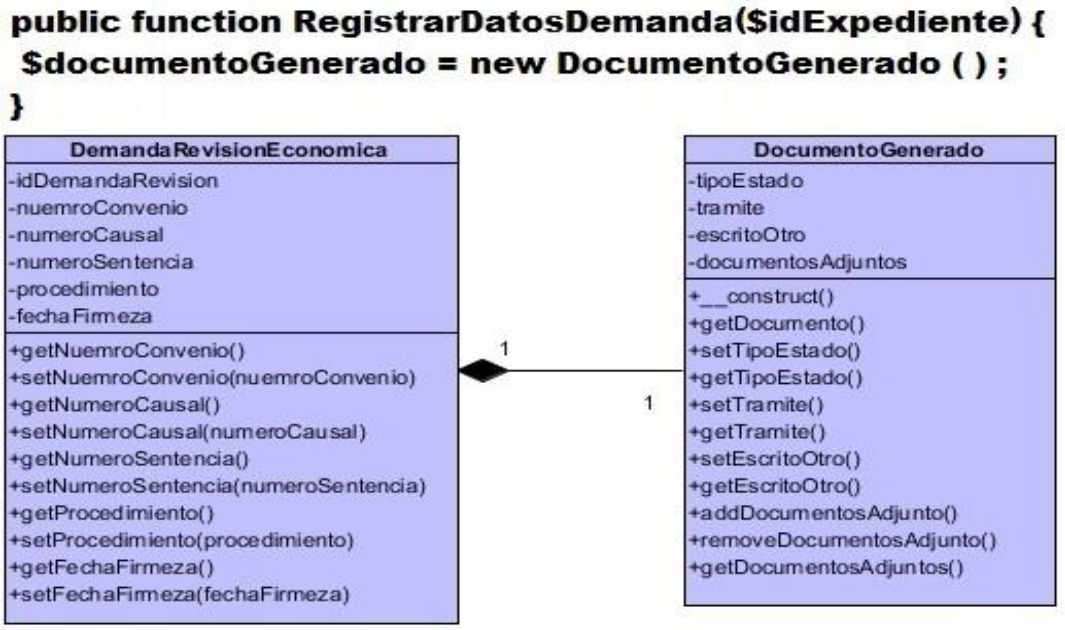


Figura 12 Creación del objeto DocumentoGenerado en la clase RegistarDemandaGtr

Patrón Bajo acoplamiento: Este patrón está estrechamente relacionado con los patrones Experto o Alta Cohesión y plantea la baja dependencia que debe existir entre las clases. Symfony favorece ampliamente el bajo acoplamiento de las clases en el sistema, esto ocurre porque a cada clase se le asignan solamente las responsabilidades necesarias de manera que no dependan en gran medida de otras. Esto favorece la escalabilidad del sistema, la reutilización y el mantenimiento futuro de la aplicación. Se evidencia claramente el uso de este patrón en las entidades, que son las clases más reutilizadas y no presentan ninguna asociación con la vista ni con el controlador.

Patrón Alta cohesión:

Las clases con baja cohesión a menudo representan un alto grado de abstracción o han asumido responsabilidades que deberían haber delegado a otros objetos por lo que se debe asignar una responsabilidad de modo que la cohesión siga siendo alta. El patrón Alta Cohesión es la meta principal que ha de tenerse en cuenta en cada momento en todas las decisiones de diseño. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño. Una clase de alta cohesión posee un número relativamente pequeño, con una importante funcionalidad relacionada y poco trabajo que hacer. Colabora con otros objetos para compartir el esfuerzo si la tarea es grande. Con el uso de este patrón mejoran la claridad y la facilidad con que se entiende el diseño. Se simplifican el mantenimiento y las mejoras en

funcionalidad. A menudo se genera un bajo acoplamiento. La ventaja de una gran funcionalidad soporta una mayor capacidad de reutilización, porque una clase muy cohesiva puede destinarse a un propósito muy específico. (Saavedra, 2011)

- **Patrones GoF**

Patrón Factory Method: Define una interfaz para crear objetos, pero deja que sean las subclases quienes decidan qué clases instanciar; permite que una clase delegue en sus subclases la creación de objetos (Gamma, 1995).En la realización de este módulo queda evidenciado cuando se crean los servicios de las clases Repository en "SIT/src/SIT/vendor/Base/ComunBundle/Repository". Estas clases son las que contienen las consultas a la base de datos. Por tanto cuando la clase gestora necesite realizar una consulta, sencillamente se llama al servicio del Repository donde dicha consulta se ubica y se accede a esta sin la necesidad de crear una instancia de la clase que la contiene.

```
public function eliminarDocumentoAdjunto($idDocumento) {
try {
$docAdjunto = $this->getEm()->getRepository('ComunBundle:AG\DocumentoAdjunto')->find($idDocumento);
$this->getEm()->remove($docAdjunto);
$this->getEm()->flush();
return true;
} catch (\Exception $e) {
    return false;
}
    }
```

Figura 13 Llamada a un servicio del repositorio

Patrón Decorador: Este patrón viene incluido en el marco de trabajo Symfony2. El motor de plantillas Twig posee como característica más poderosa la herencia entre plantillas, lo que permite crear un "esqueleto" de plantilla base que contenga todos los elementos comunes del sitio y defina los bloques que las plantillas descendientes pueden sustituir. Esta plantilla es el archivo base.html.twig, que define un simple documento donde se almacena el código HTML que es común a todas las páginas del sistema, para no tener que repetirlo en cada una de ellas.

```
{# empty Twig template #}

{% extends 'ComunBundle::plantilla_SIT.html.twig' %}

{% block contenido %}
    {{form_errors(formulario)}}
    <...52 lines />

{% endblock %}
```

Figura 14 Página decorada con la plantilla base

Patrón de base dato (Llaves subrogadas): Este patrón posibilita a cada entidad generarle una llave primara la cual es única, en vez de usar un atributo como identificador. Permite que las tablas sean más fáciles de consultar por el identificador y la uniformidad en la creación de las llaves de logra mediante el la generación automática de llaves usando secuencias, evitándose cometer errores.

Patrón de base dato (Árbol simple o DPA[22]): Se evidencia en las estructuras de base dato que por su concepto tiene un mismo comportamiento como es el caso de la estructura del país, la provincia y el municipio tienen un mismo comportamiento regidos por un nivel que forman una estructura jerárquica.

2.7.2 Diagrama de clases del diseño y de secuencia

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea una vista lógica de la información que se maneja en el sistema, los componentes que se encargarán del funcionamiento y la relación entre uno y otro (Jacobson, 1992).

Un diagrama de secuencia muestra una interacción, que representa la secuencia de mensajes entre las instancias de clases, componentes, subsistemas o actores. El tiempo fluye hacia abajo en el diagrama y muestra el flujo de control de un participante a otro. El diagrama muestra instancias y eventos de ejemplo, en lugar de clases y métodos. (Montenegro, 2005)

Se muestra la descripción de los actores perteneciente al caso de uso “Registrar demanda” .Seguidamente de su diagrama de clases del diseño y de secuencia respectivamente.

Actores	Secretaria, Juez, Abogado
Resumen	El caso de uso inicia con la presentación de una demanda, la cual será registrada por la secretaria y dispuesta por el juez asignado.
Complejidad	Alta.
Prioridad	Crítica.
Pre-condiciones	Haber agotado todas las vías para resolver el conflicto existente entre las partes en las instancias inferiores al TSP.
Post-condiciones	Se registra una nueva demanda de revisión económica.

Tabla 7 Descripción de clases del diseño del caso de uso “Registrar Demanda”

