Universidad de las Ciencias Informáticas

Facultad 3

Desarrollo del módulo Revisiones de causas penales del Sistema de Informatización para la Gestión de las Fiscalías fase II

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Remberto Loanny García Sosa

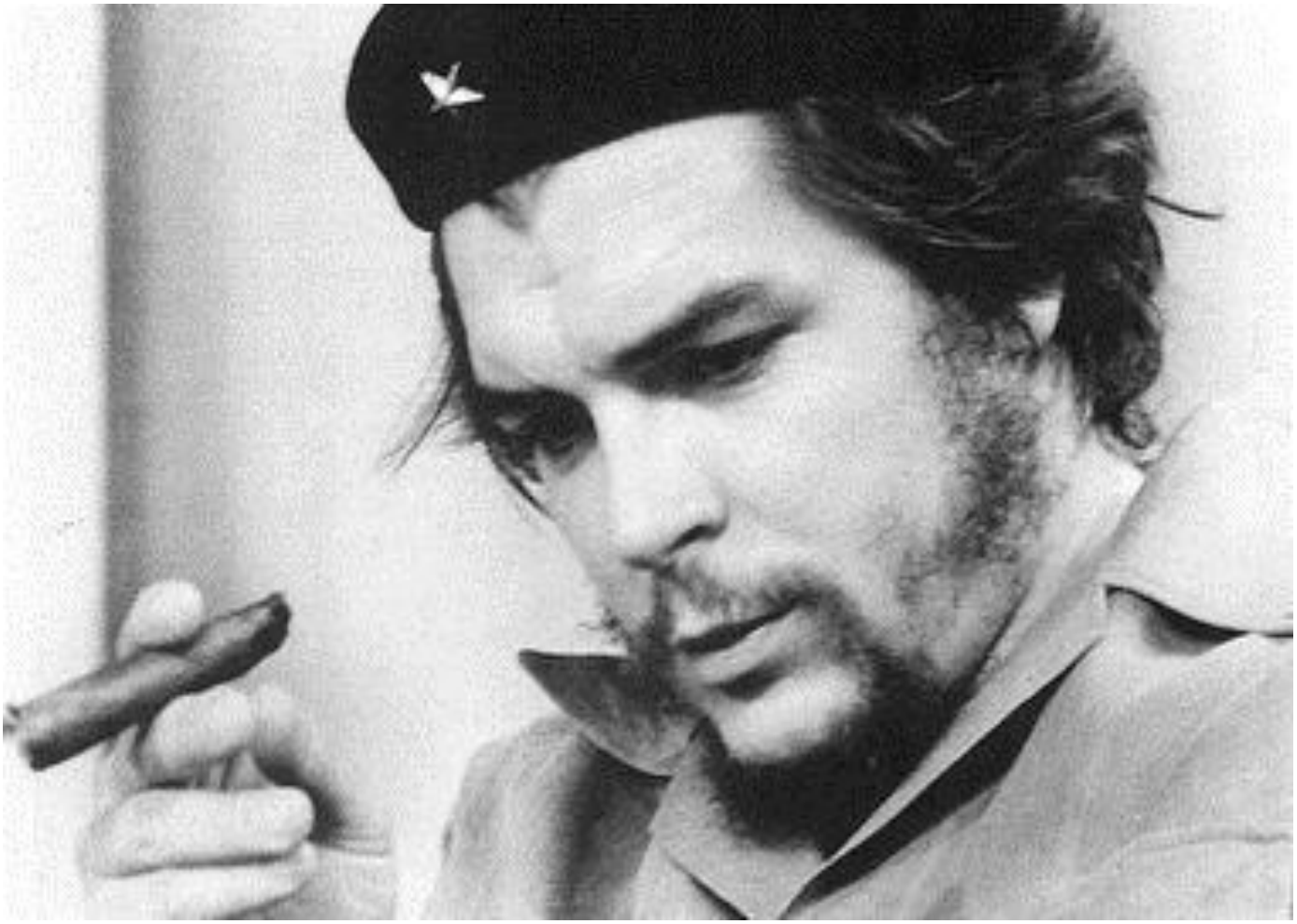
Carlos Alberto Taborda Espinosa

Tutor: MSc. Marieta Peña Abreu

Co-tutor: Ing. Frank Lemus Paz

La Habana, junio de 2013

"Año 55 de la Revolución"



"La única lucha que se pierde es la que se abandona"

Ernesto Che Guevara

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____	_____
Remberto Loanny García Sosa	Carlos Alberto Taborda Espinosa
_____	_____
MSc. Marieta Peña Abreu	Ing. Frank Lemus Paz

Datos de Contacto

Autor: Remberto Loanny García Sosa.

Correo electrónico: rlgarcia@estudiantes.uci.cu

Autor: Carlos Alberto Taborda Espinosa.

Correo electrónico: ctaborda@estudiantes.uci.cu

Tutor: MSc. Marieta Peña Abreu.

Correo electrónico: mpabreu@uci.cu

Tutor: Ing. Frank Lemus Paz.

Correo electrónico: flemus@uci.cu

AGRADECIMIENTOS

A mis padres, por su apoyo en todo momento, por confiar en mí siempre, por su enseñanza. A toda mi familia por su dedicación, por ser la mejor cada día, por confiar en mí, por la educación que me han dado. A mis amigos del IPI, a Aramis, Henry, por no dejar que la distancia se interpusiera en nuestra amistad. A mi compañero de tesis Carlos por su dedicación en este trabajo, por su amistad, por todos esos momentos que este trabajo nos hizo compartir, muchas gracias por todo. A Marieta mi tutora, por todo ese tiempo que nos dedicaste. A mi novia Wendy por toda su dedicación y paciencia conmigo. A mis hermanos Yosviel, Raidel, Ricardo, por todas las fiestas que compartimos a lo largo de estos 5 años por su amistad, su confianza, su apoyo, muchas gracias. A mis compañeros de aula, del apto 209, a todos ustedes gracias por su amistad. A todos mis compañeros de proyecto, en especial a Luis Mariano, los profesores Figueroa, Dailín, Felipe, Frank, y otros…., a todo ese equipo muchas gracias. A todos los profes que en estos años contribuyeron a mi formación como profesional. Gracias.

Remberto.

AGRADECIMIENTOS

Quiero agradecer mi hermano y a mi cuñada la negra, por todos los sacrificios que han hecho para que yo llegara hasta aquí, gracias por su dedicación, por toda la confianza que siempre me han dado y por creer siempre en mí. Los adoro con todo mi corazón. A Yoli, Jorgi, Yanelys, gracias por ser los mejores hermanos del mundo, gracias por todos los consejos, por apoyarme siempre. Y a mi primo Bob Esponja. A mi hermana Vivian por ser mi guía, y darme la luz siempre, a la que considero como la más lista, a Lilian por ayudarme tanto en los momentos de desesperación con Ariel el animal del regueton, gracias por su apoyo. A Yolanda, gracias por ser madre para mí, por todos los sacrificios que has hecho por mí, por toda tu confianza, gracias por ser la mejor tía del mundo y por dejar que mi hermano siempre este conmigo y darme sus hijos como hermanos. A toda mi familia en general, a todos mis tíos, tías, primas y primos. A Rember, por ser más que un compañero de tesis, por ser mi amigo, gracias por todo tu esfuerzo para que esta tesis saliera adelante, por todos los momentos compartidos, muchas gracias por todo. A Marieta, gracias por todo el tiempo dedicado a esta tesis a pesar de todo el trabajo que tienes. A Raydel, Michelin, Carlos del Junco, Rafa por ser los amigos de la uci y de toda la vida. A mis compañeros del proyecto, Luis Manuel Borges, Blackberry, Figueroa, Dailin, Felipito, Giorgi, Hardisco y tantos otros que le pregunté alguna que otra duda…en fin a todo el equipo muchas gracias, por todo lo que me enseñaron, por los momentos compartidos, por los días y las noches en el lab. Muchas gracias por todo. A mis compañeros del 3507 gracias por estos años maravillosos.

Carlos

RESUMEN

El departamento de Protección a los Derechos Ciudadanos (PDC), se encarga de controlar y supervisar el proceso de tramitación, supervisión y respuesta en disposición legal a todo ciudadano. El presente trabajo de diploma tiene como propósito desarrollar el módulo “Revisión de causas penales” en esta área.

Para el cumplimiento del objetivo propuesto, se realizó un estudio de los procesos relacionados con las solicitudes de revisiones que se llevan a cabo en el departamento PDC de la Fiscalía General de la República (FGR), a partir de los cuales se identificaron las funcionalidades que el sistema debe tener. Se hizo un análisis y selección de las herramientas y tecnologías existentes para el desarrollo de la aplicación. Se efectuó el diseño e implementación del sistema y la evaluación de la calidad de los resultados obtenidos.

Con la implementación de esta propuesta se espera que los fiscales tengan a su disposición un sistema que cumpla con sus expectativas, obteniendo una aplicación que mejore en gran medida el proceso de gestión fiscal, así como el manejo de la información utilizada para su trabajo.

Palabras Claves

PDC, Revisiones de causas penales, Diseño, Implementación.

TABLA DE CONTENIDOS

ÍNDICE DE TABLA

ÍNDICE DE FIGURA

INTRODUCCIÓN

El surgimiento y desarrollo de las ciencias informáticas ha intervenido directamente en los procesos empresariales y de comunicación, generando nuevos paradigmas y rompiendo muchos existentes hasta el momento. El desarrollo acelerado de esta ciencia conlleva cambios importantes en la forma en que se llevan a cabo tareas rutinarias dentro de una organización y en el ámbito social. El gobierno cubano, siempre con el ideal de tener un pueblo culto y preparado, ha emprendido la tarea de informatizar la sociedad cubana y utilizar los avances de esta ciencia en beneficio del pueblo y la economía. La Universidad de las Ciencias Informáticas (UCI), como programa de la Revolución, de importancia fundamental para lograr este objetivo; tiene la tarea de formar profesionales que agilicen y sean soporte fundamental para llevar a cabo esta labor.

Uno de los organismos del Estado que se encuentra desde el año 2007 enfrascada en la informatización de sus procesos principales, mediante el uso de las Tecnologías de la Informática y las Comunicaciones (TIC), es la Fiscalía General de la República; con el fin de hacer cumplir la Constitución de la República de Cuba con calidad, rapidez y profesionalidad. Un proceso de importancia clave para esta institución es atender las quejas, solicitudes o denuncias que dentro del ámbito legal formulen los ciudadanos sobre alguna presunta violación de sus derechos en materia penal.

La UCI cuenta con numerosos proyectos productivos, los cuales tienen la tarea de informatizar procesos claves en determinadas instituciones. Uno de estos proyectos es el Sistema de Informatización de las Gestión de las Fiscalías II (SIGEF II); que fue creado con el objetivo de incrementar los niveles de gestión de la información, garantizar su adecuada organización y preservación.

El proyecto SIGEF II está compuesto por subsistemas o áreas que incluyen los módulos correspondientes a los procesos que se realizan en la FGR. Este sistema contempla el área de PDC. Dentro de la misma, uno de los procesos es la Revisión de causas penales, que se ocupa de realizar chequeos a los distintos procesos legales, dada una solicitud de un ciudadano siempre dentro de los términos legales.

En la actualidad este proceso se torna largo, engorroso y difícil de ejecutar, por el cúmulo de información que se debe manejar y la gran cantidad de recursos humanos necesarios para procesarla, afectándose el cumplimiento de las tareas en el tiempo requerido así como el control de las mismas. Otro de los

problemas que se presenta, es la documentación de los distintos trámites; pues esta se encuentra manuscrita siendo difícil de entender en algunos casos ya que se encuentra propensa al deterioro natural. Otro aspecto necesario a tener en cuenta es el cúmulo de información almacenada en la FGR que por ser abundante dificulta la búsqueda y análisis de información, así como su almacenamiento por largos períodos de tiempo siendo necesario destruir aquella que sobrepase los cinco años considerándola no relevante.

Dadas las condiciones anteriores es necesario crear una aplicación que facilite el tratamiento y registro de toda la información relacionada con el proceso, mejorando la calidad del trabajo de los fiscales en nuestro país. Para lograrlo es de suma importancia agregarle nuevas funcionalidades al sistema ya existente (SIGEF I), de manera que se logre dar una solución eficiente *a* los problemas anteriormente tratados.

Dada las circunstancias previas, es identificado el siguiente **problema de la investigación**: ¿Cómo mejorar la gestión de la información del proceso revisiones de causas penales del área PDC de la Fiscalía General de la República, de manera que se contribuya a la ejecución y control del mismo?

Este problema se enmarca en el **objeto de estudio**: Proceso de desarrollo de software de gestión en la informática jurídica.

En función de resolver el problema propuesto se ha definido como **objetivo general**: Desarrollar el módulo Revisiones de causas penales del Sistema de Informatización de la Gestión de las Fiscalías fase II, para mejorar la ejecución y control de su proceso, enmarcándose el **campo de acción** en proceso de desarrollo de software de gestión en el proceso de revisiones de causas penales del área de PDC de la Fiscalía General de la República.

Se define como **idea a defender:** Con el desarrollo del módulo Revisiones de causas penales del Sistema de Informatización de la Gestión de las Fiscalías fase II, se mejora la gestión de la información del proceso informatizado en el área PDC de la Fiscalía General de la República, de manera que se contribuya a la ejecución y control del mismo.

Para darle cumplimiento al objetivo general propuesto se han definido los siguientes **objetivos específicos:**

- ✓ Elaborar el marco teórico de la investigación mediante un estudio del estado del arte alrededor de los procesos fiscales, modelo de desarrollo, técnicas y herramientas de software.
- ✓ Realizar el análisis y diseño del módulo Revisiones de causas penales del Sistema de Informatización de la Gestión de las Fiscalías fase II, mediante el modelo de desarrollo de software establecido.
- ✓ Implementar el módulo Revisiones de causas penales del Sistema de Informatización de la Gestión de las Fiscalías fase II, mediante las herramientas y tecnologías seleccionadas.
- ✓ Validar la solución propuesta mediante pruebas de software y el cumplimiento de los objetivos de la investigación.

Para desarrollar el presente trabajo, es necesario definir **tareas de la investigación** que guíen el camino a seguir para dar cumplimiento a los objetivos específicos:

- ✓ Estudio del estado del arte de sistemas de gestión de informática jurídica.
- ✓ Estudio del proceso de desarrollo de software en el proyecto Sistema de Informatización de la Gestión de las Fiscalías fase II.
- ✓ Estudio del proceso revisiones de causas penales del área PDC.
- ✓ Identificación, descripción, validación de los requisitos asociados al módulo Revisiones de causas penales del Sistema de Informatización de la Gestión de las Fiscalías fase II.
- ✓ Realización del diagrama de clases del módulo Revisiones de causas penales del Sistema de Informatización de la Gestión de las Fiscalías fase II.
- ✓ Validación del diseño del módulo Revisiones de causas penales del Sistema de Informatización de la Gestión de las Fiscalías fase II.
- ✓ Implementación del módulo Revisiones de causas penales del Sistema de Informatización de la Gestión de las Fiscalías fase II.
- ✓ Realización del diseño de casos de prueba del módulo Revisiones de causas penales del Sistema de Informatización de la Gestión de las Fiscalías fase II.
- ✓ Ejecución de pruebas de caja blanca al código fuente obtenido en la implementación del módulo Revisiones de causas penales del Sistema de Informatización de la Gestión de las Fiscalías fase II.
- ✓ Validación de los resultados obtenidos a través de pruebas de caja negra al sistema para comprobar su correcto funcionamiento.

Para la realización del trabajo fue necesario la utilización de los siguientes métodos y técnicas de investigación:

Métodos Teóricos:

Histórico-Lógico*:* Se utilizó para investigar todos los antecedentes en materia penal que existen sobre el proceso de revisión de causas penales en el departamento de Protección a los Derechos al Ciudadano (PDC) de la FGR, mediante la documentación recogida durante el último año.

Analítico-Sintético: Se utilizó para el procesamiento de toda la información relacionada con el tema de investigación, analizando los documentos que permitieran extraer los elementos más significativos relacionados y conclusiones importantes con el objeto de estudio.

Métodos Empíricos:

Entrevista: Se utiliza en el intercambio con el cliente para adquirir información sobre el negocio, la organización, necesidades existentes e intereses del cliente y los requisitos que debe cumplir en el desarrollo del software.

El presente trabajo de diploma está compuesto por 3 capítulos estructurados de la siguiente manera:

En el **Capítulo 1** se enmarca la **Fundamentación Teórica** que comprende el estado del arte, la cual incluye un estudio de sistemas existentes para la gestión de revisión de causas penales en el ámbito internacional; se enuncian los principales conceptos relacionados con el tema, así como un estudio de las metodologías, lenguajes y herramientas usadas como apoyo para darle solución al problema planteado.

En el **Capítulo 2** se realiza el **Análisis y Diseño** del módulo. En este capítulo se exponen los requisitos tanto funcionales como no funcionales, teniendo en cuenta las restricciones o políticas a cumplir por el sistema. Además se presenta el diseño a través de los diagramas derivados de dicha disciplina, mediante el modelo de desarrollo escogido.

En el **Capítulo 3** se presenta la **Implementación y Prueba** de la solución. En este capítulo quedan establecidos el diagrama de despliegue y el diagrama de componentes, que conformarán el Modelo de

Implementación, además de la descripción detallada de los paquetes de implementación. Igualmente se exponen las pruebas realizadas a la solución para asegurar el correcto funcionamiento de la aplicación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción.

En el presente capítulo se realiza un estudio del arte, las herramientas, tecnologías y patrones que permitieron realizar un correcto análisis, diseño e implementación de la propuesta de solución. Se incluyen aspectos relacionados con las tendencias nacionales e internacionales y las aplicaciones que informatizan el proceso de quejas, solicitudes de revisiones de las Fiscalías en el mundo. Se ofrece un estudio de las tecnologías y herramientas de desarrollo de software, para el modelado de prototipos no funcionales de Interfaz de Usuario (IU). Se presentan patrones para cada una de las disciplinas desarrolladas, y los lenguajes de programación encaminados a la web. Finalmente se hace alusión a las métricas para evaluar la calidad de las disciplinas descritas en dicho capítulo.

1.1 Conceptos fundamentales.

FGR: Órgano del Estado al que corresponden, como objetivos fundamentales, el control y la preservación de la legalidad, sobre la base de la vigilancia del estricto cumplimiento de la Constitución, las leyes y demás disposiciones legales, por los organismos del Estado, entidades económicas y sociales y por los ciudadanos; así como la promoción y el ejercicio de la acción penal pública en representación del Estado (1).

Queja: Inconformidad ciudadana sobre violación de sus derechos constitucionales, garantías legalmente establecidas e infracciones cometidas en actos o disposiciones de organismos y funcionarios del estado y de entidades económicas y sociales (2).

