Universidad de las Ciencias Informáticas.
Facultad 3

IMPLEMENTACIÓN DEL PROCEDIMIENTO DILIGENCIAS PREVIAS DEL SUBSISTEMA ECONÓMICO DEL PROYECTO DE INFORMATIZACIÓN DE LOS TRIBUNALES POPULARES CUBANOS

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

Autores:	David Estévez Díaz
	Pedro Enrique Romero Manfugas
Tutor:	Ing. Daimi Lamorú Marciel
Co-Tutor:	Ing. Angel Alexander Guillén Suárez

La Habana, Cuba
Curso 2011–2012

Declaración de autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 3 de 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 _____.

David Estévez Díaz

Autor

Pedro Enrique Romero Manfugas

Autor

Ing. Daimi Lamorú Marciel

Tutor

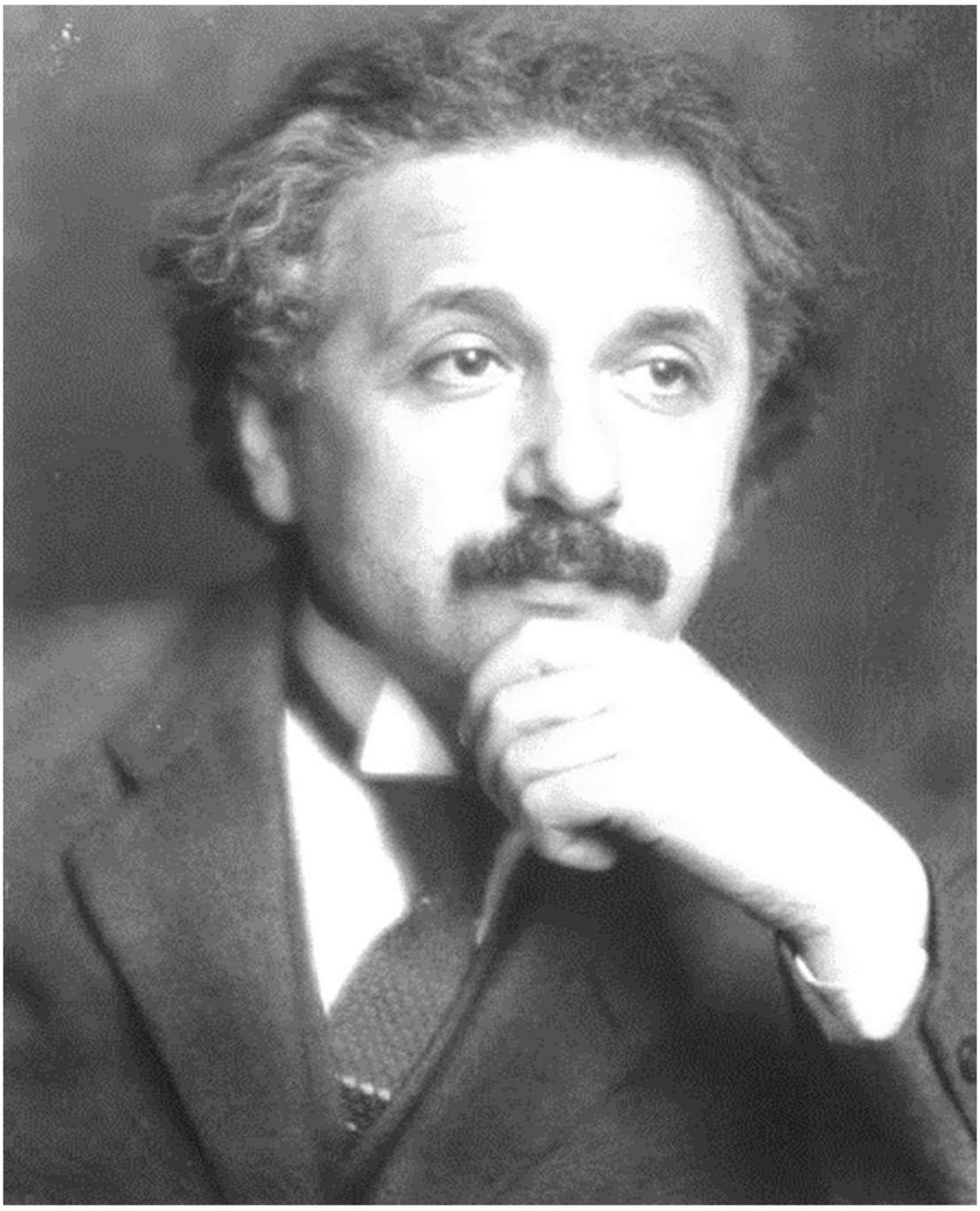
Ing. Angel Alexander Guillén Suárez

Co-Tutor

Pensamiento

"Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad."

Albert Einstein



Agradecimientos

A mi madre por ser mi padre y madre al mismo tiempo, confiar siempre en mí, por sacrificarse tanto por verme graduado y apoyarme en todas mis decisiones.

A mi tía por ser mi segunda madre, por ayudarme en todo, por inculcarme siempre buenos valores y estar siempre pendiente de mis necesidades, dispuesta a darme lo que necesitara en la medida de sus posibilidades.

A mi padrastro por acogerme como su hijo mayor y darme ese apoyo incondicional en todo momento y especialmente en los que más lo necesitaba.

A mi hermano por ser tan cariñoso conmigo y arreglarme todas las cosas que rompo.

A toda mi familia y a la de mi novia que también la considero mi familia. Gracias a ustedes he logrado vencer este difícil camino que sin su ayuda me hubiera sido imposible.

A mi novia que no ha vacilado jamás en apoyarme ante una situación difícil en estos 5 años, gracias por no dejarme caer. Has sido novia, hermana y amiga, sin ti no habría sido posible.

A Pedro por haber sido mi compañero de tesis y amigo.

A los tutores Daimi y Alexander que fueron de gran ayuda.