[22] División política administrativa

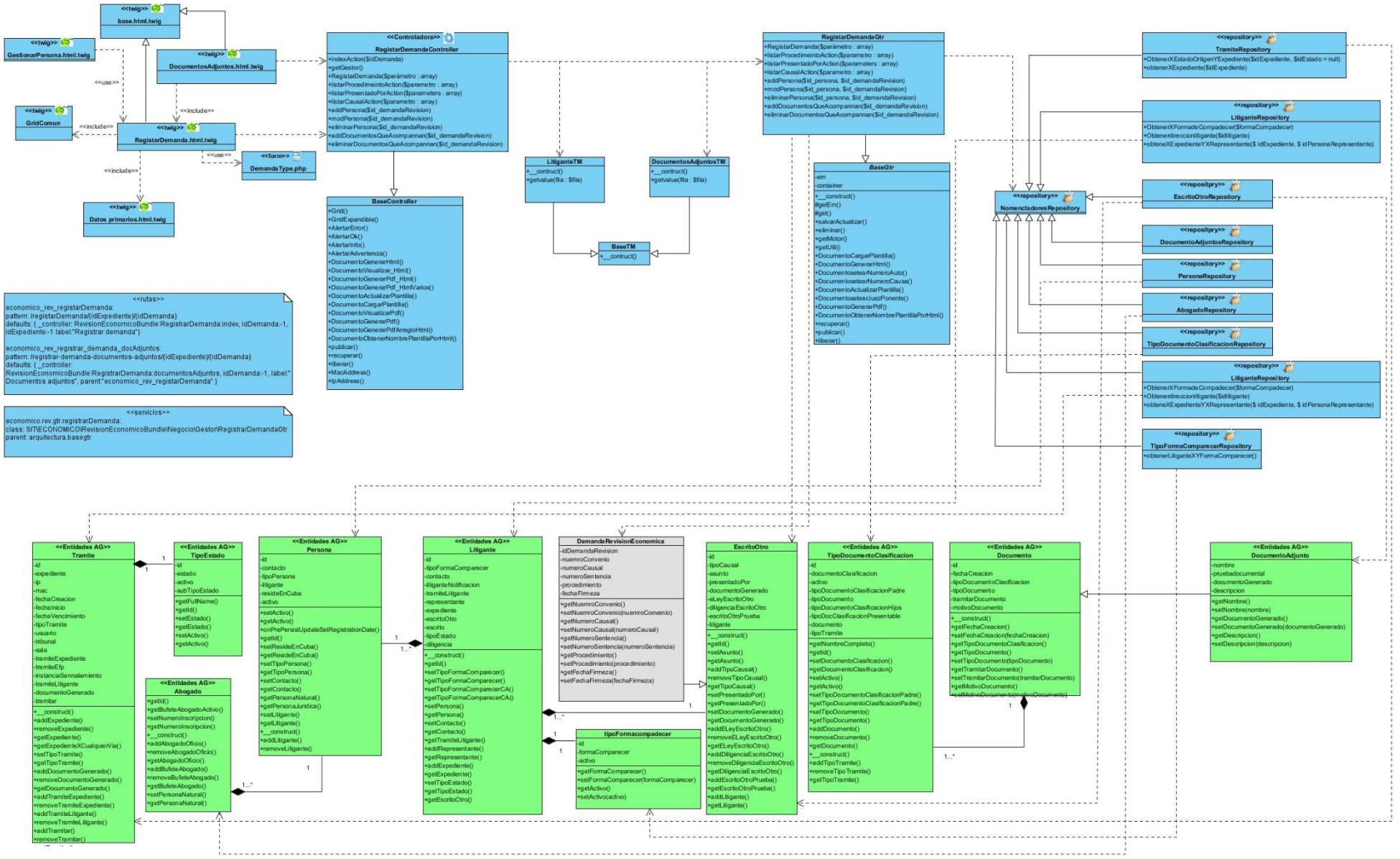


Figura 15 Diagrama de clases del diseño del caso de uso "Registrar Demanda"

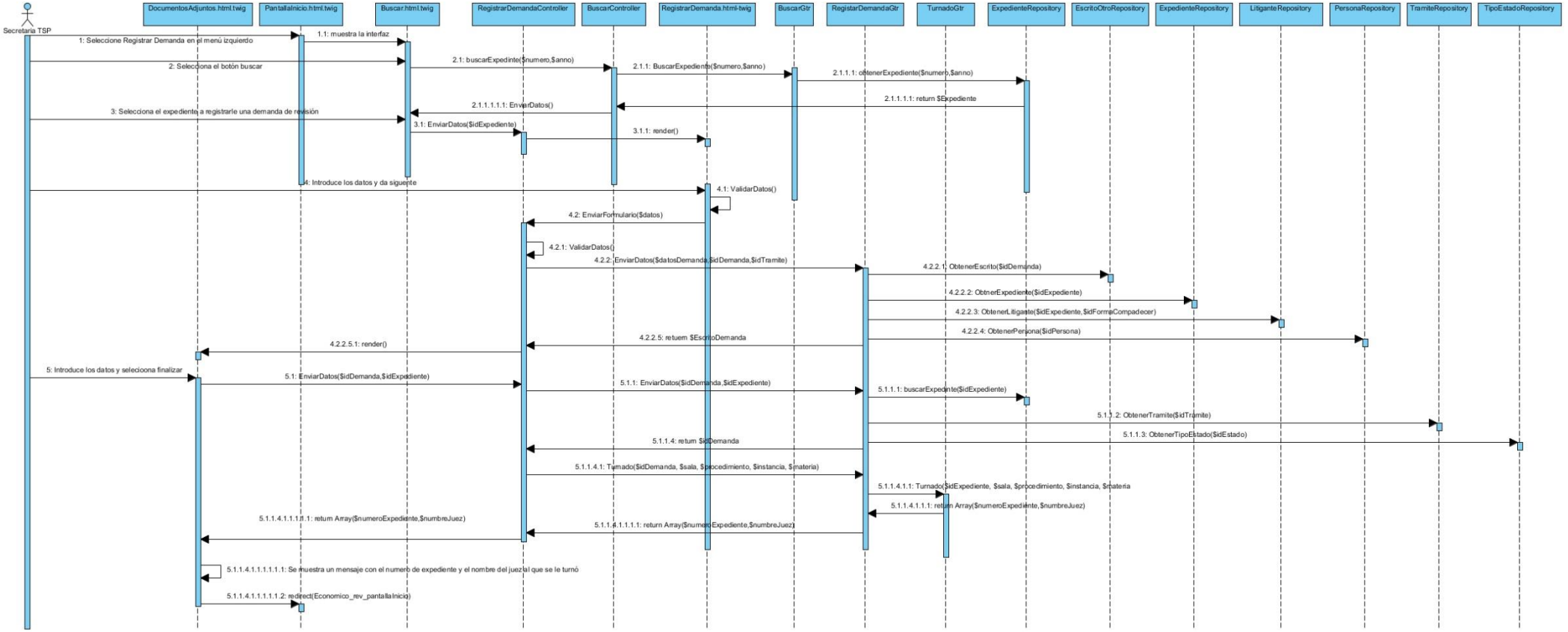


Figura 16 Diagrama de secuencia del diseño del uso "Registrar Demanda"

2.7.3 Modelo de datos

Un modelo de datos es el conjunto de conceptos, reglas y convenciones que permiten describir, a distintos niveles de abstracción, la estructura de una base de datos, la cual se denomina esquema, o con otras palabras, permiten construir una representación organizada de un sistema real. (Castro, 1999)

En la figura 17 se muestra una pequeña porción del modelo de datos del subsistema Económico acomodado al proceso de Revisión, señalando la tabla Demanda_revision_economica, la misma fue mapeada e incluida al modelo de datos del SITPC. Para ver el diagrama completo consultar **"*Anexo 2 y 3*"**.

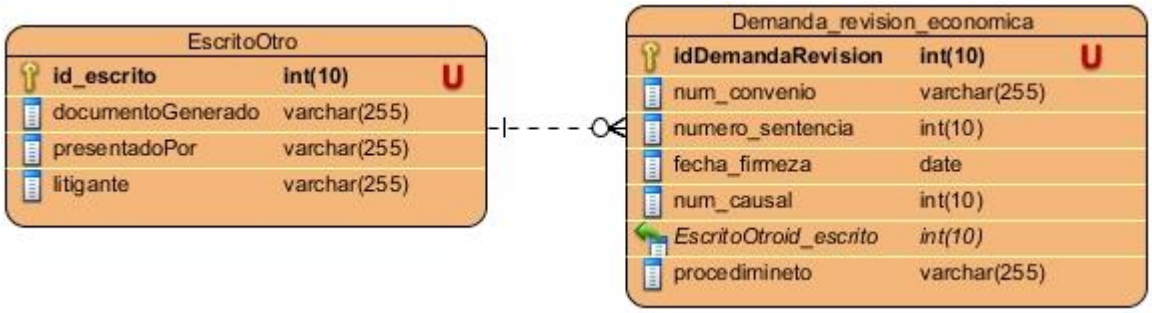


Figura 17 Nueva tabla incorporada al modelo de datos existente

2.7.4 Diagrama de despliegue

Un diagrama de despliegue muestra las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Estos diagramas describen la arquitectura física del sistema durante la ejecución en términos de procesadores, dispositivos y componentes de software (Alva, 2012).

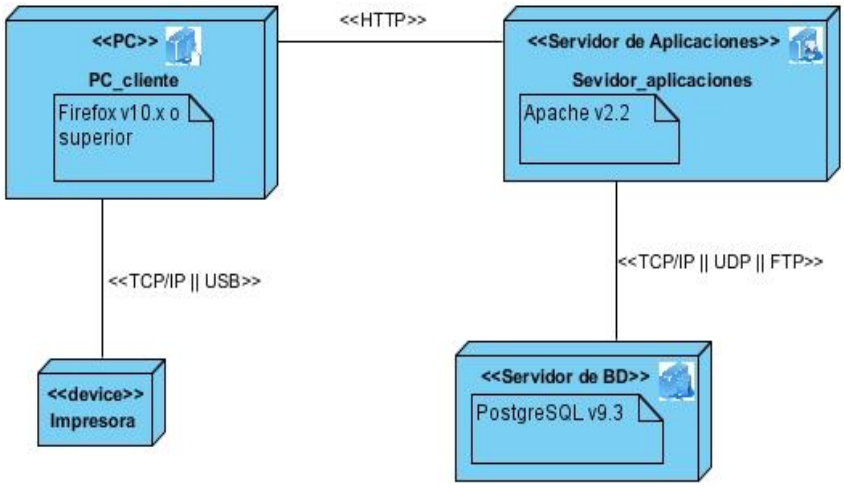


Figura 18 Diagrama de despliegue

- ✓ **Nodo PC Cliente:** Computadora con el sistema operativo Linux y el navegador Mozilla Firefox, mediante el cual los usuarios podrán acceder al sistema y hacer uso del mismo.
- ✓ **Nodo Impresora**: Dispositivo de impresión preciso para llevar a formato duro los diferentes documentos generados en la institución.
- ✓ **Nodo Servidor de aplicaciones**: Servidor que contendrá todo lo referente al sistema: se incluirán los archivos necesarios para que el usuario pueda tener acceso; manejará las informaciones específicas de los registros, así como sus clases, almacenará la configuración general del sistema, las clases y librerías externas además de los plugins de instalación de la aplicación.
- ✓ **Nodo Servidor de bases de datos (Tribunal Supremo):** Servidor en que se encontrará la base de datos que contendrá toda la información del Tribunal Supremo Popular. Este servidor se sincronizará con el Centro de Datos para enviar la información del día en horario no laboral.