Solicitante: Persona natural que presenta una queja o solicitud.

Materia Penal: “El Derecho Procesal Penal es, el conjunto de normas jurídicas emanadas del poder del Estado que ordenan el proceso, sea en su conjunto, sea en los actos particulares que lo integran y que tiene por finalidad aplicar el derecho material para restablecer la legalidad quebrantada” (3).

Rollo fiscal: Es equivalente a un expediente. En el que se acumulan una serie de documentos asociados al proceso.

Legislación: Son todas las leyes o documentos vigentes que pueden ser consultados para resolver un caso determinado.

Legalidad: Todo lo que es legal, es decir, que está permitido ante la ley.

Orientación Jurídica: Consiste en orientar, mediante escrito con el formato de una carta, al reclamante sobre como encauzar en el orden legal su pedimento o solicitud, la procedencia o no de su reclamación o asunto planteado, con fundamento en la norma jurídica que así lo establece.

Traslado Definitivo (TD): Tipo de traslado que decide darle un fiscal a una queja, solicitud o denuncia para que sea tramitada en el órgano que le compete.

Persona Natural: Persona Natural es una persona humana que ejerce derechos y cumple obligaciones a título personal.

Traslado Bajo Control (TBC): Tipo de traslado que se le da a una queja, solicitud o denuncia por no ser competencia de la fiscalía en la que se encuentra, el mismo es enviado a la fiscalía inmediata inferior bajo indicaciones.

Respuesta verbal: Intercambio directo con la persona que posibilita una comprensión mayor y más completa de todas las interrogantes del solicitante (2).

1.2 Escenario actual del proceso.

La institución encargada de atender las revisiones de causas penales que en el orden legal formulen los ciudadanos sobre presuntas violaciones de sus derechos, se encuentra en el área de PDC perteneciente a la Fiscalía General de la República de Cuba (FGR); teniendo como objetivo fundamental el control y la preservación de la legalidad sobre la base de la vigilancia del estricto cumplimiento de la Constitución, las leyes y demás disposiciones legales, tramitaciones, investigaciones y respuestas a las quejas y reclamaciones de la población y la tramitación de los procesos y procedimientos de revisión.

En dicha área el Fiscal designado atiende, investiga y responde las quejas y reclamaciones, que formulan los ciudadanos, sobre una sanción que se ha impuesto de manera injusta o que faltan elementos importantes que no se tuvieron en cuenta en el plazo de 90 días (3).

En este momento los procedimientos por parte de la FGR para tramitar el proceso de solicitudes de revisión, se realizan de forma manual, con lo que crea un alto volumen de información por lo que se utiliza gran cantidad de tiempo para componer todos los documentos pertinentes. Estos documentos se conservan en las condiciones propicias, pero al paso del tiempo se deterioran, dificultando su uso posterior; lo que contribuye a la demora de los procesos hasta después de vencido el plazo. El envío de expedientes, si es necesario, a instancias superiores se realiza por correo tradicional; lo cual suma más tiempo si el mismo es devuelto para que se realicen otras investigaciones. Existe poco control y supervisión por parte del nivel superior de la tramitación del proceso de revisiones de causas penales en los órganos provinciales, además provoca que dichos procesos sean tediosos y propensos a errores (4).

1.3 Valoración del estado del arte.

Tendencias Internacionales y Nacionales.

Las solicitudes de revisión sobre distintas situaciones que respectan a inconformidades en materia penal, es un asunto de carácter internacional. En numerosos lugares del mundo existe un órgano o persona natural que procesa las revisiones. En la mayoría de los países capitalistas se encuentra un “defensor”, que recibe las solicitudes y las dirige al fiscal correspondiente para que este investigue y lleve a cabo el proceso de revisión, para detectar cualquier violación de los derechos como ciudadano. En países como Chile, Colombia y Argentina, es la Fiscalía General del Estado quien cuenta con servicios de Atención a los Derechos del Ciudadano (ADC), que tiene entre sus funciones tramitar las quejas formuladas por los ciudadanos por correo electrónico, haciéndolas llegar en su caso al órgano competente para resolverlas (5).

En la actualidad el sistema cubano tiene una tendencia hacia el uso de las nuevas tecnologías en el sector jurídico, por lo que se ha propuesto crear sistemas de gestión fiscal que faciliten el manejo de la información para agilizar el proceso de revisión penal. En Cuba las solicitudes de revisión se envían por correo postal o se presentan personalmente en la FGR o cualquier otra fiscalía del país. No existe un defensor del pueblo encargado de estos asuntos, todo el proceso de tramitación de la queja es realizado

por el fiscal u órgano según corresponda; para esto se cuenta con un término de noventa días hábiles desde el momento en punto en que fue recibida la misma. La solicitud es estudiada y tramitada por el fiscal que corresponda según la materia que trate, obteniéndose como resultado el pronunciamiento redactado por el fiscal que constituirá la respuesta al solicitante.

Aplicaciones Informáticas existentes.

El desarrollo de las nuevas tecnologías de la información y las comunicaciones ha permitido transmitir, gestionar y compartir muchos datos de revisiones penales, pero el exceso de documentación hace que se tenga que invertir mucho tiempo en ello, de esta manera surge la necesidad de gestionar la información, que no es más que el proceso de analizar, utilizar, recuperar y almacenar la información que se ha obtenido y registrado, para permitir su aprovechamiento de una manera más eficiente, rápida y organizada.

Debido a esto se hace necesario realizar un estudio de las diferentes soluciones de gestión fiscal que tratan las revisiones.

Poder Judicial de la República de Chile.

Sitio web a través del cual la rama Judicial de la República de Chile "presta servicio a todas las personas que acudan al sistema, proporcionando información y consulta, sistemas de tramitación de causas en cortes de apelaciones en materia civil, laboral, familia, penal y de cobranza laboral y previsional" (6). Brinda la información de una de las cortes de apelaciones del país. También permite consultar por la fecha y hora de la realización de una audiencia determinada. En cuanto causas civiles, permite consultar respecto a todos los tribunales del país que se encuentran interconectados y revisa en todos los juzgados con competencia civil interconectados. Permite a los usuarios buscar las causas de la corte suprema, las causas de cobranza laboral, laborales y previsional. El sistema penal permite obtener el enunciado de las causas y los procedimientos asociados a los casos (6).

Programa Gobierno en línea.

Gobierno en línea, liderada por el Ministerio de Tecnologías de la Información y las Comunicaciones de Colombia, brinda atención a la ciudadanía de peticiones, quejas y reclamos, la cual ofrece información y/o consulta, quejas por medios electrónicos de insatisfacción con la conducta, la acción de los servidores

públicos o de los particulares que llevan a cabo una función estatal y que requiere una respuesta, reclamos de insatisfacción referida a la prestación de un servicio o la deficiente atención de una autoridad pública, es decir, es una declaración formal por el incumplimiento de un derecho que ha sido perjudicado o amenazado, ocasionado por la deficiente prestación o suspensión injustificada del servicio (7).

Poder judicial de la nación.

Poder judicial de la nación de la República de Argentina, es una aplicación web que brinda el conocimiento y decisión de todas las causas que versen sobre puntos regidos por la constitución, y por las leyes nacionales, salvo los casos que corresponden a la justicia provincial, en cuanto a materias de justicia nacional en lo penal económico, justicia nacional en lo civil y comercial federal, justicia nacional en lo criminal y correccional federal, justicia nacional en lo contencioso administrativo federal (8).

Una vez realizado un análisis de los sistemas anteriores se puede concluir que los mismos no son factibles para ser aplicados en el entorno cubano, debido que fueron creados de acuerdo a las leyes de los gobiernos a los cuales pertenecen y adaptarlos al nuestro conllevaría gran gasto de recursos; además que no están a la venta, ni disponibles para ser utilizados. La Fiscalía General de la República necesita un sistema que se ajusten a las leyes cubanas vigentes.

1.4 Formulación de la propuesta de solución.

Características del proyecto SIGEF II.

Uno de los objetivos de la UCI es informatizar la sociedad cubana; y para su cumplimiento se lleva a cabo actualmente el Sistema de Informatización de la Gestión de las Fiscalías II (SIGEF II), siendo el seguimiento de SIGEF fase I.

Esta nueva fase está enfocada en la continuación de informatización de los procesos fiscales en otras de las áreas existentes en la Fiscalía General de la República de Cuba, para crear una solución que establezca un incremento en la calidad de la tramitación, supervisión y control en tiempo real de los procesos fiscales y reducción de los términos de las actividades, así como utilización óptima de la fuerza fiscal (9).

La solución de software se desarrollará como aplicación web, usando la plataforma de desarrollo Linux Apache Postgres PHP (LAPP) como estrategia del país de alcanzar la soberanía tecnológica. Para

coordinar las actividades necesarias para transformar los requisitos de usuario en los del sistema de gestión, se usará la guía de desarrollo basada en el Proceso Unificado de Rational (RUP) empleada en la universidad descrita en el Programa de Mejora (10).

SIGEF II está estructurado por diversos subsistemas que se corresponden con las áreas de procesos que componen la Fiscalía General de la República. Dentro de estos subsistemas se encuentra el de PDC, que encierra el proceso de solicitudes de revisiones de causas penales, el cual es el objetivo fundamental del desarrollo de este trabajo de diploma.

La informatización de este proceso dará solución a una serie de problemas existentes en las fiscalías cubanas, además traerá consigo beneficios a distintos niveles:

A nivel de ciudadanos de la República de Cuba:

- ✓ Rapidez en trámites legales.

A nivel de Fiscalía General de la República de Cuba:

- ✓ Disponer de un sistema para gestionar centralmente el proceso.
- ✓ Lograr la supervisión y control en tiempo real de los procesos de revisión.
- ✓ Contribuir a la preservación de la información por un período mínimo de 30 años.

A nivel de Fiscales:

- ✓ Disponer de un sistema de ayuda a la toma de decisiones.
- ✓ Aumentar el cumplimiento de los términos establecidos.

A nivel de Trabajadores de las fiscalías:

- ✓ Rapidez en el tratamiento, envío, búsqueda y recepción de la información.

SIGEF fase II tendrá beneficios reveladores para el país, ya que contribuirá a que la FGR de Cuba cumpla eficazmente con su objetivo de controlar y preservar el estricto cumplimiento de la legalidad; por lo que se

hace preciso elaborar la solución, con el objetivo de contener el proceso de revisión de causas penales atendidas en el departamento de PDC que no fue tratado en la primera fase, de manera que al realizar el diseño, implementación y prueba de estos procesos se obtenga un sistema robusto.

1.5 Metodología de desarrollo de software.

Un proceso de desarrollo de software encierra una serie de actividades y resultados encaminados a producir un software (11). La metodología de desarrollo de software se encarga de elaborar estrategias; centradas en las personas o los equipos, orientadas hacia la funcionalidad y la entrega. Su objetivo es elevar la calidad del software a través de un mayor control sobre el proceso (12).

Desde la fase I, el proyecto SIGEF viene rigiéndose por la metodología RUP, principalmente por la ventaja de guiar a los equipos de proyecto en cómo administrar el desarrollo iterativo de un modo controlado, mientras se balancean los requisitos y los riesgos del proyecto. Este de manera clara describe los diversos pasos involucrados en la captura de los requisitos, detalla qué entregables producir y cómo desarrollarlos (13). Actualmente en la universidad se está llevando a cabo un ciclo de vida definido como parte del Programa de Mejora (PM), el cual posee fases e iteraciones que sugieren qué hacer, no cómo hacerlo. El uso de esta guía de trabajo permitirá manejar correctamente proyectos a largo plazo, haciendo especial énfasis en la generación de artefactos bien documentados que facilitan la capacitación y transferencia del producto.

Después de realizado el estudio previo correspondiente a la fase de inicio incluida en este ciclo de vida de proyecto, se decide adoptar un modelo guiado por el PM con elementos de RUP para regir el proceso de desarrollo de software de la segunda fase del proyecto.

Rational Unified Process (RUP).

La metodología RUP “Divide en 4 fases el desarrollo del software: Inicio, donde se describe el negocio y se identifican los casos de uso. Elaboración, aquí se define la arquitectura del sistema y se obtiene una aplicación ejecutable. Construcción, donde se obtiene un producto listo para su utilización y Transición donde finalmente se instala el software en condiciones reales. En cada ciclo se produce una nueva versión del sistema y cada versión es un producto preparado para su entrega. El producto terminado satisface todas las necesidades de los usuarios” (14).

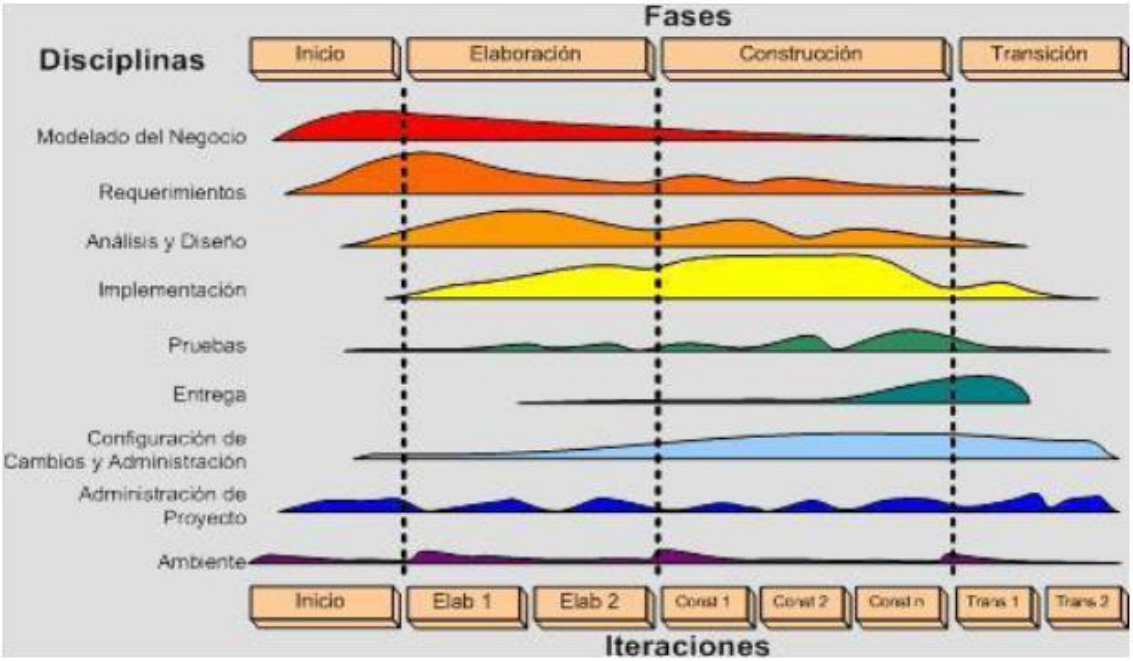


Figura 1. Fases de RUP.

Programa de mejora.

Programa de Mejora (PM) que define el Centro de Calidad para Soluciones Informáticas (CALISOFT) está basado en el modelo CMMI (Integración de Modelos de Madurez de Capacidades), teniendo como el objetivo principal de la UCI lograr una certificación internacional del nivel 2 de este modelo. El cual preverá un crecimiento en cuanto a capacidades y madurez que se enfoca en todos los procesos alcanzando una mejora considerable en el ciclo de vida dentro del desarrollo de software, mayor productividad, presupuestos predecibles en los proyectos productivos y mayor calidad en los productos (15). El uso de esta guía de trabajo, permite manejar correctamente proyectos a largo plazo haciendo especial énfasis en la generación de artefactos bien documentados que facilitan la capacitación y transferencia del producto (16).

Integración de Modelos de Madurez de Capacidades (CMMI).

CMMI es un modelo de madurez de mejora de los procesos para el desarrollo de productos y de servicios. Proporciona las guías para desarrollar los procesos que serán propios para cada organización y que permitirán desarrollar, adquirir y mantener productos o servicios generando múltiples modelos. El

propósito de CMMI es proveer una guía para mejorar los procesos de la organización y su habilidad para administrar el desarrollo, adquisición y mantenimiento de productos y servicios (17).

1.6 Herramientas utilizadas.

Una herramienta es un dispositivo o procedimiento que aumenta la capacidad de llevar a cabo determinadas tareas, por ejemplo herramientas de programación, de gestión, matemáticas, entre otras (18). Las herramientas de la Ingeniería del software proporcionan un enfoque automático o semi-automático para el proceso y para los métodos (14).

Las herramientas CASE (Ingeniería de Software Asistida por Computadoras, del inglés, Computer Aided Software Engineering), son aplicaciones que facilitan el desarrollo de software, reduciendo el esfuerzo, el costo y el tiempo, además de estructurar la documentación asociada a los artefactos generados (19).

Visual Paradigm 8.0.

Visual Paradigm es una herramienta colaborativa, que permite a múltiples usuarios trabajar sobre el mismo proyecto, presenta licencia gratuita y comercial. Es fácil de instalar y actualizar, y compatible entre sus ediciones. Permite generar Diagramas de Procesos del Negocio y Diagramas de Clases, lo cual será muy útil para llevar a cabo la primera etapa del proyecto (20).

Dentro de las características distintivas se pueden mencionar:

- ✓ Uso del lenguaje de modelado unificado (UML por sus siglas en inglés) común para todo el equipo de desarrollo lo que facilita la comunicación, además de ser conocido por los involucrados del módulo.
- ✓ Disponibilidad de integrarse con IDE (Entorno de Desarrollo Integrado o Integrated Development Environment, por sus siglas en inglés) utilizado y el gestor de base de datos.
- ✓ La UCI posee la licencia para su uso y está incluida como propuesta del Programa de Mejora.
- ✓ Disponibilidad en múltiples plataformas, facilitando la participación de todos los miembros del equipo de desarrollo de software.

Por todas las características anteriormente planteadas se escoge esta herramienta para la modelación del módulo Revisión de causas penales.

Unified Model Language (UML) 2.0.

Como lenguaje para el modelado se utilizó UML, ya que permite organizar el proceso de diseño de tal forma que los analistas, clientes, desarrolladores y otras personas involucradas en el desarrollo del sistema lo comprendan. Este lenguaje está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se le conoce como modelo. Un modelo UML describe lo que supuestamente hará el sistema, pero no dice cómo implementarlo (21).

El Lenguaje Unificado de Modelado (Unified Model Language, UML) es un lenguaje gráfico para visualizar, especificar, construir, documentar y comunicar los artefactos de un sistema de software (13). Posee formas de modelar conceptos como son los procesos de negocio y funciones de sistema, además de aspectos concretos como son: escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables.

Se selecciona UML como lenguaje de modelado para el diseño de clases por los elementos antes mencionados.

1.7 Lenguaje de Programación.

Lenguaje del lado del servidor PHP 5.3.

PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica (22).

Ventajas de utilizar PHP para el desarrollo de páginas Web:

- ✓ Es multiplataforma lo que permite desarrollar aplicaciones de software libre.
- ✓ Permite las técnicas de programación orientada a objetos.
- ✓ Manejo de excepciones.
- ✓ Posee documentación en su página oficial la cual incluye descripción y ejemplos de cada una de sus funciones.

Se selecciona este lenguaje por las ventajas antes mencionadas y además porque el marco de trabajo seleccionado tiene como requisito trabajar con la versión 5.3 de este lenguaje.

Lenguajes del lado del cliente.

Un lenguaje del lado del cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio. El código, tanto del hipertexto como de los scripts, es accesible a cualquiera y ello puede afectar a la seguridad (23).

Lenguaje HTML 5.

HTML (HyperText Markup Language) en su versión 5, no sólo trata de incorporar nuevas etiquetas o eliminar otras, sino que supone mejoras en áreas que hasta ahora quedaban fuera del lenguaje y para las que se necesitaba utilizar otras tecnologías (24)

Los cambios comienzan añadiendo semántica y accesibilidad implícitas, especificando cada detalle y borrando cualquier ambigüedad. También cuenta con nuevas API (del inglés Application Programming Interface) para interfaz de usuario: temas tan utilizados como el "drag & drop" (arrastrar y soltar) en las interfaces de usuario de los programas convencionales, serán incorporadas al HTML 5 por medio de un API (25).

Lenguaje JavaScript 1.2.

JavaScript es un lenguaje de scripts desarrollado para incrementar las funcionalidades del lenguaje HTML (26).

Sus características más importantes son:

- ✓ JavaScript es un lenguaje interpretado, es decir, no requiere compilación. El navegador del usuario se encarga de interpretar las sentencias JavaScript contenidas en una página HTML y ejecutarlas adecuadamente.
- ✓ JavaScript es un lenguaje orientado a eventos. Esto le permite al desarrollador web crear efectos dinámicos muy impresionantes mejorando así la experiencia que recibe un usuario momento de entrar a un sitio web (27).

Librería jQuery.

jQuery es una librería de JavaScript rápida, concisa y compatible con múltiples navegadores que se especializa en la interacción entre JavaScript y HTML. Simplifica el trabajo con documentos HTML, manejo de eventos, animación, e interacciones Ajax para el desarrollo rápido de aplicaciones web. JQuery es software open source y está diseñada para cambiar la forma en que se escribe JavaScript (28).

Subversion.

Para el control de versiones según las necesidades del proyecto se usa Subversion, es un software libre bajo una licencia de tipo Apache/BSD y se le conoce también como svn por ser el nombre de la herramienta utilizada en la línea de comando (29).

Brinda la posibilidad de que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones, fomentando la colaboración. Y puesto que el trabajo se encuentra bajo el control de versiones, no hay razón para temer por la calidad del mismo vaya a verse afectada, si se ha hecho un cambio incorrecto a los datos, simplemente se deshace ese cambio.

Entre otras ventajas por las que se selecciona esta herramienta están:

- ✓ Se sigue la historia de los archivos y directorios a través de copias y renombrados.
- ✓ Las modificaciones (incluyendo cambios a varios archivos) son atómicas.
- ✓ Cuando se usa integrado con el servidor Apache permite utilizar todas las opciones que este servidor provee (SQL, LDAP, etc.).

TortoiseSVN.

En el proyecto se trabaja con mucha documentación digital importante, por lo que es de gran necesidad trabajar en base a repositorios de ficheros, se escoge la herramienta TortoiseSVN el cual es un cliente Subversion y es implementado como una extensión al shell de Windows, además es software libre liberado bajo la Licencia Pública General (GPL por sus siglas en inglés) (30).

Entre otras características que brinda por el cual fue seleccionado se encuentran:

- ✓ Inexistencia de copias no controladas.

- ✓ Control del historial de cambios.
- ✓ Permite hacer comparaciones con facilidad.
- ✓ Puede recuperar versiones antiguas con facilidad.

Marco de Trabajo Symfony 2.0.11.

Symfony es un completo marco de trabajo diseñado para optimizar el desarrollo de las aplicaciones web mediante algunas de sus principales características. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación (31).

Symfony incluye uno de los marco de trabajos para el mapeo objeto-relacional (ORM por sus siglas en inglés) más importantes dentro de los existentes para PHP llamado Doctrine, el cual es el encargado de la comunicación con la base de datos, permitiendo un control casi total de los datos sin importar el motor de almacenamiento como MySQL, PostgreSQL, SQL server, Oracle, entre otros (32).

Entre otras características que se conocen se encuentran:

- ✓ Versátil porque está basado en componentes.
- ✓ Útil porque soluciona los retos de la programación web.
- ✓ Buenas prácticas porque incorpora ideas de otros marcos de trabajo.
- ✓ Flexible pues cada programador puede usar lo que desee.
- ✓ Rendimiento pues exige usar como mínimo PHP 5.3, diferentes archivos de configuración que al final se ejecutan en PHP.
- ✓ Amplia documentación.

Además, el gran esfuerzo que han hecho en la seguridad del propio marco de trabajo utilizando técnicas para evitar inyección SQL (lenguaje de consulta estructurado), XSS (del inglés Cross-site scripting) e incluso CSRF (del inglés Cross-site request forgery o falsificación de petición en sitios cruzados) así como la integración de PHP-Unit para la realización de pruebas unitarias.

Es un marco de trabajo muy documentado y se publica bajo licencia MIT22, con la que se puede desarrollar aplicaciones web comerciales, gratuitas y/o de software libre.

1.7 Entorno de desarrollo.

NetBeans 7.3.

NetBeans es un entorno de desarrollo integrado (IDE por sus siglas en inglés), una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. NetBeans es un producto libre, gratuito y sin restricciones de uso (33). Posee un amplio soporte para el lenguaje PHP así como para el marco de trabajo Symfony.

Entre las novedades principales de esta versión destaca mejoras en el soporte de HTML5 y PHP 5.3. Otras características que presenta son:

- ✓ Tiene una sección de edición de HTML5.
- ✓ Netbeans 7.3 está disponible para usuarios de sistemas operativos Windows, Mac, Linux y Solaris desde netbeans.org.

1.8 Gestor de base de datos.

PostgreSQL 9.2.

PostgreSQL 9.2 es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD(Berkeley Software Distribution) (34).

Características de PostgreSQL 9.1:

Ofrece muchas características retirando obstáculos para el despliegue de aplicaciones nuevas o migradas en PostgreSQL, estas incluyen:

- ✓ Replicación Sincrónica, permitiendo alta disponibilidad con consistencia sobre múltiples servidores.
- ✓ Regionalización por columna, soportando correctamente el ordenamiento por lenguaje en las bases de datos, tablas o columnas.

1.9 Patrones de Diseño.

Un patrón de diseño resulta ser una solución a un problema de diseño. Para que una solución sea considerada un patrón debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores (35).

1.9.1 Patrón de Arquitectura.

Un patrón de arquitectura de software es un esquema genérico probado para solucionar un problema particular recurrente que surge en un cierto contexto. Este esquema se especifica describiendo los componentes, con sus responsabilidades, relaciones, y las formas en que colaboran entre sí (35).

Patrón Modelo-Vista-Controlador (MVC).

Es un patrón de arquitectura de software MVC (Modelo-Vista-Controlador) separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Donde el modelo representa la información con la que trabaja la aplicación y se encarga de acceder a los datos, la vista transforma esta información obtenida por el modelo en las páginas web a las que acceden los usuarios y el controlador es el encargado de coordinar todos los demás elementos y transformar las peticiones del usuario en operaciones sobre el modelo y la vista (35).

1.9.2 Patrones Generales de Software para Asignación de Responsabilidades (GRASP).

Los patrones GRASP describen un conjunto de principios para la asignación de responsabilidades. Entre los más significativos se encuentran los siguientes patrones: Experto, Creador, Bajo acoplamiento, Alta cohesión y Controlador.

1.9.3 Patrones GoF (Gand of Four).

Según su propósito los patrones GoF se clasifican en creacionales, estructurales y de comportamiento (36).

El marco de trabajo Symfony2 contiene algunos de estos patrones implícitamente, dándole solución a una serie de problemas recurrentes que se presentan, el mismo implementa numerosas funcionalidades que el cliente debe conocer, para así hacer uso de estos, configurándolo según sus necesidades.

1.10 Conceptualización de las técnicas utilizadas.

Los requisitos permiten conocer las funcionalidades que debe tener el sistema, es por eso que tiene una gran importancia su correcta captura, especificación y validación (14).Las técnicas que a continuación se muestran fueron seleccionadas para aplicarlas durante la etapa de definición de requisitos.

Entrevista: Es una técnica que requiere seleccionar bien a los entrevistados para obtener la mayor información importante en un menor de tiempo. Es muy aceptada y permite acercarse al problema de una manera natural. Abarca cuatro pasos principales: identificar los entrevistados, preparar y realizar la entrevista y documentación de los resultados; en el caso del proyecto se definió la Minuta de Reunión, para reflejar los asuntos tratados, los acuerdos tomados y las partes involucradas en la reunión (37).

Tormentas de ideas: Es una técnica cuyo objetivo es la generación de ideas en un ambiente libre de críticas o juicios mediante reuniones en grupo. Puede ayudar a generar una gran variedad de vistas del problema y formularlo de diferentes formas sin evaluarlas. Suele estar formada por un número de cuatro a diez participantes, uno de los cuales es el moderador de la sesión, encargado más de comenzar la sesión que de controlarla (38).

Prototipos: Algunas propuestas se basan en obtener de la definición de requisitos mediante prototipos, que sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema con el usuario. Esta técnica tiene el problema de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final (39).

1.11 Métricas.

Las métricas a un producto de software son una medida cuantitativa que permiten tener una visión profunda de la eficacia del proceso del software. La medición permite que se mejore el proceso de desarrollo de software, ayuden en la planificación, seguimiento y control de un proyecto de software y evalúen la calidad del producto (14). El empleo de las métricas es una buena manera de conocer lo que sucede en el desarrollo y mantenimiento del sistema, lo que hace posible controlar, predecir y probar el avance del software.

1.11.1 Métricas para Requisitos.

Estabilidad de los requisitos.

Es necesario lograr una estabilidad en los requisitos para el correcto funcionamiento de los demás flujos de trabajo. Se considera que los requisitos son estables cuando no existen adiciones o supresiones en ellos que impliquen modificaciones en las funcionalidades principales de la aplicación (40). Esta métrica ofrece valores entre 0 y 100. El mejor valor de la estabilidad de requisitos (ETR) es el más cercano a 100 porque mostrará que no se están realizando cambios sobre los requisitos, son estables y, por tanto, es confiable trabajar el análisis y diseño sobre ellos (41).

ETR: Valor de la estabilidad de los requisitos.

RT: Total de requisitos definidos.

RM: Número de requisitos modificados, se obtiene a través de la sumatoria de los requisitos insertados, modificados y eliminados.

La estabilidad de los requisitos se calcula como: **ETR = [(RT – RM) / RT] * 100**

1.11.2 Métricas para diseño.

Para medir el diseño se utilizaron métricas básicas inspiradas en el estudio de la calidad del diseño orientado a objeto referenciadas por Pressman, teniendo en cuenta que este estudio brinda un esquema sencillo de implementación y que a la vez cubre los principales atributos de calidad de software. Las métricas escogidas para la validación del diseño fueron las métricas orientada a clases: Tamaño Operacional de Clase (TOC) y Acoplamiento entre Clases (AC) (14).

Tamaño Operacional de Clase (TOC).

El tamaño general de una clase se puede determinar siguiendo los planteamientos descritos a continuación:

El número de atributos (tanto atributos heredados como atributos privados de la instancia) que están encapsulados en la clase.

El número total de operaciones (tanto operaciones heredadas como operaciones privadas de la instancia) que están encapsuladas dentro de la clase.

Estos dos valores son sumados de acuerdo a la clase que se analiza y el resultado es tomado como umbrales que luego son comparados en la tabla que se presenta a continuación para determinar el TOC de cada clase.

Tabla 1. Clasificación de las clases.

Clasificación	Valores de los umbrales
Pequeño	≤20
Medio	>20 ≤30
Grande	>30

Si existen valores grandes de TOC estos estarán demostrando que una clase puede tener demasiada responsabilidad, lo cual reduciría la reutilización de la clase, haciendo complicada la implementación y la prueba. De forma contraria sucede si los valores TOC son de menor valor. Finalmente se calcula los promedios correspondientes a los diferentes valores para tener una estimación general del sistema (14).

Acoplamiento entre clases (AC)

El acoplamiento de una clase es el número de clases a las cuales una clase está ligada. Se da dependencia entre dos clases cuando una clase usa métodos o variables de la otra clase. Las clases relacionadas por herencia no se tienen en cuenta (42).

Las clases deberían ser lo más independientes posible, y aunque siempre es necesaria una dependencia entre clases, cuando esta es grande, la reutilización puede ser más costosa que la reescritura. Al reducir el acoplamiento se reduce la complejidad, se mejora la modularidad y se promueve la encapsulación (42).

1.12 Pruebas de calidad.

Las pruebas del software son un elemento crítico para garantizar de calidad del mismo y representan una revisión final de las especificaciones, del diseño y de la codificación. Pero las pruebas no pueden asegurar la ausencia de errores, solo pueden demostrar que existen defectos en el software (41).

Técnica de pruebas de caja blanca.

La prueba de caja blanca, denominada a veces prueba de caja de cristal o prueba de estructura, se encarga de asegurar que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada. Además, se comprueban los caminos lógicos del software proponiendo casos de prueba que examinen que están correctas todas las condiciones y/o bucles para determinar si el estado real coincide con el esperado o afirmado (41).

Técnica de pruebas de caja negra.

La prueba de caja negra, también denominada prueba de comportamiento se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software (41).

1.13 Conclusiones Parciales.

En este capítulo se realizó un estudio del proceso de revisiones de causas penales en Cuba y de otras soluciones informáticas que se han implantado en otros países, se analizaron las herramientas y lenguajes de modelado que se utilizarán para llevar adelante el proceso de revisiones de causas penales de la FGR, teniendo en cuenta además los patrones y métricas necesarios para lograr un producto con eficiencia y calidad.

Al finalizar este capítulo se arriba a las siguientes conclusiones:

Los conceptos estudiados resultaron de gran importancia para una mejor comprensión del objetivo de este trabajo. El análisis de los SGD permitió concluir que por sus características, ninguno es adaptable a las necesidades que presenta la FGR, por lo que se hace necesario el desarrollo de una aplicación que responda a la informatización de las mismas. Las metodologías, herramientas, tecnologías y lenguajes de programación que serán utilizados son los idóneos para el desarrollo del sistema y se resumen en las siguientes: RUP como metodología de desarrollo guiado por el PM, UML como lenguaje de modelado, Visual Paradigm como herramienta de modelado CASE, PHP como lenguaje de programación. Netbeans

como entorno de desarrollo, Symfony como marco de trabajo, PostgreSQL y Apache como servidores de BD y Web respectivamente.

Los patrones de diseño a emplear, permitirán obtener una solución más eficiente y reutilizable. Las métricas que serán usadas garantizarán la calidad de los resultados que se obtengan en cada una de las disciplinas a desarrollar.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

Introducción

En este capítulo se presentan los requisitos funcionales identificados con el cliente mediante técnicas de tormentas de ideas y la entrevista; aplicando métricas y técnicas de validación para medir la calidad. Se reflejan los requisitos no funcionales definidos para el sistema. Se muestra cómo está concebida la arquitectura y el diseño del software mediante los artefactos generados, haciendo uso de patrones identificados en el proyecto SIGEF II.

2.1 Requisitos.

En un proceso de desarrollo de software es de gran importancia la descripción de los requisitos funcionales, estos facilitan un mejor entendimiento de los procesos a desarrollar, permitiendo comprender con profundidad el problema en cuestión y facilitando una mejor identificación de las clases y funcionalidades que serán implementadas. La especificación de requisitos, es la base que permite verificar si se alcanzaron o no los objetivos establecidos en el proyecto, debido a que son un reflejo detallado de las necesidades de los clientes o usuarios del sistema (41). A continuación se muestra el listado de requisitos definidos para el sistema informático propuesto en la investigación.

2.1.1 Requisitos funcionales.

Los requisitos funcionales son lo que debe hacer y de qué manera debe reaccionar el sistema ante las entradas y cómo debe comportarse en situaciones específicas. Estos detallan la función del producto de software, entrada, salidas, excepciones y usuarios.

Después de aplicar las técnicas de obtención de requisitos mediante las entrevistas con el cliente, a través de la formulación de preguntas de gran importancia sobre lo que se desea como producto final. Se organizaron varias tormentas de ideas por parte del equipo de desarrollo involucrado, para obtener una representación de lo que realmente se desea y lograr un entendimiento común. Como resultado del uso de estas técnicas en el módulo de Revisiones de causas penales se identificaron 69 requisitos funcionales; de ellos 10 son comunes para los diferentes módulos que componen el proyecto SIGEF II; asignados con prioridades alta, media y baja en correspondencia a las necesidades del cliente quedando de la siguiente forma, 36 prioridad alta, 19 prioridad media y 14 prioridad baja.

Especificación de requisitos.

Para la especificación de requisitos se tuvo en cuenta la guía que propone el PM para generar el artefacto Especificación de requisitos. A continuación se presentan los 36 requisitos funcionales de prioridad alta para el cliente. Los restantes requisitos funcionales pueden ser consultados en el documento CEGEL_SIGEFII_0113_ERS_PDC_RCP.