Al todo el equipo de trabajo del proyecto CCV por darme la oportunidad de formar parte de ustedes, en especial a Pepe por enseñarme a programar bien y por ser uno de los mejores profesores que tiene esta universidad, gracias por permitirme ser tu amigo. A los Albertos el gordo y el no tan gordo, a Yurita, Linnet, Danaysa, Martin Polar, Yudier, el Yosma, Livan, Yasmany, Liennys y esa constelación de estrellas que conformo al proyecto CCV. De verdad, fue un placer trabajar con ustedes y que confiaran en mi tanto en Cuba como en Venezuela. Ojala algún día pudiera compartir con todos ustedes en otro proyecto.

A todos mis compañeros de cuarto y de grupo. Les deseo éxitos a todos

A todos los que de una forma u otra han colaborado con la realización de este trabajo…

David Estévez Díaz.

Agradecimientos

A la persona más especial en mi vida mi abuelita Pitu, cumpliéndole uno de sus sueños, verme graduado, ya lo lograste con una nieta, mi hermana, toda mi vida ha sido dedicada a ti, y hoy te demuestro que todo el esfuerzo que has hecho en la educación de tus sucesores no ha sido en vano.

A mi mamá por demostrarme que el que persevera triunfa, por corregirme en todos mis errores, por confiar tanto en mí, por hacerme creer que soy grande.

A mi papá por demostrarme que en la vida todo se puede, que hay que tener paciencia, pues todo llega, por darme esos consejos que me sirven de mucho, por ser quien es, porque además juntos ellos dos han sabido darme la educación que tengo, por no permitirme pasar trabajo en la vida pues siempre están ahí para cualquier problema, porque son lo más grande que tengo y yo vivo por y para ustedes, porque siempre quise que vieran el fruto de su trabajo en mi formación como persona.

A mi hermana por ser mi ángel de la guarda, por la relación tan especial que tenemos, por darme su ejemplo, por a pesar de que éramos enemigos cuando pequeños, nos hemos hecho los mejores amigos, por ser mi paradigma.