2.8 Modelo de implementación

El Modelo de Implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. (Hernández, 2013)

2.8.1 Diagrama de Componente

En las figuras siguientes se muestra el diagrama de componente para la solución y los elementos que la conforman distribuidos según el patrón MVC:

- ✓ **Capa Modelo**: Está constituida por las clases gestoras ubicadas en el paquete RevisionEconomicoBundle /Negocio/Gestor, comunicándose con los repositorios, posibilitando el acceso a la información recogida en la base de datos.
- ✓ **Capa Vista**: Contiene el paquete de las vistas que se muestran al usuario final y se encuentran ubicadas dentro del proyecto en el paquete RevisionEconomicoBundle/Resources/views/Paginas.
- ✓ **Capa de Datos**: Engloba todos los datos persistentes de la base de datos del sistema.
- ✓ **Capa Controladora**: Está formado por el controlador frontal y el paquete de clases controladoras del bundle RevisionEconomicoBundle donde se encuentran las acciones de los casos de uso.

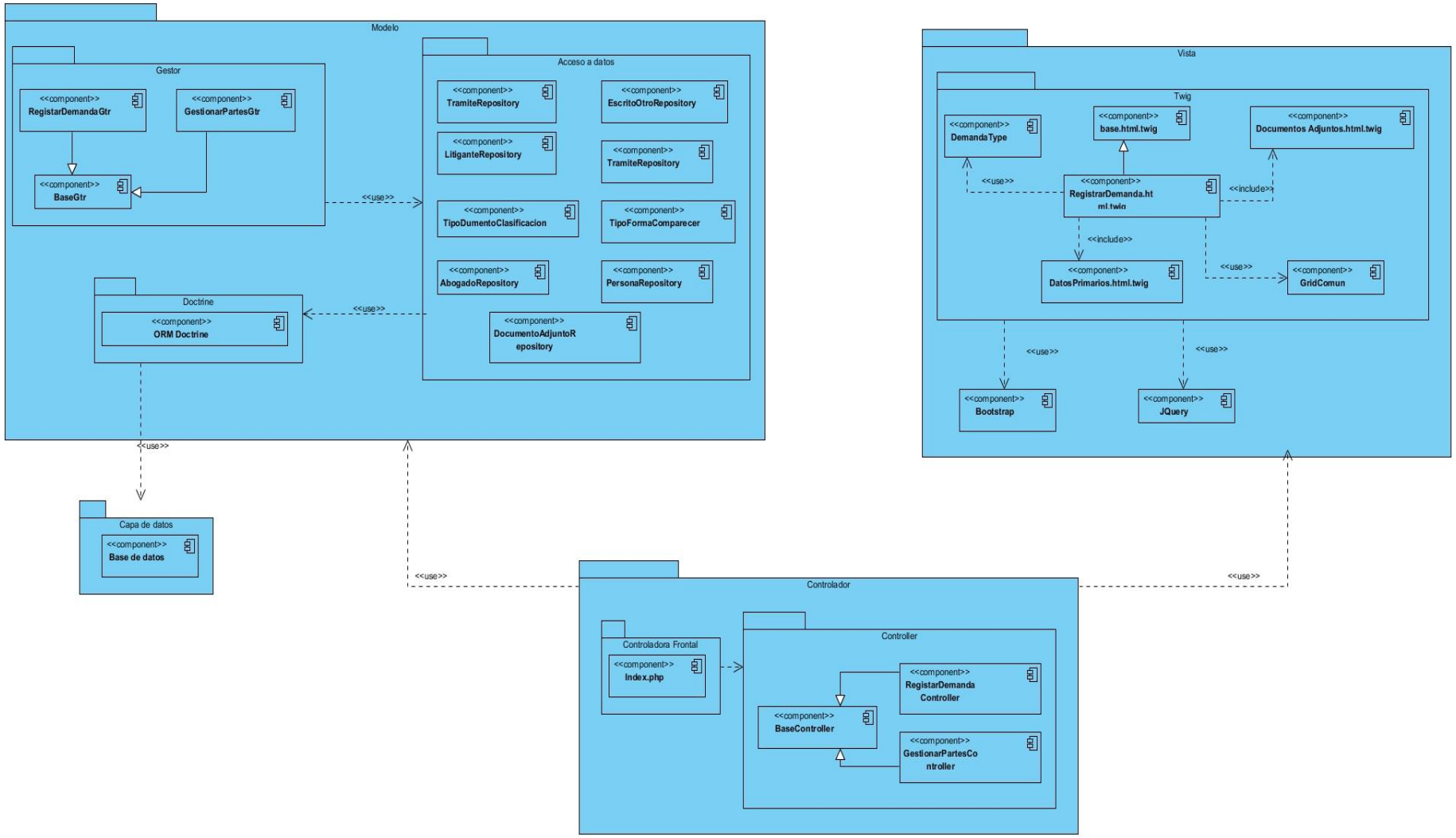


Figura 19 Diagrama de componentes

2.9 Estándares de codificación

Con el propósito de tener una mayor organización y entendimiento de los códigos se trató de respetar los estándares de codificación definidos por el proyecto SITPC evidenciándose mediante las siguientes pautas:

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

```
/**
 * version @1.16
 * Description of RegistrarDemanda
 * @modificado: 9 de junio 2015
 * @author Yadieski
 */
```

Figura 20 Cabecera del archivo

- **Comentarios en las funciones:** Todas las funciones antes de su declaración tienen un comentario explicando lo que hacen, como se puede apreciar en la figura 21. De este modo ningún programador tendrá que analizar el código de una función para conocer su utilidad.

```
//adiciona un litigante al prceso en la vista RegistarDemanda.html.twig
    public function adicionarPartesAction($tipoLitigante, $idPersona, $idPersonaFiscal) {

        $idDemanda = $this->recuperar('idDemanda');
        $var = $this->getGestor()->insertarLitigante($idDemanda, $tipoLitigante,
                $idPersona, $idPersonaFiscal);

        return new Response($var);
    }
```

Figura 21 Comentarios en las funciones

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

Para ver la información completa de los estándares de codificación definidos, ver el documento “CEGEL-SITPC Estándares de codificación” del proyecto SITPC.

Conclusiones

Una vez ejecutadas las disciplinas de Modelado del Negocio, Requisitos, Diseño e Implementación se puede arribar a las siguientes conclusiones:

- ✓ El modelado y descripción de los procesos de negocio usando la notación BPMN, contribuyó a obtener una visión global del flujo de actividades del procedimiento de revisión, así como a identificar los requisitos funcionales del sistema.
- ✓ El modelo de diseño obtenido a partir de las especificaciones de los casos de uso, permitió poner en práctica la arquitectura de software definida, los patrones de diseño necesarios, así como para lograr un lenguaje entendible para la implementación.
- ✓ El uso de buenas prácticas, como los estándares de codificación y la estructura arquitectónica propuesta por el marco de trabajo Symfony 2, posibilitó la generación de un código legible y organizado, así como la obtención del modelo de implementación.
- ✓ Cada una de las disciplinas de la metodología RUP ejecutadas, junto a las prácticas del proceso de mejora CMMI, contribuyeron a la obtención de un producto de software listo para evaluar si el resultado se corresponde con las necesidades del TSP.

CAPÍTULO 3: Validación de resultados

3.1 Introducción

En este capítulo se exponen los resultados de las pruebas y las verificaciones de calidad aplicadas al diseño y a la implementación del sistema para comprobar la calidad y fiabilidad de la solución obtenida, asegurando su correcto funcionamiento mediante la utilización de métricas que validaran nuestro resultado.

Métricas para la calidad del software

A la aplicación de técnicas basadas en la medida de los procesos de desarrollo del software, para producir una información de gestión significativa en la mejora de los procesos y sus productos, se denominan métrica de software. “Un Método y una escala cuantitativos que pueden ser usados para determinar el valor que toma cierta característica en un producto de software concreto”. “Una función que toma como entrada cierta información del software que se está midiendo, y que devuelve como salida un valor numérico sencillo, el cual es interpretada, como el grado en que el producto de software posee un atributo dado que afecta a su calidad”. (Vega Lebrún Carlos, 2008)

3.2 Métrica de la calidad de la especificación

Se utiliza esta métrica para valorar la calidad del modelo de análisis y la correspondiente especificación de los requisitos teniendo en cuenta las características especificidad (ausencia de ambigüedad) y compleción, con el objetivo de demostrar que su definición satisface al cliente y evitar que los errores trasciendan a fases posteriores. Una vez identificados los requisitos se realizó una prueba para determinar su especificidad basada en la consistencia de la interpretación del equipo de revisores, constituido por el analista principal del proyecto, un analista con vasta experiencia en el campo y jueces de tribunal Supremo.

Para realizar este proceso primeramente se determina el total de requisitos que se obtiene de la suma de los requisitos no funcionales y los requisitos funcionales identificados.

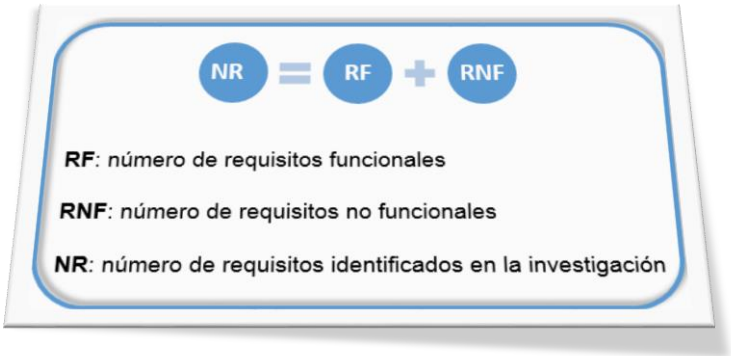


Figura 22 Fórmula para determinar la cantidad de requisitos en una especificación

Se aplica la fórmula a nuestra especificación obteniéndose el siguente resultado :

NR = RF + RNF NR = 40 + 45

NR = 95

Para determinar la especificidad se divide el total de requisitos con igual interpretación por parte de los revisores entre la cantidad total de requisitos (NR).