RF_PDC_RCP_1 Adicionar solicitud.
RF_PDC_RCP_9 Generar rollo**.**
RF_PDC_RCP_10 Listar rollo para revisión.
RF_PDC_RCP_ 11 Adicionar causa.
RF_ PDC_RCP_16 Editar persona natural solicitante.
RF_PDC_RCP_18 Adicionar depuración.
RF_PDC_RCP_20 Adicionar traslado de la solicitud**.**
RF_PDC_RCP_22 Adicionar archivo de la solicitud**.**
RF_PDC_RCP_ 24 Adicionar nota.
RF_PDC_RCP_27 Visualizar detalles de la nota.
RF_PDC_RCP_28 Adicionar decisión del fiscal.
RF_PDC_RCP_30 Adicionar dictamen denegatorio**.**
RF_PDC_RCP_32 Adicionar escrito de promoción de revisión**.**
RF_PDC_RCP_34 Adicionar respuesta**.**
RF_PDC_RCP_36 Mostrar vista previa de respuesta al solicitante**.**
RF_PDC_RCP_39 Adicionar cédula de emplazamiento**.**
RF_PDC_RCP_41 Visualizar detalles de la cédula de emplazamiento**.**
RF_PDC_RCP_49 Adicionar sentencia del TSP**.**
RF_PDC_RCP_50 Actualizar sentencia del TSP**.**
RF_PDC_RCP_51 Visualizar detalles de sentencia del TSP**.**
RF_PDC_RCP_52 Listar proceso**.**
RF_PDC_RCP_53 Asociar proceso al rollo**.**
RF_PDC_RCP_54 Adicionar revisión de documento**.**
RF_PDC_RCP_55 Actualizar revisión de documento**.**

RF_PDC_RCP_56 Visualizar detalles del rollo de la fiscalía general**.**
RF_PDC_RCP_57 Visualizar detalles del rollo fiscalía provincial en la fiscalía general**.**
RF_PDC_RCP_58 Visualizar detalles del rollo de la fiscalía provincial**.**
RF_PDC_RCP_59 Buscar rollo de la fiscalía provincial**.**
RF_PDC_RCP_61 Buscar rollo de la fiscalía general**.**
RF_PDC_RCP_62 Buscar documentos**.**
RF_PDC_RCP_63 Mostrar vista previa del dictamen denegatorio**.**
RF_PDC_RCP_63 Mostrar vista previa de escrito de promoción de revisión**.**
RF_PDC_RCP_64 Mostrar vista previa de solicitud de vista**.**
RF_PDC_RCP_65 Mostrar vista previa de práctica de pruebas**.**
RF_PDC_RF_66 Mostrar vista previa de traslado**.**
RF_PDC_RCP_69 Mostrar vista previa de la carátula del rollo**.**

Cada uno de los requisitos funcionales se describe detalladamente, especificando su nombre, breve descripción, complejidad, prioridad para el cliente, prototipo de interfaz de usuario y observaciones. Además se precisa de cada campo involucrado su tipo de datos, reglas y restricciones.

Seguidamente se muestran la descripción de uno de los requisitos funcionales de prioridad alta, que guiarán las labores de implementación de las mismas. Las restantes descripciones pueden ser consultadas en el documento *CEGEL_SIGEFII_0113_ERS_PDC_RCP*.

Tabla 2. RF_PDC_RP_11 Adicionar causa.

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF_PDC_RCP_1	Adicionar causa	Permite adicionar los datos de la causa.	Media	Alta
	Prototipo			

1

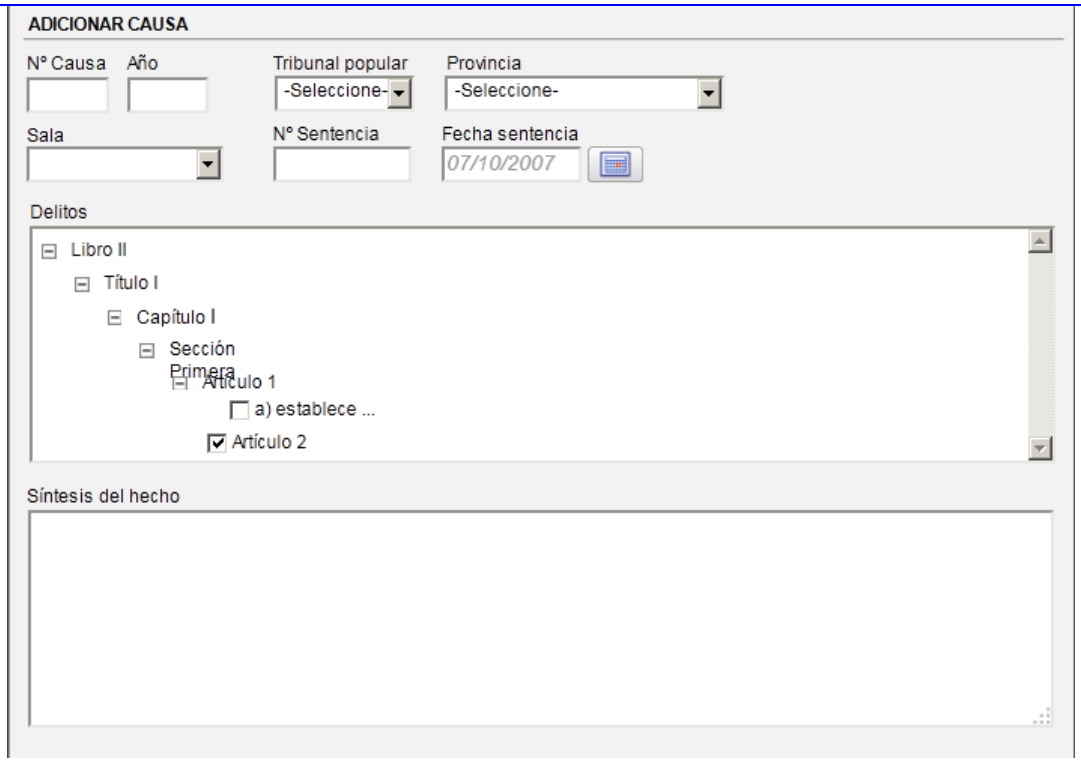


Figura 11.1 Datos de la causa.

Figura 11.2 Datos del tribunal provincial popular.

Figura 11.3 Datos del tribunal municipal popular.

Campos	**Tipos de Datos**	**Reglas o Restricciones**
Nº Causa	Número entero	Obligatorio. Mayor que 0.
Año	Número entero	Obligatorio. Compuesto por cuatro dígitos y mayor que 1900. Igual o menor que el año actual.
Tribunal popular	Cadena	Obligatorio. Solo letras. Los valores de este campo son: • Municipal

		• Provincial
Sala	Cadena	Obligatorio. Los valores que toma este campo se evidencian en el documento: NOMENCLADORES_PDC_RCP.
Nº Sentencia	Número entero	Obligatorio. Mayor que 0.
Fecha de sentencia	Fecha	Obligatorio. Menor que la fecha de registro de la solicitud.
Delitos	Cadena	Obligatorio. Los valores que toma este campo se evidencian en el documento: NOMENCLADORES_PDC_RCP.
Síntesis del hecho	Cadena	Opcional.
Figura 11.2 y **Figura 11.3**		
Provincia	Cadena	Obligatorio. Los valores que toma este campo se evidencian en el documento: NOMENCLADORES_PDC_RCP.
Figura 11.3		
Municipio	Cadena	Obligatorio. Los valores que toma este campo se evidencian en el documento: NOMENCLADORES_PDC_RCP.
Observaciones	N/A	

2.1.2 Requisitos no funcionales.

A continuación se muestran algunos de los requisitos no funcionales obtenidos, con una breve descripción:

Usabilidad:

RnF 1. La aplicación informática a desarrollar será una aplicación web.

RnF 2. La aplicación informática a desarrollar será un software de gestión.

RnF 3. Entorno de producción de la aplicación informática a desarrollar.

Confiabilidad:
RnF 4. Fiabilidad del sistema. Existirán servidores locales con capacidad necesaria para el procesamiento de las solicitudes del conjunto de aplicaciones de las diferentes oficinas.

RnF 5. Disponibilidad del sistema. El sistema estará disponible durante el horario laboral, efectuándose en períodos de tiempos definidos el proceso de actualización de la información del servidor local de base de datos con el Centro de Datos.

RnF 6. Exactitud de las respuestas. Las salidas que ofrece el sistema como resultado del manejo y procesamiento de la información estarán adecuadamente validadas durante los hitos de prueba.

Eficiencia:
RnF 7. Tiempo de respuesta. La mayoría de los procesos que se implementan con transacciones donde se modifica la base de datos deben tener tiempos de respuesta no mayores a los 3 segundos

Soporte:
RnF 8. Reparabilidad, mantenibilidad, escalabilidad. Para garantizar estas características se desarrollará el sistema orientado a componentes.

Interfaz:
RnF 9. Interfaz de usuario. El sistema tiene que ofrecer una interfaz amigable y debe ostentar un diseño sencillo, con pocas entradas, permitiendo un balance adecuado entre funcionalidad y simplicidad.

RnF 10. Interfaz de hardware. Las estaciones de trabajo pueden acceder a impresoras dependiendo de la funcionalidad establecida.

Seguridad:

RnF 11. Autenticación. Se establecerán mecanismos 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.

RnF 12. Autorización. Existirán diferentes tipos de usuarios según las acciones que puedan realizar. Los permisos serán limitados basados en los roles definidos y el usuario no podrá gestionarlos.

RnF 13. Confidencialidad e integridad de los datos. Garantizar el uso de servicios de red, puertos y protocolos compatibilizados con las políticas de seguridad trazados por la entidad que soportará el despliegue de la aplicación.

Se identificaron un total de 19 requisitos no funcionales. Para acceder a la especificación de los mismos ver el documento *0113_Especificación de Requisitos de Software SIGEF II.*

2.2 Validación de requisitos.

Dentro de la ingeniería de requisitos fue realizada la validación de los mismos a través de las siguientes técnicas:

Revisiones técnicas formales.

Se ejecutaron un conjunto de reuniones con el personal técnico del proyecto, donde se descubrieron omisiones que habían sido cometidas en el proceso de obtención de los requisitos. También se realizó una revisión por el equipo de calidad donde resultaron 36 no conformidades en una primera iteración, las cuales fueron corregidas en una segunda revisión.

Validación con el cliente.

La validación con los clientes fue mediante los prototipos de interfaz de usuario, proporcionando una visión más clara al cliente de las funcionalidades del sistema. Se revisaron cada uno de los prototipos con el fiscal que atiende el tema de las revisiones en la FGR y se corrigieron los errores.

Aplicación de la métrica Estabilidad en los Requisitos.

Es necesario garantizar que los requisitos sean estables, para esto se empleó la métrica estabilidad en los requisitos, en una primera iteración en la cual se calculó de la siguiente manera:

Teniendo en cuenta que se identificaron un total de 69 RF, dentro de los cuales 12 resultaron modificados (RM), se calcula:

ETR = [(69 – 12) / 69] * 100 = 82.6

Luego de aplicar esta métrica, se obtuvo como resultado que los requisitos no fueron lo suficientemente estables. Luego de perfeccionado todos los señalamientos o sugerencias se realizaron nuevamente las validaciones alcanzándose un 95.3% siendo un resultado favorable.

2.3 Diseño.

El objetivo principal de la fase Diseño es modelar el sistema de manera que cumpla con los requisitos de la fase anterior sirviendo como punto de partida para la implementación. En esta fase se tienen en cuenta los requisitos no funcionales: restricciones del lenguaje de programación a emplear, el tipo de interfaz, entre otros definidos por el proyecto SIGEF II. El diseño del sistema es una base importante en la elaboración del software, el cual permite que se produzcan los diversos modelos para la solución que se pondrá en desarrollo. El cual debe ser apto para facilitar las mejoras del software, además ser entendible por otros profesionales de la especialidad, de manera que permita la comprobación y mantenimiento del sistema fácilmente.

2.3.1 Patrón de Arquitectura.

El desarrollo de SIGEF II está respaldado por el patrón de arquitectura modelo vista controlador (MVC). El marco de trabajo Symfony2 cumple con la arquitectura definida en el proyecto, la cual está basada en este patrón de arquitectura; el mismo separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. A continuación se demostrará durante la descripción de las clases cómo estas quedan distribuidas dentro de la arquitectura, teniendo funciones diferentes pero relacionadas.

Figura 2. Representación de la arquitectura MVC SIGEF II.

El modelo está dividido en tres capas, la capa de acceso a datos, la capa negocio y la capa de abstracción de datos. La capa de acceso a datos es una porción de código que justamente realiza el acceso a los datos. De esta manera cuando es necesario cambiar el motor de base de datos, solo se tendría que corregir esa capa, donde se encuentran las clases Repository. Una de las partes más importantes de Symfony, es la que define la capa de abstracción de bases de datos. La cual permite cambiar de base de datos en cualquier momento de forma sencilla y la aplicación sigue funcionando perfectamente. Esta contiene las clases entidades mapeadas por el ORM que se muestran en la figura 3.

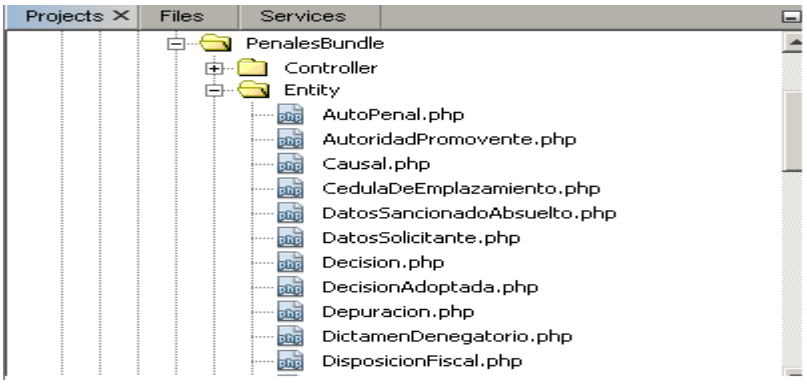


Figura 3. Clases de la capa del modelo.

La vista se encarga de que el usuario interactúe con el sistema, muestra y obtiene la información del usuario. En esta capa se encuentran las vistas o interfaces, agregando imágenes, archivos CSS y

JavaScript. En la aplicación estas vistas o interfaces gráficas se manejan con la utilización del motor de plantillas Twig, que transforman los datos obtenidos del modelo, necesarios para la interacción con el cliente. A continuación se muestra algunas clases contenidas en la capa vista.

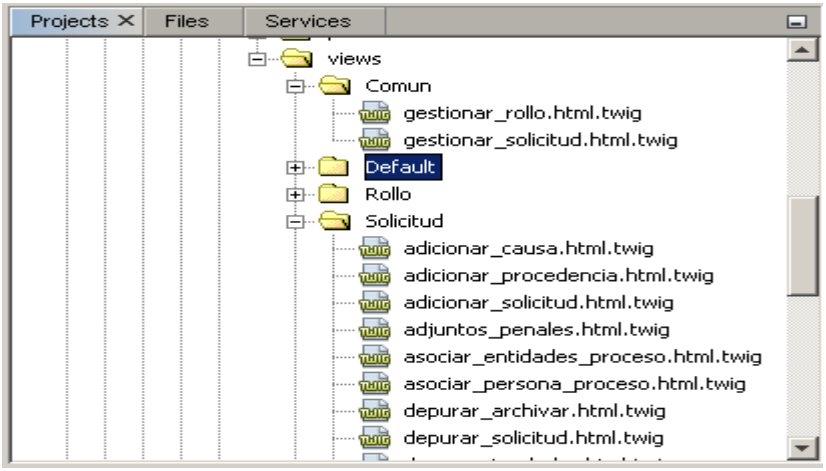


Figura 4. Clases de la capa vista.

El controlador tiene como tarea obtener los datos del modelo y pasárselos a la vista, se encarga además de procesar las peticiones. Para realizar este proceso el controlador hace uso del modelo para buscar información en la base de datos. En la figura 5 se muestra algunas clases pertenecientes a esta capa.

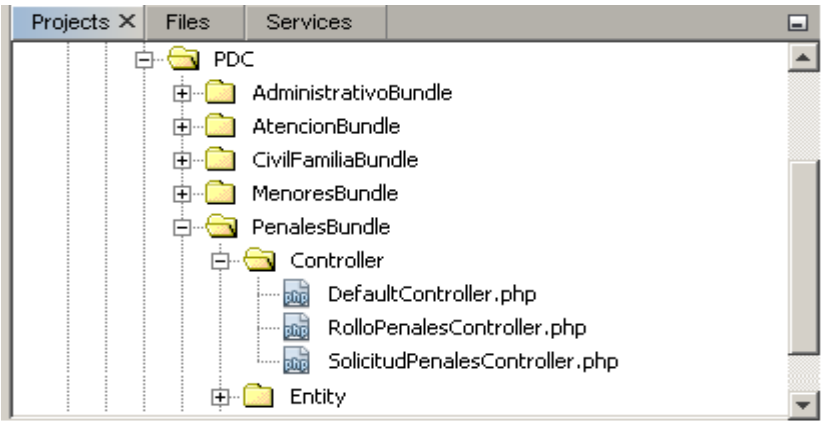


Figura 5. Clases de la capa controlador.

2.3.2 Patrones de diseño.

Patrones GRASP aplicados.

Experto: Indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo (35). El uso principal de este patrón se encuentra en clases repositorio, que son expertas en realizar operaciones de acceso a datos solo con la entidad que representa.

Creador: El patrón creador ayuda a identificar quién debe ser el responsable de la creación de nuevos objetos (35). En la solución propuesta fue utilizado el patrón en las clases gestora SolicitudPenalesGtr para crear las entidades y la clase controladora SolicitudPenalesController para crear los formularios (ver figura 6).

```
337   public function salvarActualizarDepuracionAction($grupo) {
338
339       $peticion = $this->getRequest();
340       $depuracion = new Depuracion();
341       $frm = $this->createForm(new DepurarSolicitudType(), $depuracion);
342
```

Figura 6. Ejemplo de del patrón creador de la clase SolicitudPenalesController.

Controlador: El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado (35). Este patrón se puede observar por lo general en las clases controladoras SolicitudPenalesController (ver figura 7), RolloPenalesController y en el controlador frontal de Symfony: siendo este el único punto de entrada a la aplicación en un entorno determinado.

```
if ($peticion->getMethod() == 'POST') {
    $frm->bindRequest($peticion);
    $this->getSolicitudPenalesGtr()->salvarActualizarSolicitud($solicitud
    return $this->render('PenalesBundle:Solicitud:depurar_archivar.html.t
```

Figura 7. Ejemplo de del patrón controlador de la clase SolicitudPenalesController.

Alta cohesión: Garantiza que la información que almacena una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la clase (35). Symfony permite asignar responsabilidades con una alta cohesión. Se pone de manifiesto en las clases repositorio, que tienen como única responsabilidad de realizar operaciones de acceso a datos solo con la entidad que representa; esto hace posible que el software sea flexible a cambios sustanciales sin afectarlo de manera transcendental.

Bajo acoplamiento: Este patrón tiene como idea, tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases (35)**.** Este patrón está evidenciado en el marco de trabajo ya que dentro de la capa modelo las clases de abstracción de datos son las más reutilizables y no tienen asociaciones con las clases de la capa vista ni con el controlador.

Patrones GoF aplicados.

El marco de trabajo Symfony2 contiene algunos de estos patrones implícitamente, dándole solución a una serie de problemas recurrentes que se presentan, además ayudan a construir un software basado en la reutilización. A continuación se describen varios de los patrones GoF utilizados para la implementación del sistema informático propuesto en la investigación.