A mis tíos Marilys y Gobert por siempre estar pendientes de mí y apreciarme como su hijo, créanme que me siento de tal manera, junto a ellos, agradecerle a Joaco y a Ela que también son parte de mi familia, a mi abuelo Rigo por siempre tratarme como lo ha hecho, por estar ahí cuando se le necesita, no son pocas veces.

A mis primos por ser mis segundos hermanos, Pepe, Clau, Yadi y Erne que me han tenido como su ejemplo y me tienen en cuenta para todo, por ser tan especiales conmigo, por brindarme apoyo y por dejarme ser para ellos “El Negrito de la Virgen”.

A mi familia paterna por también preguntar y estar pendientes de mí, mi abuela, mi tía Norquis que es una de las madres que encontré aquí en la Habana, mi tío David, Alexis y mis primos Migue y Carlitos.

A mis hermanos del barrio, por su apoyo y preocupación Dameiki, Yuni, Armandito y Darvi. A Anita por ser tan amiga de la familia y a Mayi por ser otro hermano para mí, a Mayte por los buenos momentos que me hizo pasar.

Agradecimientos

A Teresa por su preocupación conmigo en todos los sentidos, a Eugenia por ser tan especial, y a Alberto, entre todos han conformado otra familia para mí.

A mi gente de Venezuela que me sirvieron de mucho cuando uno nos está cerca de la familia, fueron mi familia durante 9 meses en especial a Nade que fue mi guía, y una madre que encontré allá, Elena.

A mis tutores, que mejores no los quiero, juntos demostramos ser un buen equipo, Alexander y Daimi, que gracias a las relaciones interpersonales que existen el trabajo se torna más placentero.

A mi compañero de tesis, por ser compañero y amigo, además de su novia Eilianys que nos sirvió de gran ayuda.

A mis grandes amigos de la Universidad, creo que ha sido la mejor etapa en mi vida y no la olvidaré jamás, fueron los que me hicieron pasar los mejores momentos, en las charlas, las discusiones, los café que compartíamos para quitarnos el estrés, por enseñarme un poco de cada uno, a Arlyh, William, Yan Luis, Iosmany, Ernesto Adrián, Arnaldo, Máximo, Elvis, Adrián, Ernesto Mato, Collado, Jan Pablo, Luis Alard, Giorgy, Camejo, Pompa, Julito, Pedro Luis.

A las mujeres con las que entablé una gran amistad que para mí todas son especiales y únicas, son las que más me han ayudado en cuanto a consejos se refiere, mi piquete de 3er año Marlen, Eileen, Danay, Yelenny, Galia, Nivia, Daineris y Aleidys.

A mi gente del grupo, bueno los que quedamos mi querida Annia, Leidys, Claudia, Yisel Soto y Piñeiro, Katia, Yeremi, incluyo a Baby, que me fue de gran ayuda en mi formación como profesional, además de Yarenis, Gladys, a Aime, a los socios de ese grupo Yasiel, Rodney, Rosel, Manuel, Felipe, Pedro Frank, Frank, Rafa, Dayron, Dannier. Además a todo el que ha compartido grupo conmigo que de una forma u otra ha puesto su granito de arena. A mi grupo CDI en especial pues fue un tiempo bastante grato con ellos en el aula.

A mi gente de Infodanz, con los que he compartido momentos inolvidables, y son al menos las mujeres como Belkis, Vel, Rita, Ania, y en especial a mi pareja de baile Ricnelys que bueno a pesar de llegar a ser grandes amigos, logra con sus consejos llegar a la paz mental cuando me es necesario, a su mamá y padrastro por haberme acogido de la forma que o hicieron. A Los

varones Leandro, el Gato, Javier y Michel por dejarme divertirme con ellos en las recre o cualquier actividad que vamos juntos.

Durante la carrera existen un conjunto de profesores que intervienen en nuestra formación como profesional, pero existen algunos que influyeron más en mi como futuro ingeniero, ellos son el profe Rolan, la profe Yalice y Leidily, y en especial a la profe Hilda, por ser mi tutora en la ayudantía y por enseñarme tanto con su sabiduría.

Las tías del edificio, por los días de jarana durante este curso y por atendernos como lo hace, Lachi, Suilen, Rosita, Mayra.