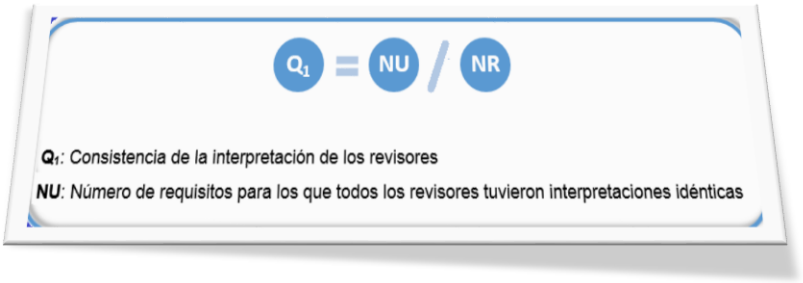


Figura 23 Fórmula para calcular la consistencia de la interpretación de los revisores

Al aplicar la fórmula en los datos generados se obtiene que :

Q_1 = NU / NR Q_1 = 77 / 95

Q_1 = 0.81

Cuanto más cerca de 1 esté el valor de Q_1 menor será la ambigüedad de la especificación (Villamizar, 2012). Por ende, teniendo en cuenta que 1 - Q_1 es igual a 0.19, se puedo verificar que el grado de ambigüedad es bajo para nuestra especificación.

3.3 Verificación de requisitos mediante prototipos

Los prototipos son un método de verificación, que simula el sistema que se quiere implementar. En el caso de los requisitos, se utilizan para comprobar la corrección y completitud de la especificación de requisitos. Para validarlos mediante prototipos no funcionales, se realizaron las siguientes tareas:

- ✓ Seleccionar quién evaluará el prototipo: Participan en la evaluación: la analista principal del proyecto, el analista del subsistema Económico y un juez del Tribunal Supremo.
- ✓ Desarrollar escenarios de validación: Se crean una serie de escenarios, los cuales son llevados a cabo por los compradores utilizando los prototipos.
- ✓ Ejecutar los escenarios: Se ejecutaron los escenarios diseñados obteniéndose como resultado una equivalencia entre el producto final y el prototipo creado.

3.4 Métricas aplicadas al diseño del software

Para validar las clases del diseño se utilizaron las métricas: Tamaño operacional de clases (TOC) y Relación entre clases (RC). Estas métricas fueron diseñadas para validar los siguientes atributos de calidad: Los atributos que se abarcan son (Pressman, 2005):

- ➢ Responsabilidad. Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto de la problemática propuesta.

- Complejidad de implementación. Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- Reutilización. Consiste en el grado de reutilización presente en una clase o estructura de clase dentro de un diseño de software.
- Acoplamiento. Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.
- Complejidad de mantenimiento. Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta pero fuertemente en los costes y la planificación del proyecto.
- Cantidad de pruebas. Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (Componente, módulo, clase, conjunto de clases, etc.) diseñado.

Tamaño Operacional de la Clase (TOC)

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

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

Figura 24 Modo en que los atributos de calidad afectan el TOC

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

Se aplicó a un total de 16 clases, las cuales fueron escogidas por ser las de mayor complejidad obteniéndose como resultado:

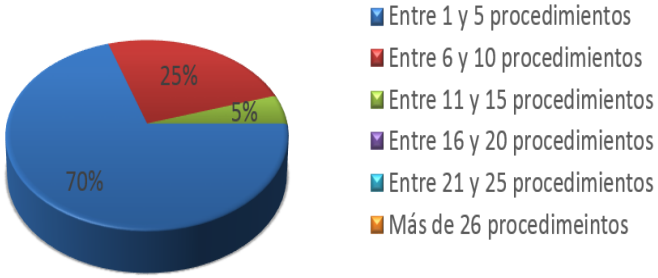


Figura 25 Resultado de la métrica TOC

Figura 26 Representación de los resultados en la evaluación de la métrica TOC para los atributos: responsabilidad, reutilización y complejidad

Después de hacer un análisis de los resultados obtenidos se concluye que la mayoría de las clases incluidas poseen de 1 a 5 procedimientos en su composición representando el 75% del total de clases analizadas. Por tanto, el resultado influye positivamente en los demás atributos de calidad, donde el 80% de las clases posee una responsabilidad baja, además se observa que el 80% de clases analizadas tiene una complejidad baja lo que permite realizar una explotación alta del atributo de reutilización. Concluyéndose que el diseño propuesto para la evaluación del software fomenta la Reutilización y reduce en menor grado la Responsabilidad y la Complejidad de implementación.

Relación entre clases (RC)

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

Figura 27 Modo en que los atributos de calidad afectan a la métrica RC

Resultado de la aplicación de la métrica Relación entre clases (RC)

Se aplicó a un total de 16 clases, las cuales fueron escogidas por ser las de mayor complejidad obteniéndose como resultado:

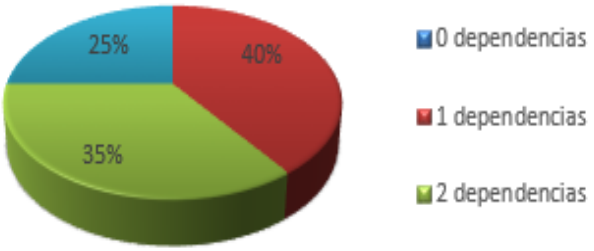


Figura 28 Resultado de la métrica RC

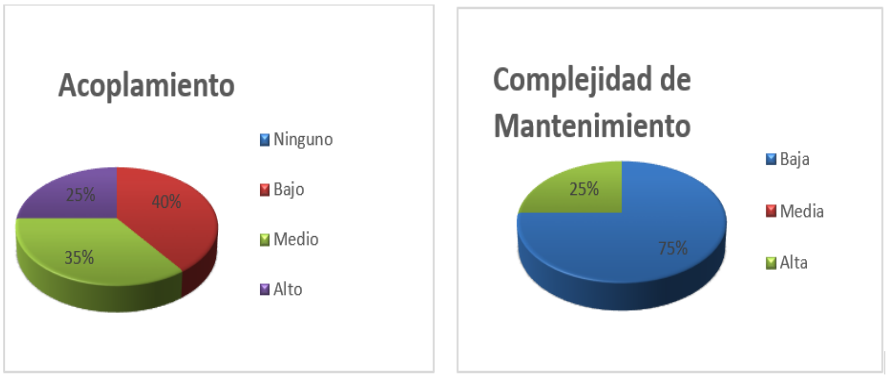


Figura 29 Representación de los resultados en la evaluación de la métrica RC para los atributos: Acoplamiento y Complejidad de Mantenimiento

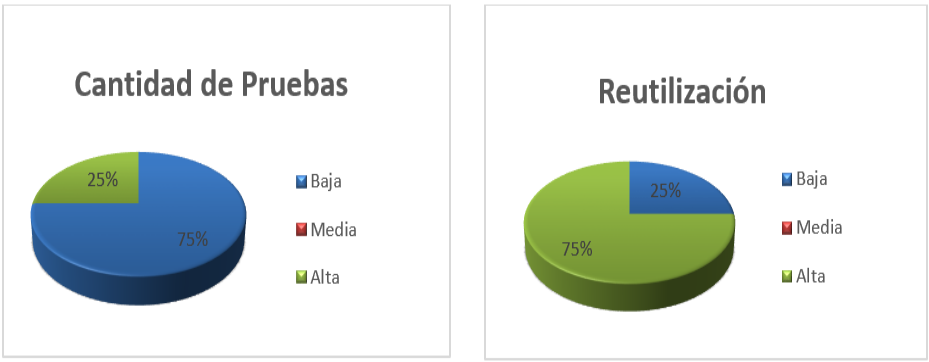


Figura 30 Representación de los resultados en la evaluación de la métrica RC para los atributos: Cantidad de Pruebas y Reutilización

Después de hacer un análisis de los resultados obtenidos a través de la métrica RC aplicada a las clases que componen el bundle RevisionEconomicoBundle se obtiene que el 75% de las clases analizadas poseen un bajo acoplamiento lo cual influye positivamente pues al ser afectada una clase este cambio trascenderá de manera mínima en las demás. El porciento de reutilización de clases es alto lo cual fomenta y potencia el uso de las mismas en otras clases, en caso de ser necesario. Además la complejidad de mantenimiento es baja, posibilitando que la optimización del código sea de manera sencilla. Por último, el atributo relacionado con la cantidad de pruebas es bajo lo cual señala que a las clases se les puede realizar un bajo número de pruebas de poca complejidad. Se concluye que el 75% de las clases poseen menos de 2 dependencias respecto a otras. Los atributos de calidad se encuentran en un nivel satisfactorio; ya que el grado de acoplamiento de las clases es bajo, la Complejidad de Mantenimiento, la Cantidad de Pruebas y la Reutilización se comportan favorablemente para un 75% de las clases.