Creacionales:

Fábrica (en el inglés Factory): Esta clase tiene entre sus responsabilidades la creación de instancias de objetos. Los contenedores de servicios de Symfony2 proporcionan una forma eficaz de controlar la creación de objetos, lo cual te permite especificar los argumentos pasados al constructor, así como llamar a los métodos y establecer parámetros. A continuación se muestra un ejemplo del uso del patrón fábrica que se encuentra dentro del archivo services.yml.

```
services:
    SolicitudPenalesGtr:
        class: PDC\PenalesBundle\Negocio\SolicitudPenalesGtr
        parent: ProcesoGtr
    RolloPenalesGtr:
        class: PDC\PenalesBundle\Negocio\RolloPenalesGtr
        parent: ProcesoGtr
```

Figura 8. Fragmento de código del fichero services.yml.

Estructurales:

Decorador (en el inglés Decorator): El patrón decorador responde a la necesidad de añadir dinámicamente funcionalidad a un objeto. Aplicado a la generación de vistas la solución que ofrece dicho patrón es la de añadir funcionalidad adicional a las plantillas. Por ejemplo, añadir el menú y el pie de página a las plantillas que lo requieran, de manera que dichos elementos puedan reutilizarse en distintas plantillas. Literalmente se trata de decorar las plantillas con elementos adicionales reutilizables (ver figura 9).

Figura 9. Uso del patrón decorador en la herencia de plantillas Twig.

Comportamiento:

Observador (en el inglés Observer): Es un patrón que define una dependencia entre objetos, de manera que cuando uno de los objetos cambia su estado, notifica este cambio a todos los dependientes (43). El despachador de eventos de Symfony2 (ver figura 10) implementa el patrón observador en una manera sencilla y efectiva. Una vez creado un objeto Respuesta, puede ser útil permitir que otros elementos en el sistema lo modifiquen (por ejemplo, añadan algunas cabeceras de caché) antes de utilizarlo realmente. Para hacer esto posible, el núcleo de Symfony2 lanza un evento —kernel.response—.

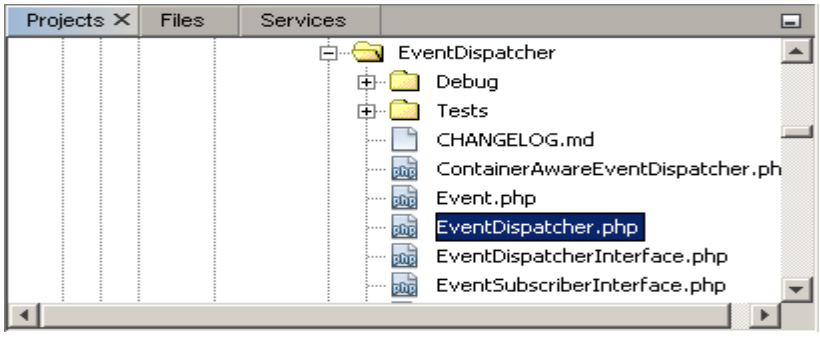


Figura 10. Paquete despachador de eventos de Symfony2.

Diagrama de clases persistentes.

Se realizó el diagrama de clases persistentes a partir del diagrama de clases del diseño, para representar las clases, atributos con sus relaciones y funcionalidades que el sistema debe ofrecer, para crear un diseño conceptual de lo que tratará el sistema, brindando información importante. A continuación se muestra una parte del diagrama de clases persistentes referente a la funcionalidad adicionar solicitud.

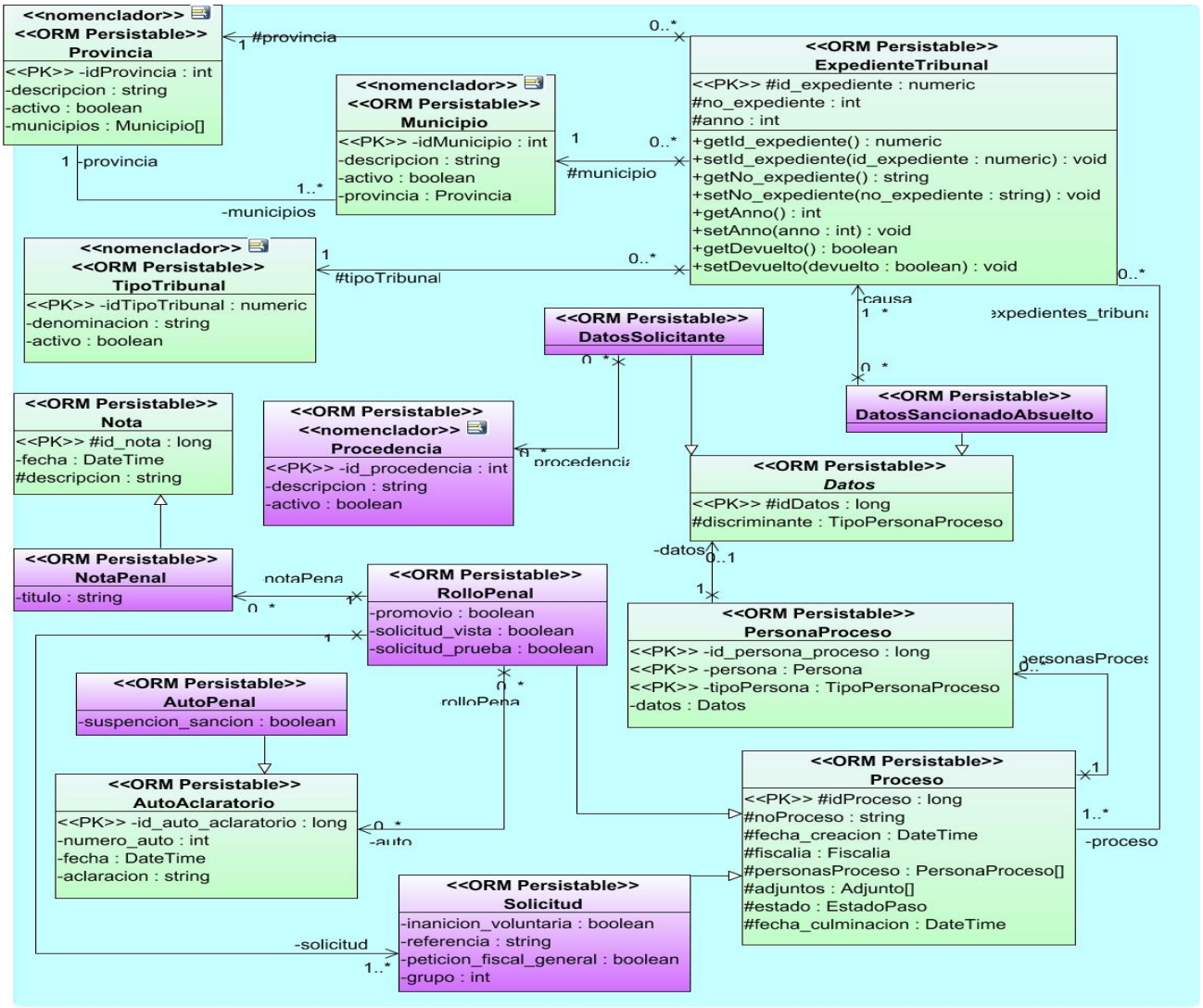


Figura 11. Diagrama de clases persistentes para la funcionalidad adicionar solicitud.

En el diagrama se representan las clases diferenciadas por dos colores. Las de color verde pertenecen al subsistema base, comunes para todos los módulos del proyecto como por ejemplo: Proceso, Documento, ExpedienteTribunal, Provincia y Municipio. Las clases pertenecientes a las funcionalidades del módulo se representan de color violeta. Se obtuvieron 4 clases nomencladores definidas por las particularidades del negocio.

Diagrama de secuencia.

El diagrama que se presenta muestra el comportamiento de un conjunto de objetos en la aplicación a través del tiempo, este contiene detalles de implementación, incluyendo los objetos y clases como son las vistas, controladoras, gestores, repositorios que se usan para implementar el escenario y mensajes intercambiados entre los objetos. A continuación se muestra el diagrama de secuencia para la funcionalidad “adicionar/actualizar causa”.

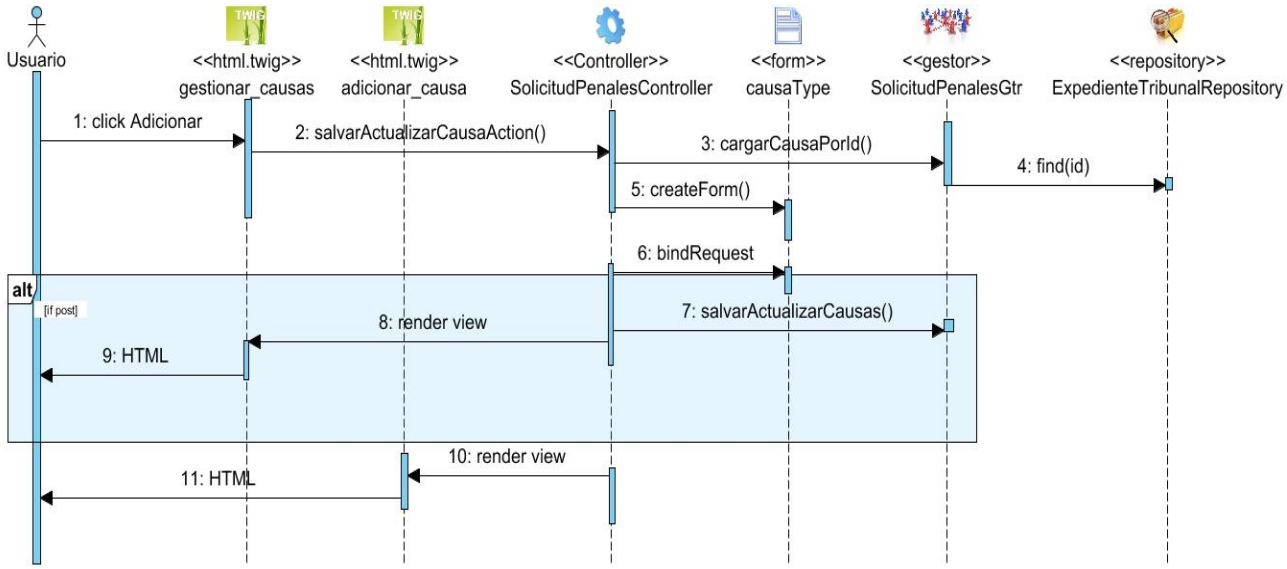


Figura 12. Diagrama de secuencia para la funcionalidad adicionar/actualizar causa.

Diagrama de clases de la vista.

El diagrama de la vista (ver figura 13) muestra cómo se relacionan las clases Twig con los distintos formularios y las clases JavaScript en la funcionalidad adicionar solicitud. Este diagrama tiene diez clases

Twig, de estas, dos son parte del subsistema base. También componen este diagrama cuatro formularios y dos archivos JavaScript.

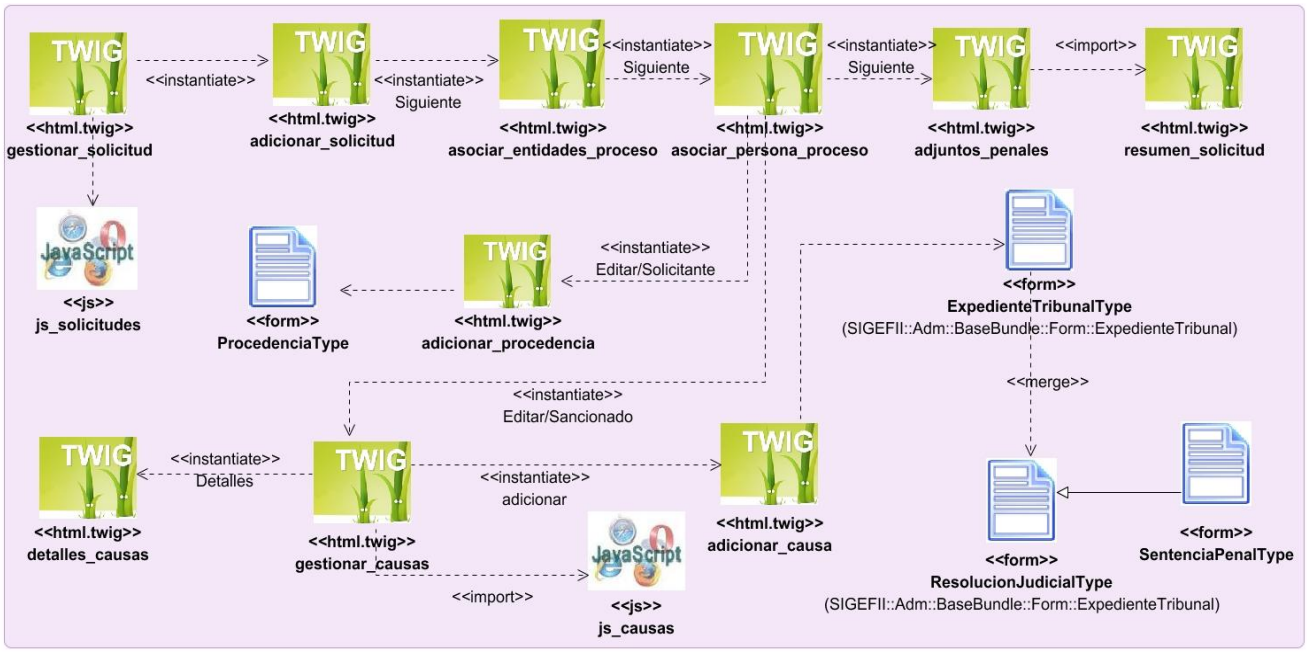


Figura 13. Diagrama de clases de la vista adicionar solicitud.

Diagrama de clases de repositorio.

La imagen (ver figura 14) representa el diagrama de Repositorios específicamente de la funcionalidad Adicionar Solicitud del módulo Revisión de causas penales en el cual se muestran las clases SolicitudRepositoriy y PersonaRepository, esta última perteneciente al subsistema base. Estas clases son las encargadas de interactuar con la base de datos.

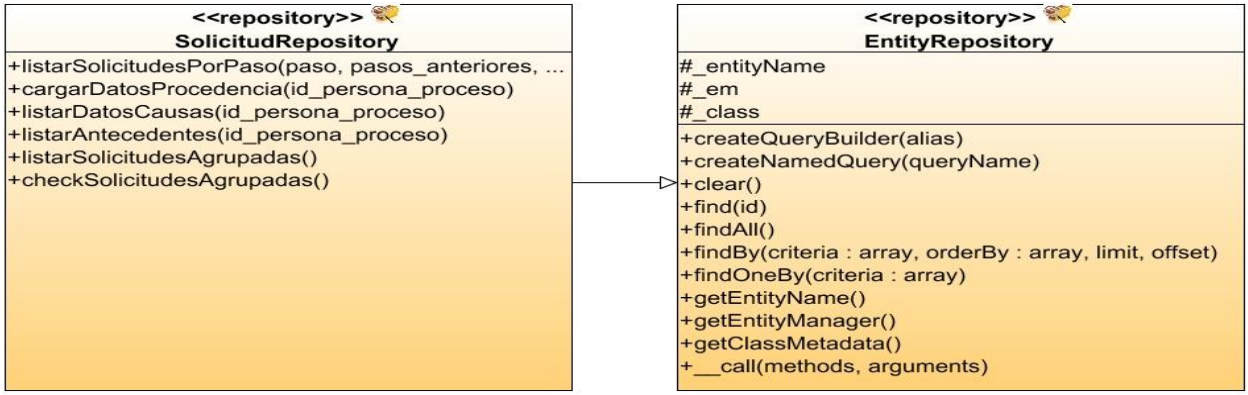


Figura 14. Diagrama de clases de repositorio.

Diagrama de clases de controladores.

El diagrama de los controladores (ver figura 15) muestra la relación entre las clases ProcesoController, RolloPenalesController y RolloPenalesGtr, de las cuales ProcesosController pertenecen al subsistema base del SIGEF II. Estas se encargan de manejar el flujo de datos y la gestión del negocio.

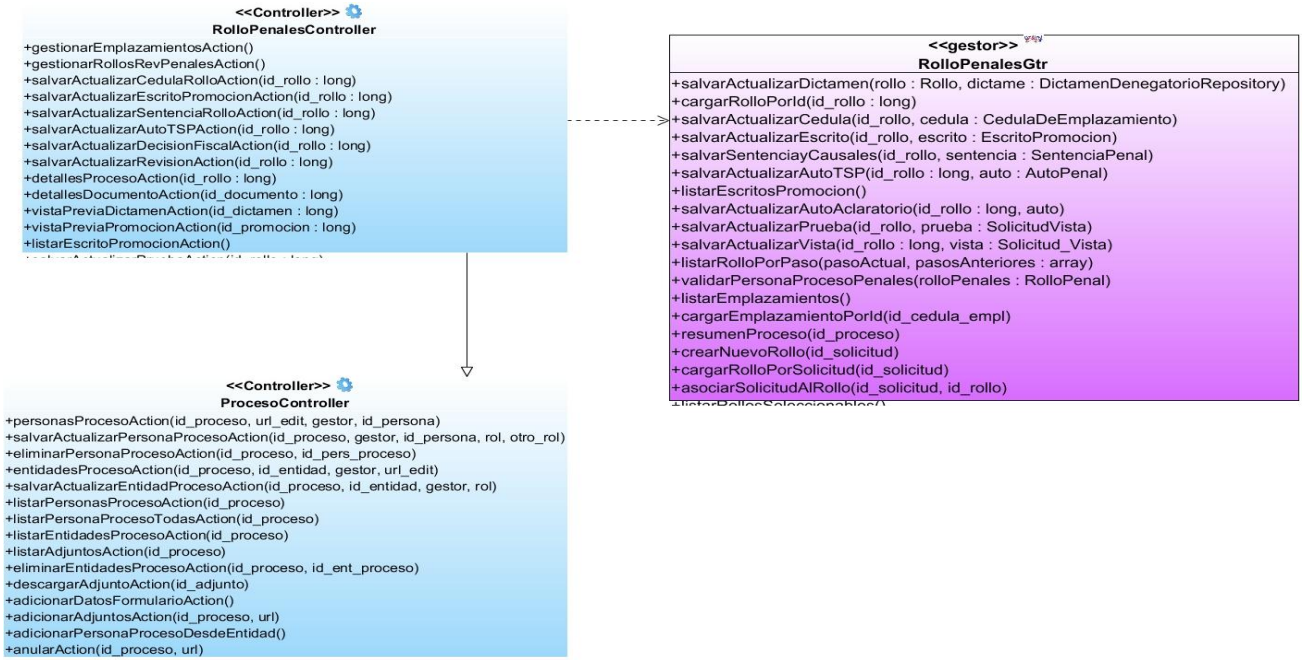


Figura 15. Diagrama de clases de controladores.