A todo el que se encuentra aquí hoy, a los que han compartido aunque se aun momento conmigo, y a muchos que no puedo mencionar pues la lista es extensa, pero donde tienen que estar, que es mi mente, están.

Pedro Enrique Romero Manfugas

Dedicatoria

Dedico este trabajo a mi familia porque es mi inspiración y siempre los llevo presente, y porque un logro mío también es un logro de ellos. En especial a mi madre y a mi tía Zeida.

David Estévez Díaz.

Dedico este trabajo a toda mi familia y amigos, pues ha aportado un granito de arena en la confección del mismo, algunos con preocupación, otros con apoyo, dando ánimos, y contribuyendo en su desarrollo, este logro no es mío sólo, ustedes han sido partícipes de este y de todos en mi vida. Especialmente mi abuela Pitu, mis padres y mi hermana.

Pedro Enrique Romero Manfugas

Resumen

En la actualidad se llevan a cabo un conjunto de procesos judiciales en las diferentes instancias de los Tribunales Populares Cubanos (TPC), generando así un gran volumen de información que es recogida dentro de los expedientes que se radica en cada una de las salas, dichos expedientes con el paso del tiempo se deterioran comprometiendo la integridad de su contenido y producto al trabajo manual al que están expuestos ocurre duplicidad en la información. El proceso Diligencias Previas es uno de los tantos procedimientos que se realiza en la Sala de lo Económico de la instancia Provincial, donde se preparan los expedientes para darle solución a los litigios que fueron provocados por una deuda entre personas naturales y jurídicas.

Los problemas detectados en este procedimiento fueron traducidos por un equipo de analistas pertenecientes al proyecto TPC de la Universidad de las Ciencias Informáticas, nombrado Sistema de Informatización de Tribunales, traduciéndolos en 18 casos de uso en los que se describe cada una de las funcionalidades con que debe contar el sistema. En esta investigación se realiza un estudio y se seleccionan y describen las tecnologías, metodología y patrones definidos que pertenecen al proceso de desarrollo de software, se implementa los CU especificados por las analistas del proyecto TPC, teniendo en cuenta los estándares de codificación definidos y se le realiza un conjunto de pruebas que validarán su correcto funcionamiento.

Índice

Índice de Ilustraciones

Introducción

Con la socialización de las Tecnologías de la Información y las Comunicaciones (TICs) se ha hecho necesario incrementar la producción de software y con ello se han creado sistemas que han permitido informatizar los procesos que se realizan en distintas empresas. Cuba no se ha quedado atrás en este tema, pues se ha propuesto desarrollar soluciones internas a problemas de gran impacto social, jugando un papel primordial la Universidad de las Ciencias Informáticas (UCI).

La UCI está compuesta por varios centros de desarrollo donde cada uno de ellos se dedica a la producción de software de una rama social específica. El Centro de Gobierno Electrónico (CEGEL), encargado de las soluciones en la esfera de gobierno electrónico, está llevando a cabo el desarrollo de un Sistema de Informatización para los Tribunales (SIT) con la misión de informatizar todos sus procesos judiciales.

El SIT está compuesto por los subsistemas Administrativo, Civil, Laboral, Penal y Económico que trabajan de forma autónoma, aunque para su correcto funcionamiento necesitan la integración de los subsistemas Administración & Gobierno, Reportes y Común, que le brindan servicios de gran importancia para la realización de distintas funcionalidades.

En cada subsistema autónomo existen varios procedimientos y el Económico cuenta con cinco: Diligencias Previas, Ejecutivo, Ordinario, Casación y Revisión. Esta investigación comprenderá la informatización del procedimiento Diligencias Previas del subsistema Económico, cuyo propósito es agilizar el proceso de gestión de los expedientes en las distintas instancias de los tribunales.

En la actualidad este procedimiento se realiza de forma manual lo que conlleva que en ocasiones se cometan duplicaciones en la radicación de los números de los expedientes, en los asientos de los escritos y en el foliado de las páginas; lo que provoca borrones y tachaduras.