3.5 Pruebas de software

El único instrumento adecuado para determinar el status de la calidad de un producto software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requisitos. (Jústiz Núñez, 2014)

Técnicas de Pruebas

Dentro de las pruebas de caja blanca se encuentra la técnica **Prueba del Camino Básico** que permite obtener una medida de la complejidad lógica de un diseño y usar la misma como guía para la definición de un conjunto de caminos básicos. (Ruiz, 2011). En las pruebas de caja negra se localiza la técnica **Partición de equivalencia** que divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software. (Pressman, 2007)

Para comprobar el comportamiento del sistema se aplican las pruebas de caja blanca y caja negra con el propósito de corregir fallos de implementación y calidad de la solución propuesta. En correspondencia con el estudio antes realizado se utilizan los siguientes métodos de pruebas.

Pruebas de caja blanca

Para valorar la calidad con la que se llevó a cabo la implementación propuesta el método de caja blanca mediante la técnica del camino básico fue el seleccionado debido a que permite obtener una medida de la complejidad lógica de un diseño y usar la misma como guía para la definición de un conjunto de caminos básicos. A continuación se muestra el método obternerNombreLitigante perteneciente a la clase GestionarPartesGtr que se encarga de gestionar los litigantes cuando se realiza una demanda de revisión económica.

```
//Devuelve el nombre del Litigante
public function obtenerNombreLitigante($idExpediente, $idFormaComparecer) {
    $expediente = $this->getEm()->find('ComunBundle:AG\Expediente',$idExpediente);//1

    //Se obtienen los litigantes
    $litigantes = $expediente->getLitigante(); //1

    for ($index = 0; $index < count($litigantes); $index++) { //2
        $persona = $litigantes->get($index)->getPersona();//3

        //Buscar el nombre de la persona
        $nombrePersona = "";//3
        if ($persona instanceof PersonaNatural) {//4
            $nombrePersona = $persona->getNombreCompleto();//5
        } else {//6
            $nombrePersona = $persona->getNombrepj();//7
        }

        //Buscar si es demandado o demandante
        if ($litigantes->get($index)->getTipoFormaComparecer()->getId() == $idFormaComparecer) {//8
            return $nombrePersona;//9
        }
    }
    return "";//10
}
```

Figura 31 Método obtenerNombreLitigante ( )

Como inicio de la aplicación del método camino básico se obtiene el grafo correspondiente al método que se analiza:

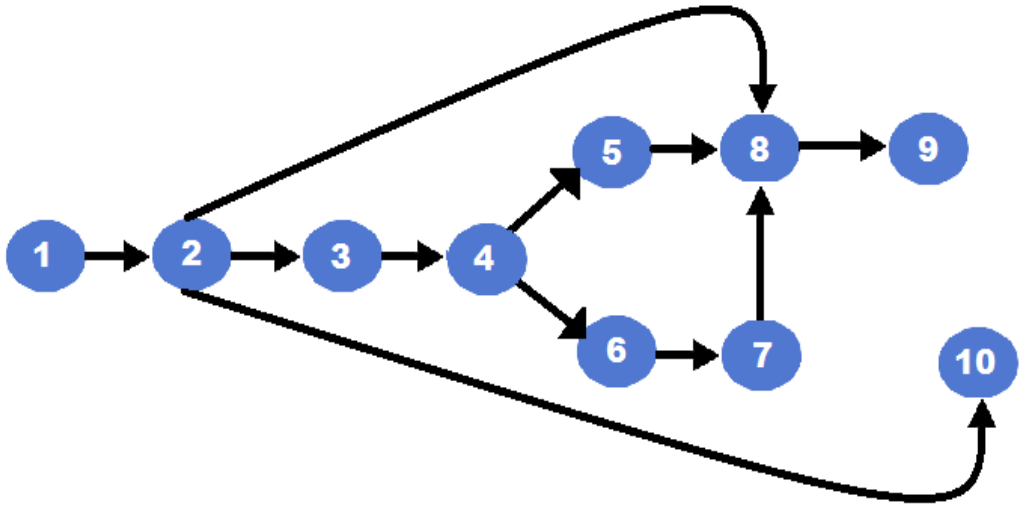


Figura 32 Grafo del flujo correspondiente al método obtenerNombreLitigante ( )

Una vez construido el grafo de flujo asociado al procedimiento anterior se determina la complejidad ciclomática, el cálculo es necesario efectuarlo mediante tres vías o fórmulas de manera tal que quede justificado el resultado, siendo el mismo en cada caso:

1. V (G) = (A – N) + 2

Resultado: V (G) = (11– 10) +2 = 3

2. V (G) = P + 1

Resultado: V (G) =2+1=3

3. V (G) = R

Resultado: V (G) =3

A: cantidad total de aristas del grafo.

N: cantidad total de nodos del grafo.

P: cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

R: cantidad de áreas que conforman el grafo de flujo.

El cálculo efectuado mediante las fórmulas antes presentadas muestra una complejidad ciclomática de valor 3, de manera que existen tres posibles caminos para la ejecución de la funcionalidad, este valor representa el número mínimo de casos de pruebas para el procedimiento tratado.

Camino básico #1: 1 – 2 – 3 – 4 – 5 – 8 – 9.

Camino básico #2: 1 – 2 – 3 – 4 – 6 – 7 – 8 – 9.

Camino básico #2: 1 – 2 – 10.

A continuación se describe el caso de prueba para el camino básico determinado en el grafo de flujo:

Camino básico #1: 1–2–3–4–5–8–9	
Descripción	Se gestionan las partes en una demanda de revisión económica.
Condición de ejecución	Realizar la búsqueda de un litigante asociado a la demanda que se está formulando.
Entradas	Se le asocia un número de expediente y el id de una forma de compadecer.
Resultados esperados:	Obtener el nombre para saber si es demandante o demandado en la nueva demanda de revisión que se registra.

Tabla 8 Descripción del camino básico #1

Luego de aplicar los casos de prueba correspondientes a cada camino básico identificado, se comprobó el correcto funcionamiento del procedimiento obtenerNombreLitigante ( ).

Pruebas de caja negra

Para la solución propuesta se realizaron pruebas funcionales utilizando las pruebas de caja negra mediante el método de partición de equivalencia que permite examinar los valores válidos e inválidos de las entradas existentes en el software. Se aplicó sobre la interfaz que responde al requisito funcional Registrar Demanda. Para ello se definieron variables de equivalencia que representan un conjunto de estados válidos (representan entradas válidas al programa) y no válidos (representan valores de entradas erróneas) para las condiciones de entradas del sistema.

Condiciones de ejecución

- Se debe identificar y autenticar ante el sistema, además debe tener los permisos para ejecutar esta acción.
- Se debe seleccionar la opción del menú: Registrar demanda

En la siguiente tabla se muestra el diseño del caso de prueba para el caso de uso del paquete Presentación de la demanda donde se muestran los escenarios que lo componen.

No de variable	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Expediente	Campo de texto	Si	Se inserta el número del expediente para filtrar una búsqueda.
2	Año	Campo de texto	Si	Se selecciona el año para filtrar una búsqueda.

3	Número de convenio jurídico	Campo de texto	no	Numero identificativo del contrato entre las partes y su representante
4	Número de causal	Campo de texto	no	Causa por la que se registra la revisión.
5	Presentado por	Lista desplegable	no	Representante de la parte demandante

Tabla 9 Descripción de las variables

Escenario	Descripción	V1	V2	Respuesta del sistema	Flujo central
Esc 1: Buscar un expediente.	Se seleccionan las opciones de búsquedas, por año o por número de expediente.	V	V	Se lista expediente correspondiente a los datos buscado	1. Seleccionar Registrar demanda en el menú izquierdo. 2. Seleccionar el botón Buscar.
		Vacío	V	Lista a los expediente de ese año	
		vacío	vacío	Se listan todos los expedientes existentes	
Esc 2: Cancelar la búsqueda del expediente.	Se seleccionan las opciones de búsquedas, por año o por número de expediente.	V	V	Cierra la interfaz y regresa a la pantalla inicial.	1. Seleccionar Registrar demanda en el menú izquierdo. 2. Seleccionar el botón Buscar.
		Vacío	V	Cierra la interfaz y regresa a la pantalla inicial.	
		vacío	vacío	Cierra la interfaz y regresa a la pantalla inicial.	

Escenario	Descripción	V3	V4	V5	Respuesta del sistema	Flujo central
Esc 4: Registrar una demanda con datos correctos.	Se introducen los datos en la nueva demanda de revisión que será registrada en la base de datos.	V	V	V	1. Se muestra un mensaje con el número de expediente y el juez al que se le turno. 2. Vuelve a la pantalla inicial.	1. Seleccionar Registrar demanda en el menú izquierdo. 2. Seleccionar el botón Buscar. 3. Se selecciona el botón siguiente. 4. Se selecciona el botón Finalizar una vez formulada la demanda. 5. Se selecciona el botón aceptar.
Esc 4: Registrar una demanda con datos incorrectos.	Se introducen los datos en la nueva demanda de revisión que será registrada en la base de datos.	+ Caracter extraño letras Letras V	V Caracter extraño letras V letras	- Caracter extraño letras Letras V	El sistema muestra un mensaje de alerta mostrando que los datos introducidos no son válidos.	1. Seleccionar Registrar demanda en el menú izquierdo. 2. Seleccionar el botón Buscar. 3. Se selecciona el botón siguiente. 4. Se selecciona el botón Finalizar una vez formulada la demanda. 5. Se selecciona el botón aceptar.

Esc 5: Cancelar una demanda de revisión.	Se introducen los datos en la nueva demanda de revisión que será registrada en la base de datos.	V	V	V	El sistema muestra un mensaje de confirmación	1. Seleccionar Registrar Demanda en el menú izquierdo. 2. Seleccionar el botón Buscar. 3. Se selecciona el botón Cancelar.