2.4 Validación del diseño del proceso.

Métrica Tamaño Operacional de Clase (TOC):

La Métrica Tamaño Operacional de Clase se aplicó para las clases obtenidas en esta etapa, donde se utilizaron un total de 47 clases para un promedio de 3.8 atributos por clases y de 6.8 de operaciones por clases.

Luego fue necesaria una evaluación concreta de la métrica mediante los umbrales definidos por la misma (ver figura 16). Se tuvo en cuenta el tamaño de las clases, esta se calcula sumando la cantidad de atributos y la cantidad de operaciones, comparando luego con los valores de los umbrales para clasificar las clases en pequeña, media o grande.

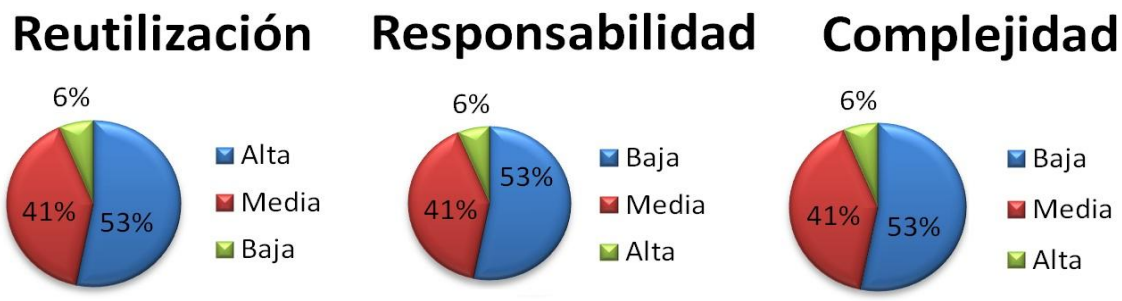


Figura 16. Resultados de la Métrica Tamaño de Clases de responsabilidad y complejidad para 47 clases.

Se alcanzó como resultado un 53.1 % de las clases en la categoría Baja, un 40.3 % Media y sólo un 5.5 % Alta. Con estos indicadores se demuestra que el tamaño de las clases en su mayor porciento posee valores pequeños. Esto evidencia que las clases no tienen grandes responsabilidades y posibilitan la reutilización, teniendo en cuenta que más de la mitad de las clases poseen baja dependencia respecto a otras, evidenciando que el sistema no tendrá una compleja implementación. Por estas razones el resultado de la aplicación de esta métrica resultó positivo.

Métrica Acoplamiento entre Clases (AC):

La métrica Acoplamiento entre Clases fue empleada para medir los niveles de responsabilidad, acoplamiento, complejidad de mantenimiento y reutilización de las clases del diseño. Para cada uno de

estos niveles se definen los siguientes criterios y categorías de evaluación:

Tabla 3. Criterios de evaluación de la métrica AC.

Atributo	Categoría	Criterio
Acoplamiento.	Ninguno.	0
	Bajo.	1
	Medio.	2
	Alto.	>2
Complejidad de mantenimiento.	Baja.	≤Promedio
	Media.	Entre Promedio y 2*Promedio
	Alta.	>2*Promedio
Reutilización.	Baja.	>2*Promedio
	Media.	Entre Promedio y 2*Promedio
	Alta.	≤Promedio
Cantidad de pruebas.	Baja.	≤Promedio
	Media.	Entre Promedio y 2*Promedio
	Alta.	>2*Promedio

Esta métrica fue aplicada a un total de 47 clases con un promedio de asociaciones de uso de 1.1.

Tabla 4. Cantidad de dependencias por clasificación.

Criterio	Categoría	Cantidad de clases	Porciento
0 dependencias	Muy Bueno	30	63,82978723
1 dependencias	Bueno	8	17,0212766
2 dependencias	Regular	2	4,255319149
3 dependencias	Malo	2	4,255319149
> 3 dependencias	Muy Malo	5	10,63829787
Total		47	100

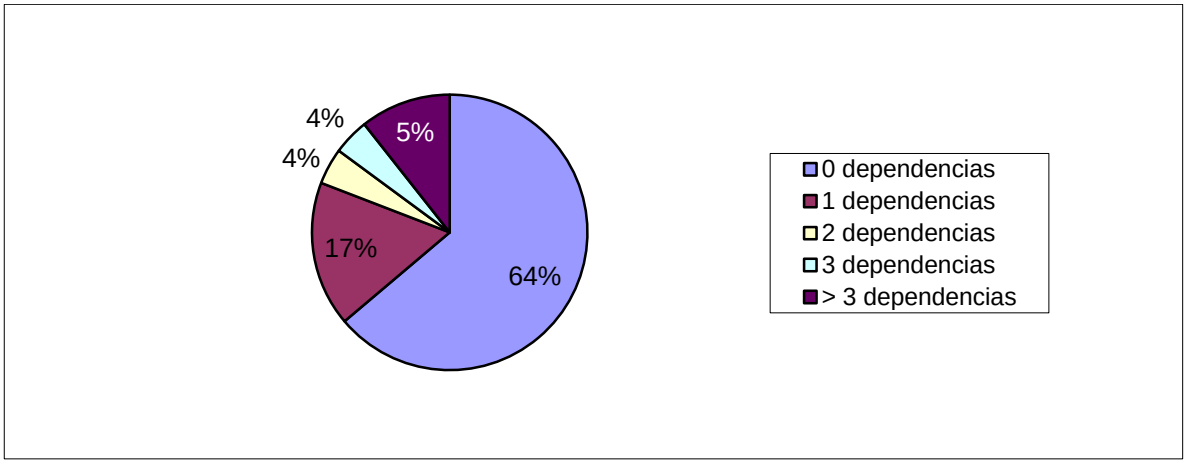


Figura 17. Comportamiento de los valores de dependencia.

Tabla 5. Resultados de los atributos de calidad de la métrica AC.

Cantidad de clases 47	**Ninguno**	**Baja**	**Media**	**Alta**
Acoplamiento	73%	20%	5%	2%
Complejidad de mantenimiento		81%	6%	13%
Cantidad de pruebas		81%	6%	13%
Reutilización		15%	6%	79%

La aplicación de la métrica AC arrojó resultados satisfactorios, evidenciando que el diseño realizado se encuentra en un estado aceptable. Entre los resultados obtenidos se nota un bajo acoplamiento, lo que hace posible la reutilización y facilita la comprensibilidad, haciendo fácil el mantenimiento. La complejidad de mantenimiento se muestra en valores pequeños, no se requieren grandes cantidades de pruebas y la reutilización se comporta favorablemente para un 79%.

2.5 Conclusiones parciales.

En este capítulo se aplicaron las técnicas de la ingeniería de requisitos: tormenta de ideas y la entrevista al módulo de Revisiones de causas penales, ampliando la visión de lo que debe realizar el sistema. Esto permitió identificar, analizar, especificar y validar los requisitos funcionales, así como dar a conocer los no funcionales.

Se obtuvieron varios artefactos del modelo del diseño como el diagrama de clases persistentes, diagrama de vistas, diagrama de clases repositorio, diagrama de clases controladores y el diagrama de secuencia utilizando la herramienta Visual Paradigm, brindando una mejor comprensión al equipo encargado del módulo; sirviendo como punto de partida a las actividades de implementación. La aplicación de patrones de diseño favoreció la obtención de artefactos reutilizables.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

Introducción

En este capítulo se describen elementos importantes en la implementación y prueba de la solución, teniendo como objetivo principal desarrollar la arquitectura y el sistema como un todo. Se hace referencia al modelo de implementación, formado por los diagramas de componentes y de despliegue. Además se comprueba la calidad del resultado de la implementación, mediante los artefactos elaborados durante el flujo de trabajo de pruebas, mostrando los resultados obtenidos al aplicarlas al módulo.

3.1 Modelo de Implementación.

El modelo de implementación está compuesto por los Diagramas de Componentes y de Despliegue, describiendo cómo los elementos del Modelo de Diseño se implementan en términos de componentes, entre los que se encuentran datos, archivos, ejecutables y código. Este artefacto describe cómo se implementan los componentes, juntándolos en subsistemas organizados en capas y jerarquías, y señala las dependencias entre estos (44).

3.1.2 Diagrama de Componentes.

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre sus elementos. El uso más importante de estos diagramas es expresar la forma en que un sistema de software se divide en componentes, visualizando las dependencias entre dichos componentes (21). A continuación se muestra el Diagrama de Componentes para el sistema.

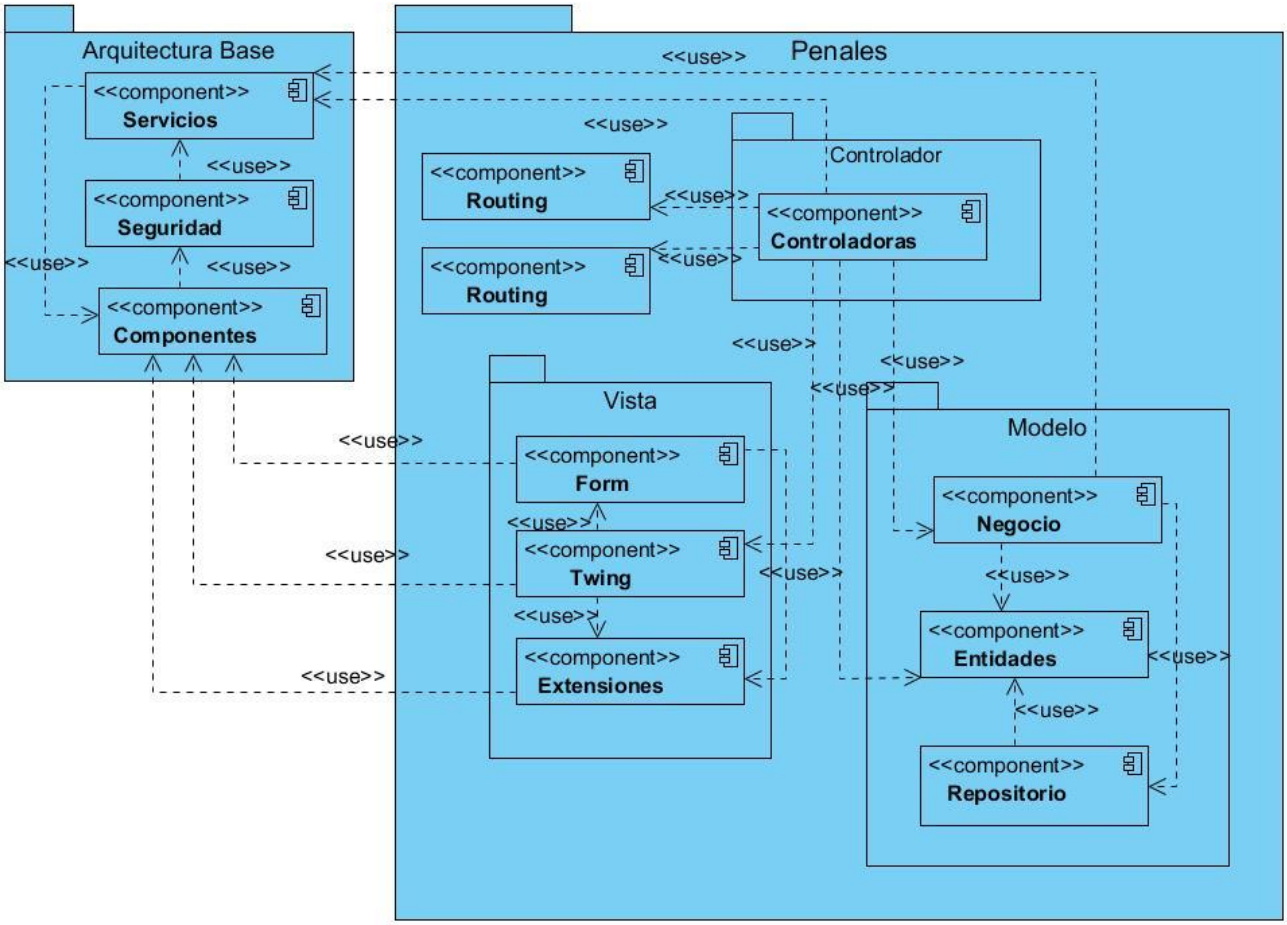


Figura 18. Diagrama de componentes del módulo Revisiones de causas penales.

El diagrama representa la relación entre los componentes de la Arquitectura Base y el módulo de Penales, agrupadas por paquetes para una mayor comprensión.

En el paquete Arquitectura Base se encuentran los componentes servicios, seguridad y componentes. En el componente Seguridad se gestionan los roles y usuarios de la aplicación, así como la asignación de permisos. Otra forma de gestionar la seguridad es a través del archivo security.yml que se muestra a continuación.

```
security:
    acl:
        connection: default
    role_hierarchy:
        ROLE_ADMIN:       ROLE_USERIO
        ROLE_SUPER_ADMIN: [ROLE_USERIO, ROLE_ADMIN, ROLE_ALLOWED_TO_SWITCH]
    firewalls:
        dev:
            pattern:  ^/(_(profiler|wdt)|css|images|js)/
            security: false
        area_segura:
            remember_me:
                key:      "%secret%"
                lifetime: 31536000 # 365 days in seconds
            pattern:    ^/
            anonymous: ~
            form_login:
                check_path: /sigef2/login_check
                login_path: /sigef2/login
                always_use_default_target_path: true
                default_target_path: /sigef2/principal
            logout:
                path:       /sigef2/logout
                target:     /sigef2/login
    access_control:
        - { path: ^/sigef2/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
        - { path: ^/sigef2, roles: ROLE_ADMIN }
    providers:
        usuarios:
            entity: { class: SeguridadBundle:Usuario, property: username }
```

Figura 19. Código fuente del archivo security.yml.

El componente Controladoras lo integran las clases SolicitudPenalesController y RolloPenalesController, que son las que generan las vistas. El componente Negocio, constituida por la clase SolicitudPenalesGtr y RolloPenalesGtr que encierran la lógica del negocio.

El componente Entidades abarca las clases entidades y repositorio, contiene las clases que interactúan con la base de datos. A continuación se muestra un método de la clase SolicitudRepository, encargada de hacer consulta a la base de datos.

```
18      public function listarSolicitudes($id_paso) {
19          $qb = $this->getEntityManager()->createQueryBuilder();
20          $qb->select('s')
21                  ->from('PenalesBundle:Solicitud', 's')
22                  ->innerJoin('s.estado', 'ep')
23                  ->innerJoin('ep.paso', 'p')
24                  ->innerJoin('ep.estado', 'e')
25                  ->where($qb->expr()->eq('p.id_paso', ':prmPaso'))
26                  ->andWhere($qb->expr()->neq('e.id_estado', ConfigUtil::estado_finalizado))
27                  ->andWhere($qb->expr()->neq('e.id_estado', ConfigUtil::estado_anulado))
28                  ->andWhere($qb->expr()->neq('e.id_estado', ConfigUtil::estado_concluido))
29                  ->setParameter('prmPaso', $id_paso);
30
31          $result = $qb->getQuery()->getResult();
32          return $result;
33      }
```

Figura 20. Método listar solicitudes de la clase SolicitudRepository.

Dentro del paquete Vista, se encuentran los componentes Form, los cuales contienen los formularios y Twig como motor de plantillas de interfaces de usuario, estos son usados por los controladores los cuales se encargan de crearlos como se muestra en la figura.

```
//Proceso de creación del formulario
$frm = $this->createForm(new SolicitudType(), $solicitud,array('sin_entidad' => $sin_entidad));

//mostrando la página twig con el formulario
return $this->render('PenalesBundle:Solicitud:adicionar_solicitud.html.twig',
array('frm' => $frm->createView(),'id_solicitud' => $solicitud->getIdProceso()));
```

Figura 21. Construcción de formularios y vista desde el controlador.

El componente Extensiones encierra los archivos JavaScript y CSS. La siguiente figura representa como se incluyen estos en las páginas Twig. Se muestra cómo se reutizan funciones JavaScript para validar el texto.

```
{% block javascripts %}
<script type="text/javascript" src={{ asset('bundles/componentes/js/
componentesVista/jquery.validarEntradaTexto.js')}}></script>
<script type="text/javascript" src={{ asset('bundles/componentes/js/errores/
validaciones_JavaScript.js')}}></script>
{% endblock %}
```

Figura 22. Bloque de JavaScripts de la página adicionar_solicitud.html.

El componente Routing, gestiona las rutas para los diferentes flujos del módulo. Cuando se hace una petición a la aplicación, esta contiene una dirección al recurso exacto que solicitó el cliente. El objetivo del sistema de rutas de Symfony2 es analizar esta dirección y determina qué controlador se debe ejecutar. En la figura 23 muestra una parte del fichero routing.yml del módulo.

```
rcp_salvar_actualizar_solicitud:
    pattern:  /salvar_actualizar_solicitud/{id_solicitud}
    defaults: { _controller: PenalesBundle:SolicitudPenales:salvarActualizarSolicitud,
    id_solicitud null}
    requirements:
        id_solicitud: \d+
```

Figura 23. Fragmento de código del fichero routing.yml.

El componente Services contiene los servicios, estos son cualquier objeto PHP que realiza algún tipo de tarea global. Para que funcionen, se debe configurar el contenedor de servicios de manera que sepa cómo crear los objetos. Los servicios son utilizados por las clases controladoras para el manejo de las clases del componente negocio como se muestra en la figura 24.

```
    public function salvarActualizarSolicitudAction($id_solicitud ){
      $peticion = $this->getRequest();
      $solicitud = $this->getSolicitudPenalesGtr()->cargarSolicitudPorId($id_solicitud);
```

Figura 24. Fragmento de código de la clase SolicitudPenalesController.

3.1.2 Modelo de despliegue

El modelo de despliegue es un modelo de objetos que representa la distribución física del sistema. El diagrama de despliegue es empleado para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Muestra las relaciones físicas de los distintos nodos que componen un sistema (45).