Los trámites que se realizan en el procedimiento Diligencias Previas tienen muy poco término porque así lo estipula la ley, lo que conlleva que los expedientes se encuentren en una tramitación constante y que tanto los jueces como el personal de secretaría se mantengan con un alto contenido de trabajo, provocando en ocasiones, que el término de algunos expedientes se venza sin poder ser atendidos.

Los expedientes son archivados en estantes, lugar donde la búsqueda de algunos de ellos se torna complicada por la gran cantidad de ejemplares guardados, a pesar de que estos se encuentran organizados por año. Al pasar el tiempo las carátulas de los expedientes se deterioran y cuando se necesita volverlos a usar en muchas ocasiones tienen que ser renovadas.

Actualmente en la sala de lo económico del Tribunal Provincial Popular (TPP) de La Habana existe un software estadístico, que brinda informes numéricos de los expedientes que se han radicado en el procedimiento Diligencias Previas. Este sistema no resuelve los problemas anteriormente definidos, porque no confecciona los expedientes de forma digital ni garantiza la continuidad de los procesos que normalmente se llevan a cabo en la sala, ya que sólo puede ser resuelto por un software de gestión procesal.

Como parte del proceso de desarrollo del software, el cliente, en este caso los directivos de los tribunales, identificó las necesidades existentes en este procedimiento y contrató a la UCI para la realización del software. El equipo de analistas del proyecto Tribunales Populares Cubanos (TPC), que son los designados por la universidad para el desarrollo del sistema, identificó las funcionalidades que debía tener el software para resolver estos problemas y se tradujeron en **51** requisitos funcionales que se resumen en **18** casos de uso, además el equipo de arquitectura identificó **34** requisitos no funcionales.(1)

Analizando lo anteriormente expuesto, se plantea el siguiente **problema a resolver:** la informatización existente en el procedimiento Diligencias Previas de la sala de lo económico de los TPC no satisface los requisitos definidos por el equipo de analistas del proyecto.

Se tiene como **objeto de estudio** el proceso de desarrollo de software, estableciendo como **objetivo general** implementar un software de gestión procesal que satisfaga los requisitos de software definidos por el equipo de analistas del proyecto TPC para el procedimiento Diligencias Previas de la sala de lo económico de los TPC.

El **campo de acción** de esta investigación es la implementación de un software de gestión procesal para la sala de lo económico de los TPC.

Para dar solución al objetivo trazado se han derivado un conjunto de **objetivos específicos** dirigidos esencialmente a suministrar los elementos necesarios para la implementación de la solución informática, los cuales se listan a continuación:

- Realizar el marco teórico de la investigación.
- Implementar los casos de uso del módulo Diligencias Previas.
- Validar la solución propuesta.

La **idea a defender** se plantea de la siguiente manera: con la implementación de un software de gestión procesal se podrán satisfacer los requisitos de software definidos por el equipo de analistas para el procedimiento Diligencias Previas de la sala de lo económico de los TPC.

Para cumplir con los objetivos específicos propuestos se plantearon las siguientes **tareas de la investigación:**

- Análisis de soluciones informáticas existentes en el ámbito nacional e internacional.
- Estudio de la metodología, tecnologías y herramientas propuestas por el equipo de arquitectura del proyecto TPC para el desarrollo de la aplicación.
- Diseño del diagrama de componentes.
- Implementación de los casos de uso del módulo Diligencias Previas.
- Diseño del diagrama de despliegue.
- Realización de las pruebas de software correspondientes.
- Análisis de los resultados de las pruebas realizadas.

Para la realización de las tareas propuestas se hace uso de algunos de los **métodos de investigación.**

Como **métodos teóricos** se utilizaron**:**

- El **método Analítico-Sintético** fue puesto en práctica en el estudio y análisis de la bibliografía, que permitió hacer una correcta selección de los conceptos y definiciones, sobre todo, lo que concierne al proceso de desarrollo de software que sirvieron para comprender mejor el problema y darle un buen cumplimiento a los objetivos, para lograr resultados satisfactorios en la investigación.
- El **método de Modelación** se utilizó en la creación de modelos que representan abstracciones, con el objetivo de comprender la estructura y funcionamiento de la implementación del proceso Diligencias Previas asociado a la sala de lo económico de los TPC.