Tabla 10 Caso de prueba de paquete Presentación de la demanda

Durante la ejecución de las pruebas se realizaron tres iteraciones a la aplicación, detectándose no conformidades (NC) como faltas de ortográficas, errores en las validaciones de la interfaz del usuario y otras relacionadas con el negocio que no cumplían con el prototipo definido por el proyecto como parte de la estandarización definida para el mismo. Seguidamente se muestra el gráfico de los resultados obtenidos:

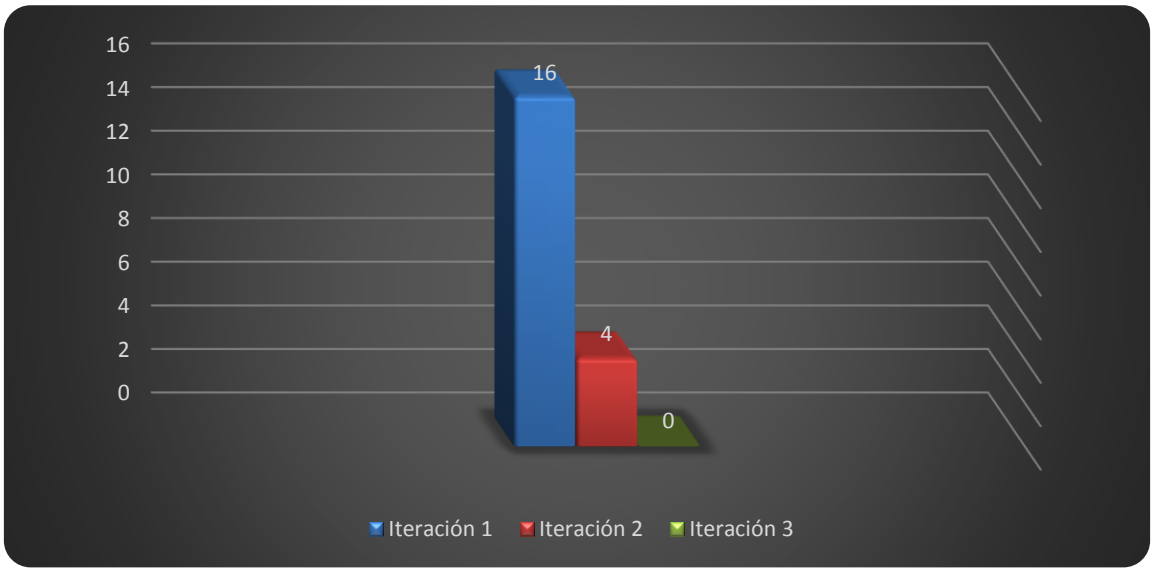


Figura 33 Resultados de las pruebas de caja negra a los paquetes de caso de uso que componen la aplicación

Resultados de las pruebas aplicadas

Luego de aplicar el método para medir la calidad de las funcionalidades implementadas, se lograron resultados satisfactorios desde el punto de vista interno y funcional, atendiendo al correcto comportamiento del sistema ante las disímiles situaciones. Las no conformidades detectadas fueron debidamente atendidas y corregidas durante el transcurso de las iteraciones con el fin de lograr el correcto funcionamiento de la solución desarrollada.

3.6 Validación de las variables

Se realizó una encuesta a los principales analistas del proyecto y a representantes del cliente, para este caso un abogado del bufete colectivo de la provincia Camagüey. A partir del análisis de las respuestas y de los elementos técnicos propios de la solución se obtiene la siguiente tabla:

Variables	Antes	Después
Integridad de la información	Los expedientes archivados solo pueden ser accedidos por la secretaria. El presidente del tribunal debe emitir una resolución para tener acceso a los archivos si no es la secretaria. Los expedientes son almacenados en forjas que suelen ser muy grandes y tienden a deteriorarse con el tiempo. Los datos de cada expediente a la hora de ser modificados provocan las tachaduras y borrones. Los expedientes se encuentran organizados por secciones en los estantes donde se ubican, haciendo muy difícil su búsqueda.	La seguridad del sistema es manejada por Acaxia, sistema que valida los permisos de ejecución de una ruta o servicio al cual se quiere acceder, garantizando: Un mecanismo de autenticación personalizada para todos los usuarios que harán uso de la aplicación mediante dirección o rango de direcciones IP autorizadas a hacer uso del sistema. Se contará con un histórico de las contraseñas que no permitirá que el usuario establezca alguna de sus últimas 24 usadas. Se establece el tiempo de expiración de la contraseña, por defecto es de 90 días. El empleo de protocolos https (Protocolo seguro de transferencia de hipertexto) para la transmisión de las credenciales de autenticación entre los clientes y el servidor.
		Que cada usuario para consultas a la base de datos, cuente solo con los privilegios mínimos e indispensables. Que exista un correcto mecanismo del manejo de sesiones.

		Con la creación de una base de datos para el almacenamiento de los expedientes y el manejo de los datos se facilita su búsqueda y evita la pérdida o deterioro de la información. Los datos se manejan de forma digital permitiendo que su modificación no provoque, las tachaduras y los borrones. El sistema permite imprimir a formato duro los expedientes en caso de ser necesario.
Tiempo de ejecución	En la tramitación de un proceso no existe una forma de verificar el cumplimiento de los términos legales de tiempo. El proceso de revisión llega a ser tan largo que cuando termina muchas veces a las partes ya no les interesa.	El sistema obliga a que se cumplan los términos legales de tiempo pues al personal que participaba en el proceso en cuanto al estado del trámite que está en ejecución no podrá modificar datos una vez cumplido el límite de tiempo establecido.

A partir del análisis realizado se evidencia una contribución en el manejo de la integridad de la información y el tiempo de ejecución durante todo el proceso de revisión. Asegurando que la investigación cumplió con las metas definidas obteniéndose un producto funcional, acorde a las necesidades existentes en los Tribunales Populares Cubanos.

Conclusiones parciales

Una vez culminada la construcción del producto de software con la realización de pruebas al mismo, se puede arribar a las siguientes conclusiones:

- El uso de métricas, así como las pruebas de caja blanca y caja negra, permitieron evaluar la calidad del producto obtenido, corregir errores y asegurar que el resultado satisface los requisitos establecidos.
- La evaluación del comportamiento de las variables dependientes de la investigación permitió corroborar que la solución contribuye a la integridad de la información y disminuye el tiempo de ejecución del procedimiento de revisión.

CONCLUSIONES GENERALES

La realización del presente trabajo y los resultados obtenidos permitieron arribar a las siguientes conclusiones.

- ✓ La elaboración del marco teórico de la investigación, permitió identificar que no existe un sistema informático que pueda ser utilizado para la ejecución del procedimiento Revisión en los Tribunales Populares Cubanos. Se determina por ello la necesidad de desarrollar un nuevo sistema.
- ✓ La fundamentación de la metodología, las herramientas, tecnologías y lenguajes contribuyó a la obtención de los elementos teóricos necesarios para el desarrollo de un subsistema para la gestión del procedimiento “Revisión”.
- ✓ La obtención del modelado del negocio permitió comprender la estructura, dinámica y los problemas existentes en la organización, así como identificar las acciones a informatizar. Asimismo la creación del modelo de diseño e implementación permitió la obtención de una propuesta de solución que respondió a las necesidades de desarrollo del SITPC.
- ✓ La aplicación de métricas y pruebas de software permitió la obtención de una solución funcional que cumple con la calidad requerida y potencia la reutilización y la mantenibilidad del software.
- ✓ Las encuestas aplicadas a especialistas del proyecto y los elementos técnicos de la solución permitieron corroborar el cumplimiento del objetivo del presente trabajo de diploma.

RECOMENDACIONES

Una vez terminado el desarrollo de los paquetes del subsistema Económico para el procedimiento Revisión se recomienda:

- ✓ Continuar con el análisis, diseño y posterior implementación de las restantes instancias de los Tribunales Populares Cubanos para la Materia Económica.

BIBLIOGRAFÍA

1. Tribunal Supremo Popular República de Cuba. [En línea] [Citado el: 1 de febrero de 2015.] www.tsp.cu.

2. Barchini, Graciela Elisa. *Métodos "I + D" de la Informática.* Universidad Nacional de Santiago del Estero, Avenida Belgrano (S) 1912 : s.n.

3. El wiki USCODERECHO. [En línea] [Citado el: 16 de 5 de 2015.] http://uscoderecho.wikispaces.com/INFORMATICA+JURIDICA+DE+GESTION.

4. Ministros, Grupo Especial encargado de dar cumplimiento a las Recomendaciones de las Reuniones de. [En línea] [Citado el: 1 de febrero de 2015.] https://www.google.com.cu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0CCUQFjAB&url=https%3A%2F%2Fwww.oas.org%2Fconsejo%2Fsp%2FCAJP%2Fdocs%2Fcp09044s04.doc&ei=wejXVK73N8qMyATQlICQDQ&usg=AFQjCNEfWA9oztD-jE5xLJ36GW5DnZu_iw&bvm=bv.85464276,d.aWw&cad=rja.

5. Sistema Automatizado de Gestión Judicial (SAGJ) _ Órgano Judicial. *Autoridad nacinal para la innovasion gubernamental.* [En línea] [Citado el: 1 de febrero de 2015.] http://innovacion.gob.pa/noticia/1656.

6. Sitio Oficial Poder Judicial. [En línea] [Citado el: 1 de febrero de 2015.] www.poder-judicial.go.cr/.

7. Castro Morell, MSc. Daniel E. *Sistema para la tramitación de procesos penales.* Universidad "Marta Abreu" de Las Villas. Villa Clara : s.n., 2008.