A continuación se expone el diagrama de despliegue de la solución propuesta:

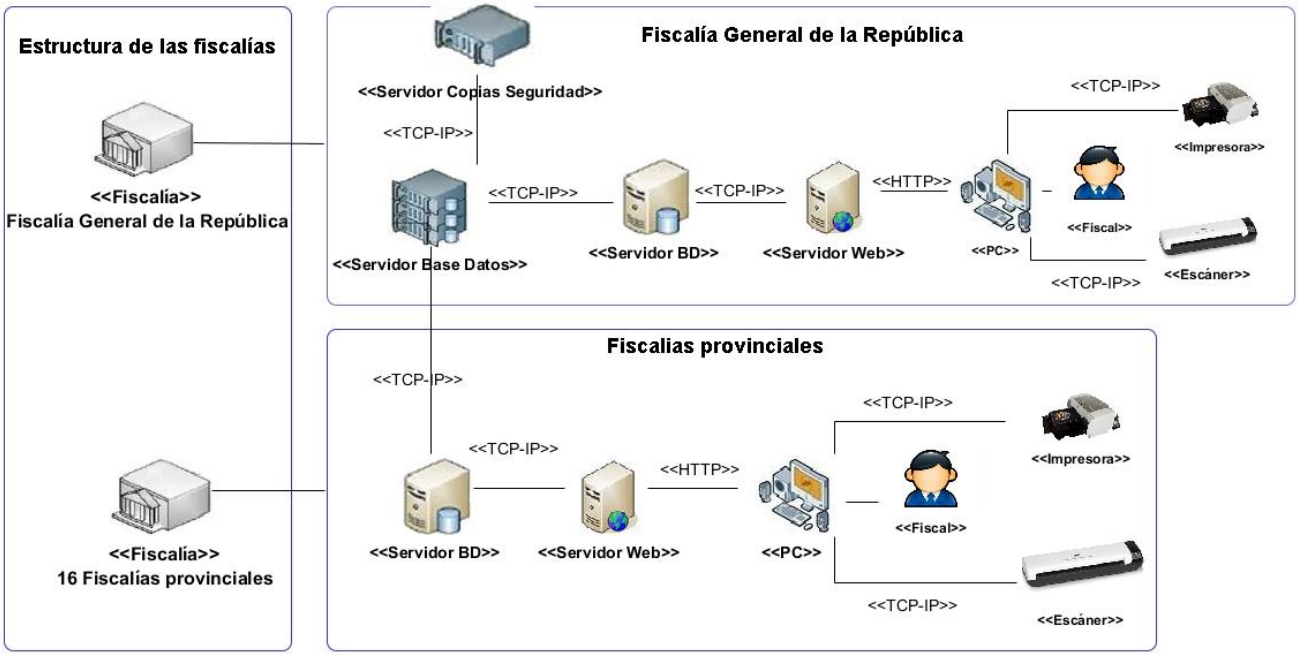


Figura 25. Diagrama de despliegue.

El diagrama muestra la distribución de los nodos necesarios para el despliegue del sistema. En la FGR y en cada instancia de las fiscalías provinciales se ubicarán ordenadores clientes, impresoras, escáneres, un servidor web, y un servidor de base de datos, mientras que en la FGR se ubicará también un servidor de copias de seguridad. El servidor de base de datos de las instancias provinciales se conectará directamente con el servidor de base de datos de la FGR encargado de la replicación de datos. La comunicación entre los ordenadores cliente y los servidores web será a través del protocolo HTTP. La comunicación entre todos los demás nodos se realizará mediante el protocolo TCP/IP.

3.2 Estándares de codificación.

Los estándares de codificación son una serie de convenciones que deben seguir los desarrolladores manteniendo buenas prácticas definidas por la Ingeniería de Software, para obtener un código fácil de comprender y de alta calidad. Posibilita que un equipo de programadores mantenga un código de calidad sobre el que se efectuarán luego revisiones del código (46); con el objetivo de regular la calidad de la implementación estableciendo un estándar de desarrollo común por el cual se rige la programación del sistema informático integrado.

En el caso del módulo de Revisiones de causas penales, el estándar de codificación fue definido por la dirección del proyecto que se encuentra en el documento “0120_54 Estándares de codificación para PHP”.

A continuación se muestra cómo nombrar algunos de los elementos en el módulo de Revisiones de causas penales:

- Todas las nomenclaturas a utilizar son en español.
- El identificador para las variables, los parámetros y los métodos se definen escribiendo, las palabras simples con la primera letra en minúsculas, y en caso de ser un nombre compuesto usando la notación Camello, para el cual existen dos variantes:
    - **UpperCamelCase:** las palabras que forman el nombre se escriben juntas y la primera letra de cada una de ellas en mayúscula.
    - **lowerCamelCase:** las palabras que forman el nombre se escriben juntas, la primera letra de la primera palabra en minúscula y del resto de las palabras, la primera letra en mayúscula.

En el caso del módulo de Revisiones de causas penales se utilizará la segunda variante para definir los atributos y variables, en otro caso se aplicará la primera variante.

- El bundle del módulo de Revisiones de causas penales se escribe con el último nombre seguido de la palabra Bundle (PenalesBundle).

Propiedades y Funcionalidades del sistema.

- Los formularios comienzan con nombre del formulario según su función, seguido de la palabra Type (EscritoPromocionType.php)
- Los gestores de negocio comienzan con el nombre del gestor seguido de Gtr (SolicitudPenalesGtr.php)

Nomenclatura para los componentes de formularios:

- Los botones comienzan con las siglas Btn seguido del nombre de la acción (BtnAdicionar)
- El nombre del formulario está compuesto por el nombre del formulario, éste a su vez puede contener dentro otros formularios, este nombre estaría compuesto por el nombre del primer formulario_nombre del segundo y así sucesivamente si hay otros formularios anidados (pdc_penalesbundle_adicionarsolicitudtype_solicitud).

Todo lo anterior refleja características de implementación de código php propias de un mismo programador en conformidad con la legibilidad y uniformidad a establecerse como buena práctica de programación en todo sistema.

3.3 Pruebas.

Las pruebas de software son el conjunto de herramientas, técnicas y métodos que permiten determinar la calidad de un producto. Un buen caso de prueba es aquel que tiene alta probabilidad de mostrar un error no descubierto hasta entonces. Estos pueden ocurrir desde el comienzo del proceso, ya sea en la definición de los objetivos, el diseño, la implementación o en otras fases. Involucran las operaciones del sistema bajo condiciones controladas y evaluando los resultados. Las pruebas no pueden confirmar la ausencia de errores del software, pero sí demostrar que tiene defectos. La fase de pruebas es la que añade al producto final el valor para afirmar que ya se ha alcanzado la calidad requerida. Es por ello que probar es una de las fases más importantes para que un producto salga con calidad (47).

3.3.1 Prueba de caja negra.

Las pruebas de caja negra se llevan a cabo sobre la interfaz del software, obviando el comportamiento interno y la estructura del programa, verificando que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como la integridad de la información externa se mantiene (41).

Las pruebas de caja negra pretenden encontrar estos tipos de errores:

- Funciones incorrectas o ausentes.
- Errores en la interfaz.
- Errores en estructuras de datos o en accesos a bases de datos.

Este método consta de varias técnicas:

- Técnica de la Partición de Equivalencia: Técnica que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a una definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar (41).
- Técnica del Análisis de Valores Límites: La experiencia muestra que los casos de prueba que exploran las condiciones límite producen mejor resultado que aquellos que no lo hacen. Las condiciones límites son aquellas que se hallan en los márgenes de la clase de equivalencia, tanto de entrada como de salida. Por ello, se ha desarrollado el análisis de valores límite como técnica de prueba. Esta técnica indica los casos de prueba que ejerciten los valores límites (41).

La técnica a emplear es la de Partición de Equivalencia, donde una clase de equivalencia representa un conjunto de estados válidos, inválidos o que no aplican, para varias condiciones de entrada. Las condiciones de entrada son valores numéricos específicos, un rango de valores, un conjunto de valores relacionados o una condición lógica. Para ver el resultado del caso de prueba para el requisito Adicionar Solicitud del módulo Revisiones causas penales ver: Anexo # 1.

Se realizaron pruebas de caja negra a los 36 requisitos de mayor importancia. De estos, en la primera iteración se obtuvieron 41 no conformidades y el resto resultaron correctas. En una segunda iteración se corrigieron las no conformidades y resultó un total de 13 errores; luego para una tercera iteración con un total de 36 casos de prueba se obtuvo un resultado correcto.

3.3.2 Pruebas de unidad.

Las pruebas de unidad son un procedimiento para comprobar el funcionamiento correcto de una porción del código del sistema de manera aislada. En caso de una programación de procedimientos se entiende

como un programa individual, una función o procedimiento, etcétera. Cuando se está programando orientado a objetos la menor de estas unidades es siempre una clase (41). Las “pruebas de unidades” son orientados casi siempre a las pruebas de “caja blanca” aunque para realizar una de ellas es necesario probar el flujo de datos desde la interfaz del componente (48).

3.3.3 Pruebas de caja blanca o funcionales.

Estas pruebas se basan en la especificación del programa o componente a ser probado para elaborar los casos de prueba. En este tipo de prueba se emplea la técnica del camino básico.

Prueba del camino básico.

Para aplicar esta prueba se han definido una serie de pasos a seguir que a continuación se describen:

1. **Notación del grafo de flujo:** Usando el código como base se realiza la representación del grafo de flujo, mediante una sencilla notación. Cada construcción estructurada tiene su correspondiente símbolo.
    - Nodo: Cada círculo denominado nodo, representa una o más sentencias procedimentales.
    - Arista: Las flechas del grafo de flujo, denominadas aristas, representan el flujo de control y son análogas a las flechas del diagrama de flujo.
    - Región: Las áreas delimitadas por aristas y nodos se denominan regiones.
2. **Complejidad ciclomática:** Es una métrica que proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado define el número de caminos independientes del conjunto básico de un programa, lo que indica el límite superior para el número de pruebas que se deben realizar, para asegurar que se ejecuta cada sentencia al menos una vez. La complejidad ciclomática se calcula de tres formas:
    1. El número de regiones del grafo de flujo coincide con la complejidad ciclomática.
    2. La complejidad ciclomática, V (G), de un grafo de flujo G se define como: V (G)=A-N+2, donde A es el número de aristas del grafo de flujo y N es el número de nodos.
    3. La complejidad ciclomática, V (G), de un grafo de flujo G también se define como V (G) P+ 1, donde P es el número de nodos predicado contenidos en el grafo de flujo G.
3. **Determinar un conjunto básico de caminos linealmente independientes**: El valor de V (G) es el número de caminos linealmente independientes de la estructura de control del programa.

4. **Obtención de casos de prueba:** Se realizan los casos de pruebas que forzaran la ejecución de cada camino del conjunto básico.

Aplicación de la prueba.

La siguiente figura muestra el código que se tomó como base para realizar el procedimiento anteriormente descrito para la técnica de camino básico, correspondiente al método **salvarActualizarSolicitudAction.**

```
public function salvarActualizarSolicitudAction($id_solicitud ){
   //1
   $peticion = $this->getRequest();
   $solicitud = $this->getSolicitudPenalesGtr()->cargarSolicitudPorId($id_solicitud);
       //2
   if(count($solicitud->getEntidadesProceso())>0)
   //3
   $sin_entidad=1;
   else
   //4
   $sin_entidad=0;
   //5
   $frm = $this->createForm(new SolicitudType(), $solicitud,array('sin_entidad' => $sin_entidad));
    //6
   if ($peticion->getMethod() == 'POST') {
   //8
       $frm->bindRequest($peticion);
       //9
      if ($frm->isValid()) {
      //10
            $usuario = $this->getUser();
                  $tipo_boton = $peticion->get(ConfigUtil::btn_guardar);
            //11
            if (isset($tipo_boton)) {
            //12
                $this->getSolicitudPenalesGtr()->salvarActualizarSolicitud($solicitud,
                ConfigUtil::tipo_proceso_solicitudes_penales, ConfigUtil::paso_iniciar_solicitud,
                ConfigUtil::estado_pendiente, $usuario);
                return $this->redirect($this->generateUrl('gestionar_solicitud',
                array('paso' => ConfigUtil::paso_iniciar_solicitud)));
            } else {
            //13
                $this->getSolicitudPenalesGtr()->salvarActualizarSolicitud($solicitud,
                ConfigUtil::tipo_proceso_solicitudes_penales, ConfigUtil::paso_iniciar_solicitud,
                ConfigUtil::estado_edicion, $usuario);
                $datos=$this->getRequest()->request->get('pdc_solicitud_type');
                //14
                if(isset($datos['entidad_involucrada']))
                //15
                    return $this->redirect($this->generateUrl('rcp_salvar_actualizar_entidad',
                    array('id_solicitud' => $solicitud->getIdProceso())));
                else
                //16
                   return $this->redirect($this->generateUrl('rcp_salvar_actualizar_personas',
                   array('id_solicitud' => $solicitud->getIdProceso())));
            }

      }
}
//17
return $this->render('PenalesBundle:Solicitud:adicionar_solicitud.html.twig',
array('frm' => $frm->createView(),
            'id_solicitud' => $solicitud->getIdProceso()));
```

Figura 26. Código fuente del método salvarActualizarSolicitudAction.

Se realiza el **grafo de flujo** del método seleccionado:

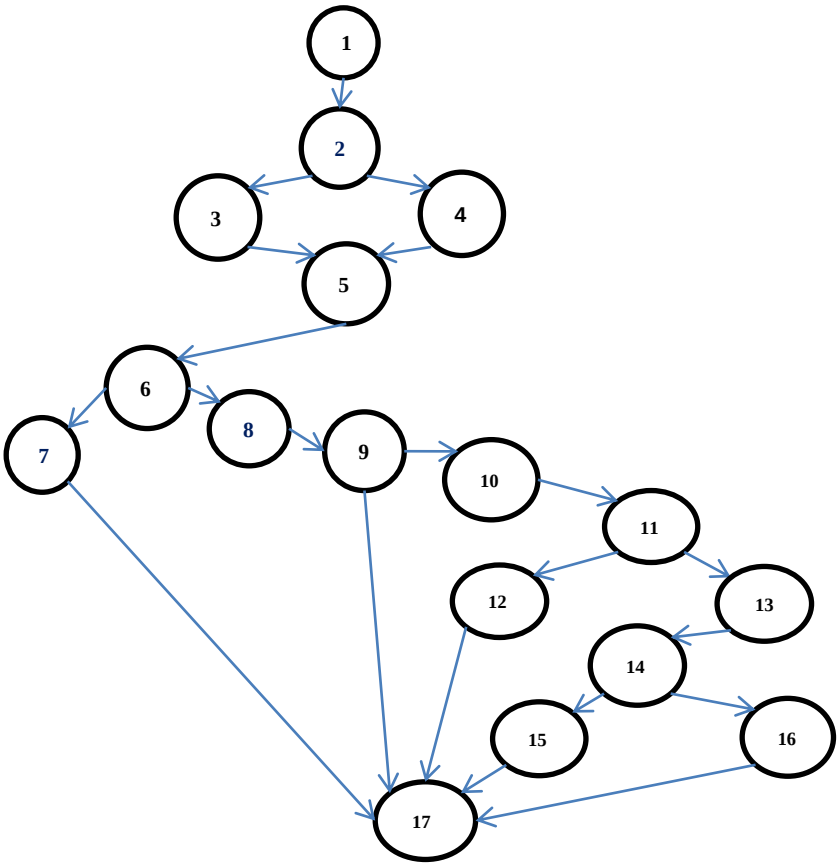


Figura 27. Grafo de flujo del método salvarActualizarSolicitudAction.

Luego se calcula la complejidad ciclomática, mediante las siguientes tres métodos:

V (G)= A.

V (G)= 6.

Siendo A la cantidad de áreas. Incluyendo el grafo como otra región.

V(G)= A-N+2.

V (G)= 21-17+2=6.

Siendo A la suma de las aristas y N la suma de los nodos.

V (G)=P+1.
V (G)=5+1=6.
Siendo P el total de nodos predicados.

El valor de V (G) es 6 lo que representa la cantidad de **caminos linealmente independientes** de la estructura de control del programa. Seguidamente se representan los caminos básicos por los que puede recorrer el código.

Camino1: 1-2-3-5-6-7-17.
Camino2: 1-2-4-5-6-7-17.
Camino3: 1-2-3-5-6-8-9-17.
Camino4: 1-2-4-5-6-8-9-10-11-12-17.
Camino5: 1-2-4-5-6-8-9-10-11-13-14-15-17.
Camino6: 1-2-3-5-6-8-9-10-11-13-14-16-17.

Cada camino independiente es un **caso de prueba** a realizar, de forma que los datos introducidos provoquen que se visiten las sentencias vinculadas a cada nodo del camino; a continuación se procede a ejecutar los casos de pruebas para este procedimiento.

Tabla 6. Descripción de las variables probadas en el caso de prueba.

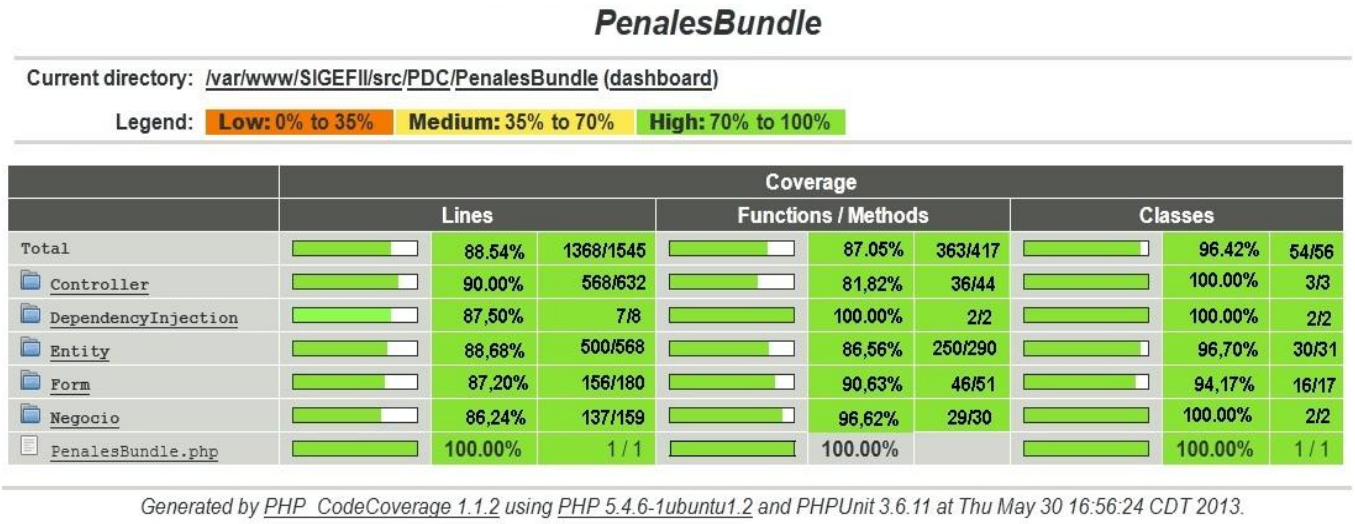
No	Descripción	Condición	Entrada	Resultado
1	Se muestra una interfaz con los atributos de la solicitud para guardar y actualizar.	count=($solicitud->getEntidadesProceso()>0) $peticion->getMethod() = 'GET'	Como entrada se necesita que la cantidad de solicitudes sea mayor que 0 y que la petición se haya realizado por el método GET enviados.	Se muestra la interfaz adicionar solicitud.
2	Se muestra una interfaz con los atributos de la solicitud para guardar	count=($solicitud->getEntidadesProceso()>0) $peticion->getMethod() = 'GET'	Como entrada se necesita que la cantidad de solicitudes sea menor o igual que cero y que la	Se muestra la interfaz adicionar solicitud.

	y actualizar.		petición se haya realizado por el método GET enviados	
3	Se muestra una interfaz con los atributos de la solicitud para guardar y actualizar.	count=($solicitud->getEntidadesProceso()>0) $peticion->getMethod() = 'POST' $frm->isValid()=false	Como entrada se necesita que la cantidad de solicitudes sea menor o igual que cero, que la petición se haya realizado por el método POST y que los datos del formulario sean válidos.	Se muestra la interfaz adicionar solicitud.
4	Se muestra una interfaz con los atributos de la solicitud para guardar y actualizar.	count=($solicitud->getEntidadesProceso()>0) $peticion->getMethod() = 'POST' $frm->isValid()=true isset($tipo_boton) =false	Como entrada se necesita que la cantidad de solicitudes sea menor o igual que cero, que la petición se haya realizado por el método POST, que los datos del formulario sean válidos y que la variable tipo_botón sea nula.	redirecciona hacia la ruta gestionar_solicitud.
5	Se muestra una interfaz con los atributos de la solicitud para guardar y actualizar.	count=($solicitud->getEntidadesProceso()>0) $peticion->getMethod() = 'POST' $frm->isValid()=true isset($tipo_boton) =true isset($datos['entidad_involucrada']) =true	Como entrada se necesita que la cantidad de solicitudes sea menor o igual que cero, que la petición se haya realizado por el método POST, que los datos del formulario sean válidos y	Los datos del rollo se salvan en la base de datos y se redirecciona hacia la ruta rcp_salvar_actualizar_entidad.

			la variable tipo_botón no sea nula.	
6	Se muestra una interfaz con los atributos de la solicitud para guardar y actualizar.	count=($solicitud->getEntidadesProceso()>0) $peticion->getMethod() = 'POST' $frm->isValid()=true isset($tipo_boton) =true isset($datos['entidad_involucrada']) =false	Como entrada se necesita que la cantidad de solicitudes sea menor o igual que cero, que la petición se haya realizado por el método POST, que los datos del formulario son válidos y que la variable tipo_botón no sea nula.	Los datos del rollo se salvan en la base de datos y se redirecciona hacia la ruta rcp_salvar_actualizar_persona.