La presente investigación está estructurada de la siguiente forma:

Capítulo 1. FUNDAMENTACIÓN TEÓRICA: En este capítulo se aborda sobre las soluciones existentes para procedimientos similares al de Diligencias Previas, la metodología de desarrollo, los patrones de diseño y de arquitectura, las tecnologías, herramientas a utilizar para el desarrollo del sistema y las pruebas realizadas a la aplicación.

Capítulo 2. DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA: En este capítulo se lleva a cabo una descripción de la propuesta del sistema, así como de las principales funcionalidades. Se argumenta el uso de los patrones de diseño, se modelan los diagramas de componentes y de despliegue. Comprenderá además la utilización de los estándares de codificación.

Capítulo 3. VALIDACIÓN DE LA SOLUCIÓN: En este capítulo se hace un análisis de los resultados obtenidos luego de la implementación de la solución propuesta mediante pruebas de caja negra y de caja blanca para validar su buen funcionamiento y desempeño.

Se espera, al finalizar esta investigación, obtener un sistema informático de gestión procesal para el procedimiento Diligencias Previas de la sala de lo económico de los TPC que cumpla con los casos de uso predefinidos por el equipo de analistas del proyecto TPC.

Capítulo 1: Fundamentación teórica

1.1 Introducción

Para comprender las necesidades de automatización de las distintas instituciones que existen en Cuba, es necesario hacer un estudio de las distintas soluciones que se han propuesto para darles respuesta.

En este capítulo se hace un estudio de los software que tienen como fin la gestión procesal de distintas instituciones judiciales internacionalmente y con el análisis, se puede demostrar por qué ninguno cumple con los casos de uso definidos por los analistas del proyecto TPC para el procedimiento Diligencias Previas, además se aborda sobre la metodología de desarrollo, los patrones de diseño y de arquitectura, las tecnologías, herramientas a utilizar para el desarrollo del sistema, así como las pruebas que se le realizarán a la aplicación.

1.2 Procedimiento Diligencias Previas

Se define como Diligencias Previas aquel acto judicial preparatorio, que sucede antes del proceso Ejecutivo y con el fin de obtener la confesión de la deuda y/o el reconocimiento de documento o el de su firma, los que a partir de este momento podrán adquirir fuerza ejecutiva como títulos de crédito, líquido, vencidos y exigibles, y de esta forma declarar preparada la acción ejecutiva. Se asegura, por tanto, el pago de una deuda reconocida por la entidad deudora.(2)

1.3 Soluciones informáticas existentes

Actualmente la creación de software para el campo jurídico está comenzando a crecer producto a la utilización de las TICs, con fines de agilizar y modernizar los procesos llevados a cabo por las instituciones que rigen en sus respectivos países, además de aprovechar las ventajas que el avance tecnológico aporta en las diferentes ramas de la sociedad. Ejemplo de ello son los sistemas informáticos que se han comenzado a utilizar en distintos países, como en España con el MINERVA, LEXNET y SEINSIR y Argentina

con el GIAJ - TRAMIX, para facilitar la tramitación de las causas, de los expedientes y acortar los plazos de duración de los juicios. Estos sistemas actúan sobre la base de la tecnología de los certificados digitales que permiten el uso de la firma digital y garantizan la autenticidad de los documentos que se envían.

Cuba es un país subdesarrollado, que tiene trabas para adquirir licencias informáticas que le permitan el uso de sistemas en distintas instituciones, y además de no poder, en ocasiones, acceder a estas tecnologías. Estos sistemas fueron creados para leyes vigentes en otros países, principalmente España, de donde son la mayoría. Adaptarlos al sistema judicial cubano sería más complicado y costoso que crear un software hecho a la medida y confeccionado completamente con tecnologías libres, más si se tiene a la UCI donde se cuenta con el personal capacitado para la creación de una nueva solución.