8. *EL ÁMBITO PROCESAL DEL DERECHO PENAL ECONÓMICO EN CUBA .* Megías, Carlos Alberto Rodrigues. http://www.ciidpe.com.ar/area5/dpre%20rodriguez%20mejias.pdf, La Habana : Editorial En torno al Acuerdo No. 172 del Consejo de Gobierno del TSP , 2010.

9. Pressman, Roger S. *La Ingeniería del Software, un enfoque práctico.* 2005. ISBN: 9701054733.

10. Guerra, Prof: Lic. Roberto. Metodología RUP. [En línea] [Citado el: 7 de enero de 2015.] http://es.scribd.com/doc/31440864/Metodologia-RUP.

11. BLANCO, K.R. y AGUERO, D.N. CICLO DE VIDA DE PROYECTOS PILOTOS DEL PROGRAMA DE MEJORA. [En línea] 2010. [Citado el: 9 de 5 de 2015.] publicaciones.uci.cu/index.php/SC/article/download/234/367.

12. International, Visual Paradigm. Visual Paradigm for UML product overview. [En línea] [Citado el: 19 de enero de 2015.] http://www.visual-paradigm.com/support/documents/vpumluserguide/12/13/5963_visualparadi.html.

13. Ecured. [En línea] [Citado el: 1 de 3 de 2015.] http://www.ecured.cu/index.php/Visual_Paradigm.

14. Ecured. [En línea] [Citado el: 9 de 4 de 2015.] http://www.ecured.cu/index.php/Framework.

15. Eguiluz, Javier. *Desarrollo web ágil con Symfony2.* 2011.

16. —. *Desarrollo web ágil con Symfony2.* 2011.

17. github.com. Doctrine 2 ORM. [En línea] [Citado el: 15 de enero de 2015.] https://github.com/doctrine/doctrine2.

18. twitter.github.com. Bootstrap. Sleek, intuitive, and powerful front-end framework for faster and easier web development. [En línea] [Citado el: 15 de enero de 2015.] http://twitter.github.com/bootstrap.

19. Flanagan, David. *jQuery Pocket Reference.* EEUU : O'Reilly Media, Inc., 2011.

20. Pacheco, Nacho. *Manual de Twig.* 2011.

21. Group, The PHP. ¿Qué es PHP? - Manual. [En línea] [Citado el: 20 de enero de 2015.] http://php.net/manual/es/intro-whatis.php.

22. desarrolloweb.com. [En línea] [Citado el: 5 de febrero de 2015.] http://www.desarrolloweb.com/javascript/.

23. www.netbeans.org. NetBeans IDE 7.2.1 Release Information. [En línea] [Citado el: 27 de enero de 2015.] http://netbeans.org/community/releases/72/index.html.

24. postgresql.org.es. Sobre PostgreSQL. [En línea] [Citado el: 5 de febrero de 2015.] http://www.postgresql.org.es/sobre_postgresql.

25. apache.org. What is apache. [En línea] [Citado el: 20 de enero de 2015.] http://wiki.apache.org/httpd/FAQ#What_is_Apache.3F.

26. Ben Collins-Sussman, Brian W. Fitzpatrick, and C. Michael Pilato. Version Control with Subversion. [En línea] [Citado el: 5 de febrero de 2015.] http://svnbook.red-bean.com/en/1.6/svn.intro.whatis.html.

27. voigtmann informaions technologien. [En línea] [Citado el: 9 de 4 de 2015.] http://www.voigtmann.de/es/desarrollo-de-software/arquitectura-de-software/.

28. Reynoso, Carlos. Kiccillof, Nicolás. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. [En línea] [Citado el: 6 de enero de 2015.] http://cic.javerianacali.edu.co/wiki/lib/exe/fetch.php?media=materias:estiloypatron.pdf.

29. Alexander, Christopher. *A Pattern Language (vol II).* 1997.

30. Juan, Pavón Mestras. *Patrones de diseño orientado a objetos.* Facultad de Informática Universidad Complutense, Madrid : s.n., 2004.

31. Gamma, Erich. Helm, Richard. Johnson, Ralph. Vlissides, John. *Design Patterns: Elements of Reusable Object-Oriented Software.* 1995. ISBN: 0-201-63361-2.

32. informática, Facultad de. *Patrones del "Gang of Four".* Facultad de informática - Universidad Politécnica de Madrid : s.n.

33. symfony.es. Inyección de dependencias en Symfony 2. [En línea] 2 de abril de 2009. [Citado el: 3 de febrero de 2015.] http://www.symfony.es/noticias/2009/04/02/inyeccion-de-dependencias-en-symfony-2/.

34. Pressman, Roger S. http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/flores_r_mi/capitulo4.pdf. [En línea] 1998. [Citado el: 4 de 4 de 2015.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/flores_r_mi/capitulo4.pdf.

35. hernandez, Leovigilda. MODELO DE IMPLEMENTACIÓN . [En línea] 2013. [Citado el: 4 de 4 de 2015.] http://ithleovi.blogspot.com/2013/06/unidad-5-modelo-deimplementacion-el.html.

36. saavedra, Jorge. Ecured. [En línea] [Citado el: 11 de 4 de 2015.] http://www.ecured.cu/index.php/Patrones_de_Asignaci%C3%B3n_de_Responsabilidades.

37. librosweb. [En línea] http://librosweb.es/libro/symfony_2_x/capitulo_16/que_es_un_servicio.html.

38. Adictosaltrabajo.com. [En línea] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=Spring_Container_Inyeccion_Dependencias.

39. Jacobson, Ivar y otros. *Object-Oriented Software Engineering—A Use Case.* 1992.

40. Microsoft devoper netwok. [En línea] [Citado el: 10 de 4 de 2015.] https://msdn.microsoft.com/es-es/library/dd409377.aspx.

41. Castro, Fernando Lorenzo. Modelo de Datos. Conceptos y clasificación. Universidad de Castilla-La Mancha, Escuela Superior de Informática : s.n., 1999.

42. MIcrosoft Diveloper Network. [En línea] [Citado el: 19 de 04 de 2015.] https://msdn.microsoft.com/es-es/library/dd409390.aspx.

43. Alva, Eduardo Rivera. Arquitectura de Software II. Diagrama de Componentes y Despliegue. [En línea] 2012. [Citado el: 10 de 4 de 2015.]

44. Vega Lebrún Carlos, Rivera Prieto Laura Susana,García Santillán Arturo. eumed.net. *MEJORES PRÁCTICAS PARA EL ESTABLECIMIENTO Y ASEGURAMIENTO DE LA CALIDAD DE SOFTWAR.* [En línea] [Citado el: 11 de 5 de 2015.] http://www.eumed.net/libros-gratis/2008a/351/Metricas%20de%20Software.htm.

45. Pressman, Roger. [En línea] [Citado el: 17 de 5 de 2015.] http://www.uv.mx/personal/asumano/files/2012/08/MetricasTecnicas.pdf.

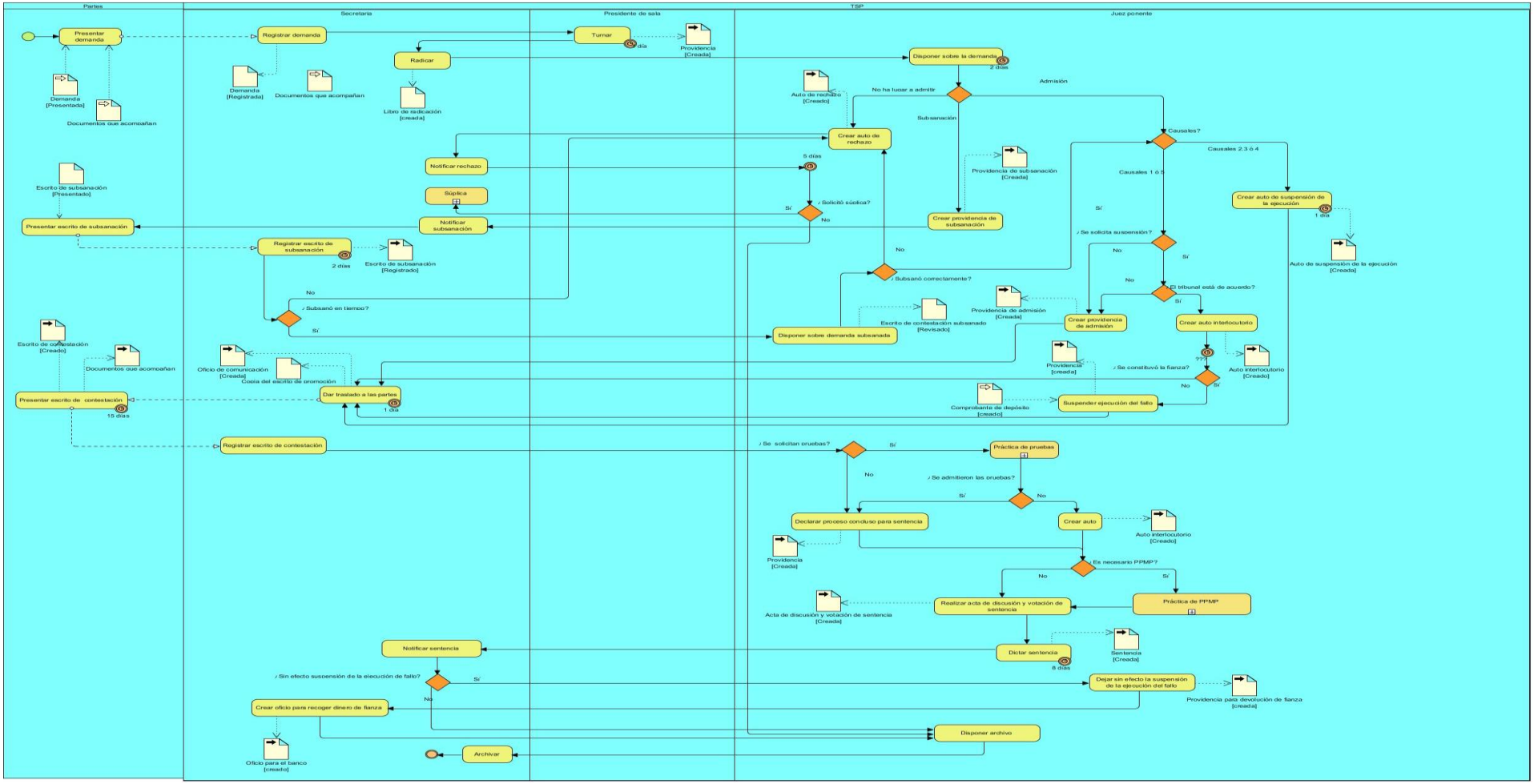
46. S, Pressman Roger. *Ingenieria de Software ,Metricas_Producto.* 2005. ISBN: 9701054733.