Resultados obtenidos.

Luego de realizar la prueba de caja blanca al método salvarActualizarSolicitudAction a través de la técnica del camino básico, se obtuvo el resultado esperado. Se realizó la prueba de caja blanca al módulo Revisiones de causas penales a través de la herramienta PHPUnit; en una primera iteración se descubrieron 18 fallos que posteriormente fueron analizados y resueltos. Seguidamente se muestran los resultados de la aplicación de dichas pruebas.



PenalesBundle

Current directory: /var/www/SIGEFII/src/PDC/PenalesBundle (dashboard)

Legend: **Low:** 0% to 35% **Medium:** 35% to 70% **High:** 70% to 100%

	Coverage								
	Lines			Functions / Methods			Classes		
Total		88.54%	1368/1545		87.05%	363/417		96.42%	54/56
Controller		90.00%	568/632		81,82%	36/44		100.00%	3/3
DependencyInjection		87,50%	7/8		100.00%	2/2		100.00%	2/2
Entity		88,68%	500/568		86,56%	250/290		96,70%	30/31
Form		87,20%	156/180		90,63%	46/51		94,17%	16/17
Negocio		86,24%	137/159		96,62%	29/30		100.00%	2/2
PenalesBundle.php		100.00%	1 / 1		100.00%			100.00%	1 / 1

Generated by PHP_CodeCoverage 1.1.2 using PHP 5.4.6-1ubuntu1.2 and PHPUnit 3.6.11 at Thu May 30 16:56:24 CDT 2013.

Figura 28. Resultados de la aplicación de pruebas unitarias con PHPUnit.

Después de aplicadas las pruebas al módulo Revisiones de causas penales se obtuvo como resultado un 88.54% de las líneas de códigos ejecutadas, un 87.05% de las funcionalidades ejecutadas y un 96.42% de las clases ejecutadas. La prueba de la caja blanca demostró que el estado real del software coincide con el esperado, comprobándose a través de los casos de prueba desarrollados a través de la herramienta PHPUnit donde la ejecución de las condiciones y los bucles tuvieron resultados satisfactorios.

3.4 Validación de las variables.

Para darle cumplimiento al objetivo planteado se procede a validar las variables identificadas. A continuación se muestran los resultados obtenidos:

Variable independiente: Gestión de la información.

Variables dependientes: Ejecución y control.

Tabla 7. Cuadro operacional de las variables.

Variables	Propiedades	Indicadores
Gestión de la información	Documentos generados	Tiempo de creación y tramitación.
		Tiempo de traslado de la documentación.

	Acceso a la información	Tiempo para realizar búsquedas.
		Persistencia de la información.
		Acceso por niveles de usuarios.
Ejecución	Acceso a la información	Tiempo empleado en la búsqueda de un rollo o documento.
	Documentos generados	Tiempo de creación.
		Tiempo de traslado.
Control	Leyes y metodología	Cumplimiento de la metodología de trabajo.
		Realización de un proceso.
	Acceso a la información	Visualización de los rollos que se crean a diario o en un período.
		Acceso por niveles de usuarios.

Una vez desarrollada la aplicación se pudo comprobar que las búsquedas de información que antes demoraba hasta horas, con la aplicación se demora solo segundos. Además se tiene siempre un control de los usuarios del sistema, así como quien realiza los procesos y genera los documentos. Otra ventaja es que la trasferencia de información de las fiscalías, que hasta el momento se realiza mediante una persona que lleva los documentos hacia las distintas instancias, mediante la aplicación se reduce considerablemente el tiempo de transferencia.

3.5 Conclusiones Parciales.

En el desarrollo del presente capítulo se determinaron los elementos de la fase de implementación así como su validación. Se obtuvieron los diagramas de componentes y de despliegue, se tuvo en cuenta las pautas de codificación para logar una uniformidad y entendimiento del código fuente. Se valida el sistema desarrollado mediante pruebas de caja negra y caja blanca, realizando casos de prueba a varias de las funcionalidades. Este proceso permite detectar no conformidades presentes en el software para así darle solución. Luego de una segunda iteración no se encontraron no conformidades, obteniendo finalmente una aplicación que satisface los requisitos definidos.

CONCLUSIONES GENERALES

Luego de realizar el presente trabajo se puede concluir que:

- ✓ El estudio y análisis de sistemas existentes, demostró la necesidad de crear una solución informática con las leyes y procedimientos de las fiscalías cubanas.
- ✓ El Programa de Mejora como guía de desarrollo permitió seleccionar herramientas y lenguajes que logran ajustarse a las necesidades del proceso de revisiones de causas penales.
- ✓ Fueron utilizadas varias técnicas y patrones en las fases desarrolladas, que permitieron obtener resultados eficaces, los cuales se vieron reflejados en cada uno de los artefactos generados.
- ✓ Las técnicas y métricas utilizadas para validar los artefactos obtenidos en cada una de las disciplinas, propiciaron la correcta medición de los mismos garantizando así resultados eficientes que lograron contribuir con la rápida implementación del sistema.
- ✓ Mediante el estudio de diferentes técnicas y métodos de pruebas se logró comprobar la calidad que ofrece el sistema.

RECOMENDACIONES

- ✓ Integrar el módulo Revisiones de causas penales al subsistema PDC del Sistema de Informatización de la Gestión de las Fiscalías fase II.
- ✓ Realizar el despliegue del módulo Revisiones de causas penales como parte del Sistema de Informatización de la Gestión de las Fiscalías fase II.
- ✓ Crear una infraestructura de interoperabilidad que permita integrar el módulo implementado con el Sistema de Informatización para la Gestión de los Tribunales Populares Cubanos.

BIBLIOGRAFÍA

1. **Cuba, Fiscalía General de la República de.** *Ley de la Fiscalía General de la República.* La Habana-Cuba : s.n., 2012.

2. —. *Indicaciones metodológicas para la atención a la población.*

3. **Pereira, Lic. Julio A.Fernández.** *Derecho Procesal Penal. Tema I.* La Habana : s.n.

4. **Cuba, Fiscalía General de la República de.** *Metodología sobre la Protección a los Derechos al Cuidadano.*

5. diariovea.com.v. *diariovea.com.v.* [En línea] [Citado el: 18 de diciembre de 2013.] http://diariovea.com.ve/sucesos/inicia-revision-de-causas-judiciales-de-poblacion-penitenciaria/.

6. PoderJudicialChile. [En línea] http://www.poderjudicial.cl.

7. ProgramaGobiernoenlínea. *ProgramaGobiernoenlínea.* [En línea] http://www.gobiernoenlinea.gov.co.

8. gobierno en linea argentina. [En línea] http://www.pjn.gov.ar/.

9. **Yenier Figueroa, Hector Fuentes, Manuel Delgado, Yanisleidy Torres.** *expediente proyecto.* La Habana Cuba : s.n., Octubre 2012.

10. **Figueroa Machado, Yenier.** *Proyecto Técnico: Sistema de informatización de la Gestión de las Fiscalías fase II.*

11. **Sommerville, Ian.** *Ingeniería del Software.* México DF : Editora Pearson, 2005.

12. **Barzanallana, Rafael.** Universidad de Murcia. [En línea] [Citado el: 2 de Diciembre de 2012.] http://www.um.es/docencia/barzana.

13. **Ivar Jacobson, Grady Booch, James Rumnaugh.** *El Proceso Unificado de Desarrollo de Software.* 2000.

14. **S. Pressman, Roger.** *Ingeniería de Software, un enfoque práctico.* 2005.

15. Calisoft. [En línea] http://calisoft.uci.cu/index.php/proceso-de-mejora.

16. **Machado, Yenier Figueroa.** *Proyecto Técnico del Sistema de Informatización de las Fiscalías fase II.*

17. **Mary Beth Chrissis, Mike Konrad, Sandy Shrum.** *CMMI. Guía para la integración de procesos.*

18. DefiniciónABC. *DefiniciónABC.* [En línea] [Citado el: 4 de diciembre de 2012.] http://www.definicionabc.com/.

19. **Kendall.** *Análisis y Diseño de Sistemas.* 1997.

20. Visual Paradigm. *Visual Paradigm.* [En línea] http://www.visual-paradigm.com/aboutus/10reasons.jsp.

21. **Schmuller, Joseph.** *Aprendiendo UML en 24 horas.*

22. Sitio oficial de PHP. [En línea] [Citado el: 8 de diciembre de 2012.] http://www.php.net/.

23. Adelat. [En línea] 2000. [Citado el: 6 de diciembre de 2012.] http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html.

24. **Franganillo, Jorge.** *Html5: el nuevo estándar básico de la Web.*

25. mozilla. *mozilla.* [En línea] [Citado el: 23 de diciembre de 2013.] https://developer.mozilla.org/es/docs/Web/HTML.

26. **Bradenbaugh, Jerry.** *Aplicaciones JavaScript.* s.l. : Anaya Multimedia, 2000.

27. **Flanagan, David.** *JavaScript: The Definitive Guide (4 ª Edición edición).* 2002. ISBN 0-596-00048-0..

28. jquery. *jquery.* [En línea] [Citado el: 23 de 12 de 2012.] http://jquery.com/.

29. subversion. *subversion.* [En línea] [Citado el: 23 de diciembre de 2013.] http://subversion.apache.org/.

30. tortoisesvn. *tortoisesvn.* [En línea] [Citado el: 21 de noviembre de 2013.] http://tortoisesvn.net/.

31. **Eguiluz, Javier.** *Desarrollo Web Ágil Con Symfony2.*

32. symfony. *symfony.* [En línea] [Citado el: 21 de diciembre de 2013.] http://www.symfony.es/.

33. **Sun Microsystem.** Netbeans. [En línea] Sun Microsystem, 2006. [Citado el: 21 de 12 de 2010.] http:\\www.netbeans.org\index_es.html.

34. postgresql. *postgresql.* [En línea] [Citado el: 20 de Diciembre de 2013.] http://www.postgresql.org/.

35. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : PRENTICE HALL, 1999. ISBN 970-17-0261-1.

36. Patrones del "Gand of Four". Madrid, España : s.n.

37. **Mc Graw Hill. Pérez, F.** *Metodología de la Investigación.* México : s.n., 2005.

38. **Robbins, Stephen P.** *Comportamiento Organizacional, Teoría y Práctica. Séptima Edición.* México : Editorial Prentice Hall, 1996.

39. **Olsina, L., Bertoa M. F. y Lafuente G. J., Martín M. A., Katrib M., Vallecillo A.** *Un Marco Conceptual para la Definición y Explotación de Métricas de Calidad.* Madrid : s.n., 2002.

40. **Sánchez Fornaris, Maite y Alcantara Rabí, Dayanis.** Propuesta de una guía de métricas para evaluar el desarrollo de los Sistemas de Información Geográfica. [En línea] http://vinculando.org/articulos/sociedad_america_latina/propuesta_guia_de_medidas_para_evaluacion_sistemas_informacion.html..

41. **Pressman, Roger S.** *Ingeniería de Software. Un enfoque práctico. 5ta edición.* s.l. : McGraw-Hill, 2002. ISBN: 8448132149.

42. **Chidamber S. R., Kemerer C. F.** *"A metric suite forsz".* s.l. : IEEE Transactions on Software Engineering, 1994. pp. 467-493.

43. **Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides.** *Design Patterns: Elements of Reusable Object-Oriented Software.* 2005. ISBN 0-201-63361-2.

44. **Bernd Bruegge, Pearson Educación.** *Ingeniería de Software Orientado a Objetos.*

45. *Ayuda de RUP. Rational Unified Process.* s.l. : Copyright (C) IBM Corporation 1987. Versión 7.0.12005.

46. **Machado, Ing. Yenier Figueroa.** Proyecto Técnico del Sistema de Informatización de la Gestión de las Fiscalías II. [En línea] 2011.

47. **Collado, Manuel.** www.lml.ls.fi.upm.es. [En línea] marzo de 2003. [Citado el: 26 de abril de 2013.] http://www.lml.ls.fi.upm.es/ftp/ed2/0203/Apuntes/pruebas.ppt..

48. **Fernando.** http://www.inwebwetrust.net/post/2006/12/01/testeando-varios-valores-un-atributo-un-test-unidad. *inwebwetrust.* [En línea] 1 de diciembre de 2006. [Citado el: 26 de abril de 1013.] http://www.inwebwetrust.net/post/2006/12/01/testeando-varios-valores-un-atributo-un-test-unidad..

ANEXO

Anexo # 1: Caso de prueba para el requisito adicionar causa.

Tabla 8. Caso de prueba para el requisito adicionar causa.

Escenario	Descripción	Número de causa	Año	Tribunal Popular	Provincia	Sala	Número de sentencia	Fecha de sentencia	Delitos	Síntesis del hecho	Respuesta del sistema	Flujo central
EC 1.1. Adicionar causa con los datos correcto.	Permite adicionar una causa a una solicitud de revisión.	V 25	V 1999	V Provincial	V Matanzas	V Primera	V 222	V 12/04/2012	V Hurto	V Se ha cometido un delito	El sistema debe adicionar los datos de la causa.	1- Selecciona la opción Iniciar del menú principal del proceso de revisiones penales. 2- El sistema muestra una interfaz con el listado de las solicitudes adicionados. 3- Selecciona la opción Adicionar. 4- El sistema muestra una interfaz que
EC 1.2. Adicionar causa con campos vacíos	Faltan por llenar datos obligatorios.	N/A vacío	V 1999	V Provincial	V Matanzas	V Primera	V 222	V 12/04/2012	V Hurto	V Se ha cometido un delito	El sistema debe mostrar un mensaje indicando que existen campos obligatorios vacíos, al seleccionar la opción aceptar.	
		V 25	N/A vacío	V Provincial	V Matanzas	V Primera	V 222	V 12/04/2012	V Hurto	V Se ha cometido un delito		
		V 25	V 1999	N/A vacío	V Matanzas	V Primera	V 222	V 12/04/2012	V Hurto	V Se ha cometido un delito		
		V 25	V 1999	V Provincial	N/A vacío	V Primera	V 222	V 12/04/2012	V Hurto	V Se ha cometido un delito tribunal.		
		V 25	V 1999	V Provincial	V Matanzas	N/A vacío	V 222	V 12/04/2012	V Hurto	V Se ha cometido un delito		

		V	V	V	V	V	N/A	V	V	V		contiene los datos a introducir. 5- Selecciona la opción Siguiente. 6- El sistema muestra una interfaz para buscar los involucrados 7- Selecciona el involucrado. 8- El sistema muestra una interfaz con el listado de las causas. 9- Selecciona la opción adicionar 10- El sistema muestra una interfaz con los datos de la causa a introducir 11- Registra la solicitud en
		25	1999	Provincial	Matanzas	Primera	vacío	12/04/2012	Hurto	Se ha cometido un delito		
		V	V	V	V	V	V	N/A	V	V		
		25	1999	Provincial	Matanzas	Primera	222	vacío	Hurto	Se ha cometido un delito.		
		V	V	V	V	V	V	V	N/A	V		
		25	1999	Provincial	Matanzas	Primera	222	12/04/2012	vacío	Se ha cometido un delito		
		V	V	V	V	V	V	V	V	N/A		
		25	1999	Provincial	Matanzas	Primera	222	12/04/2012	Hurto	vacío		
EC 1.3.Adionar causas con datinválido.	Proceso de adición de la causa con datos incorrectos	I	V	V	V	V	V	V	V	V	Debe mostrar un mensaje indicando que el número de causa **es incorrecto**	
		8p	1999	Provincial	Matanzas	Primera	222	12/04/2012	Hurto	Se ha cometido		
		V	V	V	V	V	V	V	V	V	Debe mostrar un mensaje indicando que la Fecha de sentencia debe ser menor que la fecha de registro de la solicitud.	
		23	2010	Municipal	Granma	Segunda	213	12/12/2013	Hurto	Se ha cometido un delito		
		V	I	V	V	V	V	V	V	V	Debe mostrar un mensaje indicando que el campo: **Año** solo admite números, el valor debe ser de cuatro dígitos y mayor que 1900.	
		36	202w	Municipal	Matanzas	Segunda	40	12/04/2012	Hurto	Se ha cometido un delito		
		V	V	V	V	V	I	V	V	V	Debe mostrar un mensaje indicando que el campo: **Número de Sentencia** solo admite números, y el valor debe ser mayor o igual que 1.	
		36	2010	Municipal	Cienfuegos	Tercera	3*	12/04/2012	Hurto	Se ha cometido un delito		

												el listado de solicitudes.