47. pruebasdesoftware. *Gestión de Calidad y Pruebas de Software.* [En línea] [Citado el: 18 de 5 de 2015.] http://pruebasdesoftware.com/laspruebasdesoftware.htm.

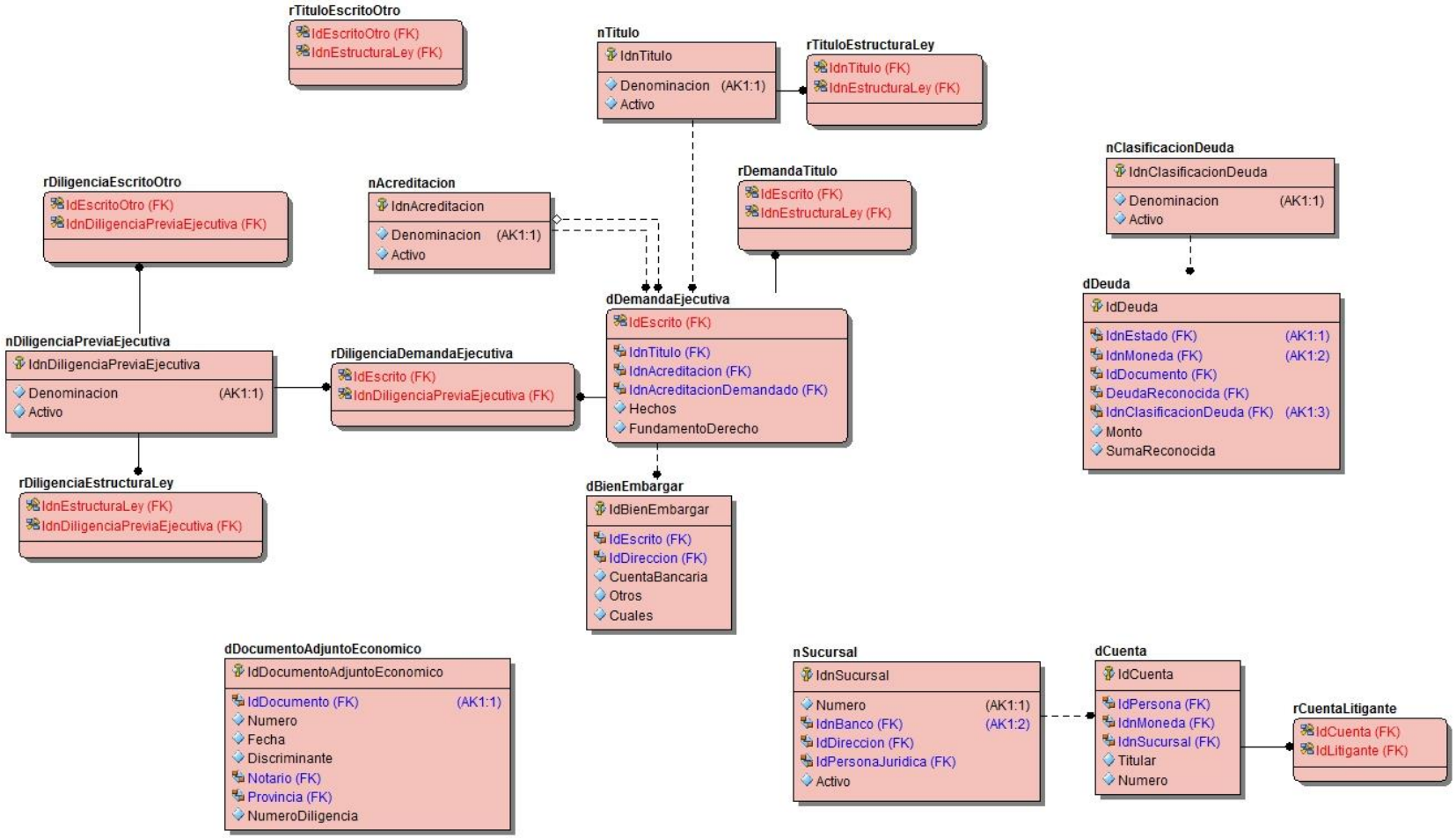
48. Ruiz, Roberto. *Las pruebas de software y la importancia en las organizaciones.* Veracurz : s.n., 2011.

49. Pressman, Roger S. *La Ingeniería del Software, un enfoque práctico.* ISBN: 9701054733.

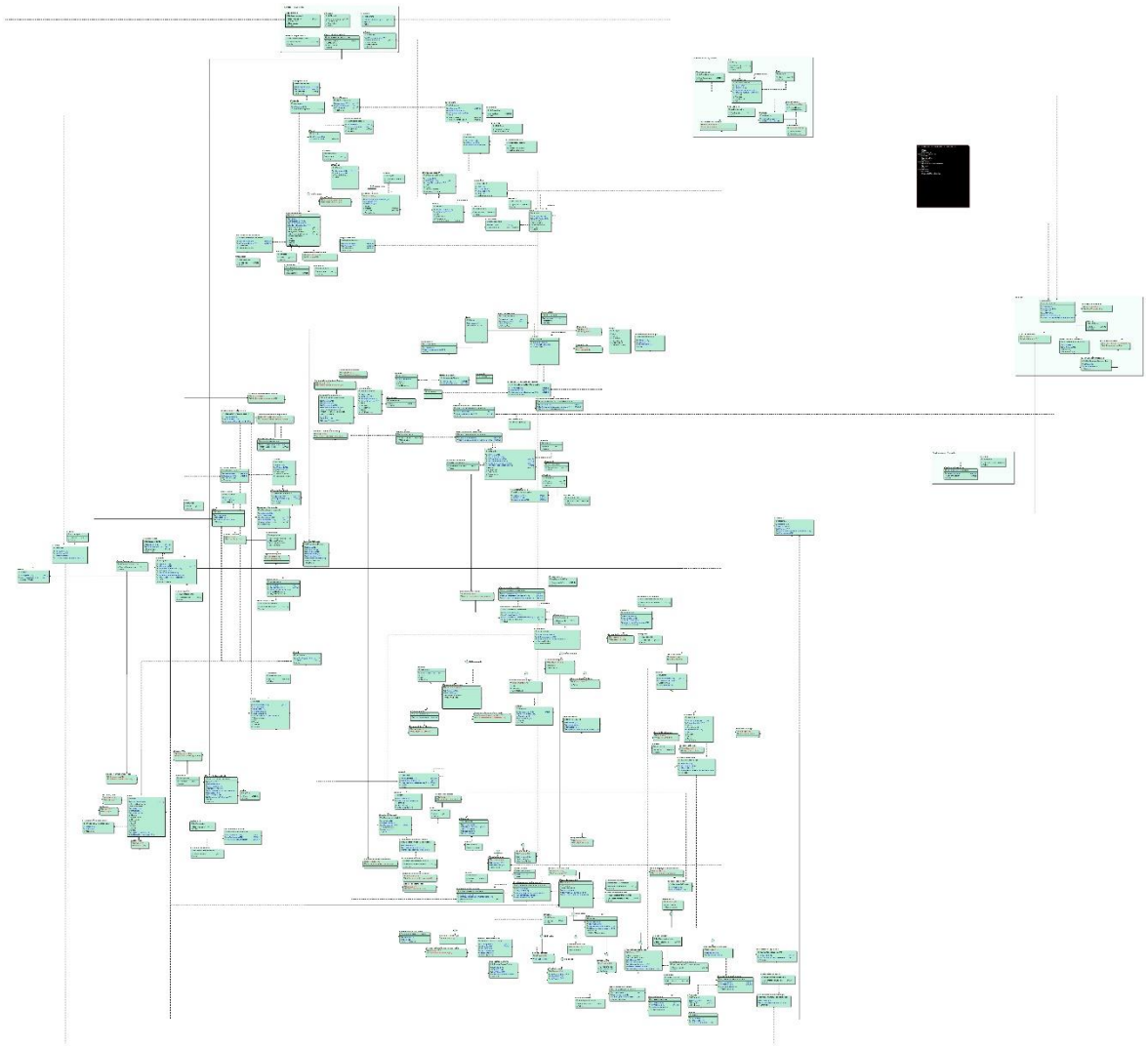
ANEXOS



Anexo 1 Diagrama de proceso de negocio para la instancia Suprema de los TPC



Anexo 2 Diagrama modelo de datos



Anexo 3 Diagrama modelo de datos de Común