Por tanto, Cuba debe apostar por la realización de un sistema que cumpla con las características requeridas para satisfacer lo concerniente a la gestión de los servicios legales que se pueda utilizar en cualquier institución de este tipo. Un sistema que se ajuste a la realidad jurídica cubana, desarrollado por especialistas con herramientas de código abierto que permitan su mantenimiento, adaptaciones y posteriores versiones.

En este sentido el país ha realizado varios intentos de informatización, tal es el caso de los sistemas:

- SISPROP (Procesos penales, Villa Clara, Cienfuegos).
- SISECO (Estadística de los procesos económicos).

A continuación se expone una breve descripción de los sistemas anteriormente mencionados.

✓ **MINERVA (España)**

Es un sistema de gestión procesal, que permite el registro de los asuntos, su seguimiento estadístico y la tramitación de los procedimientos judiciales. Además automatiza e integra el seguimiento de los recursos y sentencias de la Consejería de Educación de la Junta de Andalucía incluyendo el seguimiento de la vertiente económica de estos expedientes. El mismo es un tramitador de expedientes que ha gestionado alrededor de más de 5000 expedientes hasta el momento y se encuentra en una fase de desarrollo para incluirle mejoras que contribuyan a una mayor eficiencia y eficacia de sus funcionalidades.(3)

- ✓ **LEXNET (España)**

Lexnet es un sistema de gestión de notificaciones telemáticas desde los juzgados a los profesionales de la justicia (abogados y procuradores) usado en la Administración de Justicia española que interopera con Minerva y Cicerone (Comunidad Valenciana), sistemas de gestión procesal de los juzgados promovido por el Ministerio de Justicia español. Su utilización se basa en el uso de un programa similar a un sistema de correo webmail que permite, previa identificación con certificado y firma electrónica con una tarjeta criptográfica, enviar notificaciones a profesionales desde los juzgados con efectos legales plenos.

El sistema Lexnet ha incluido posteriormente otros colectivos como la Abogacía del Estado y los Graduados Sociales y ha ampliado sus funcionalidades para que los profesionales puedan enviar escritos a los juzgados y los procuradores puedan comunicarse traslados de copias entre sí. Además ofrece interoperabilidad con otros sistemas de gestión procesal de las comunidades autónomas con competencias de justicia transferida.

El sistema Lexnet es un sistema de intercambio de documentos judiciales en formato electrónico, que funciona de manera similar al correo electrónico pero garantizando las premisas básicas de seguridad:

1. **Autenticación**: el emisor del documento es realmente quién dice ser.
2. **Confidencialidad y seguridad del sistema**, sólo el destinatario puede leer el contenido del documento.
3. **Integridad**, el contenido del documento no puede alterarse durante su transmisión.
4. **No repudio**, el emisor del documento no podrá negar el hecho de su envío.
5. **Fechado**, permite fijar el orden de llegada de escritos.

Este sistema ha traído un conjunto de no conformidades al cliente que se resumen en los siguientes:

1. El problema de la aplicación de la normativa supletoria.
2. Problemas generados por el tiempo de recepción de las notificaciones.
3. Problemas derivados por los traslados de copias.
4. Problemas en el cómputo de los plazos comunes.
5. La imposibilidad de traslado por la naturaleza del documento.

6. Notificaciones defectuosas.

El principal problema de la actual implementación de Lexnet es que para llevar a cabo el proceso de firma de la comunicación por quien la envía, utiliza un binario ActiveX (nativo de determinadas versiones de Windows y de Internet Explorer) que se ha de ejecutar localmente en el sistema operativo del usuario. Este hecho plantea problemas de evolución de la infraestructura informática, de falta de interoperabilidad con otras plataformas cliente como puedan ser MacOS o Linux, con otros navegadores como Firefox, Opera, Safari y Chrome, e incluso algunas versiones de Microsoft Internet Explorer, y con la imposibilidad de que su uso pueda ser automatizado por otros servicios informáticos, a modo de servicio web, sin mediar la intervención de un usuario para efectuar a mano la remisión del escrito, ya que generalmente dichos escritos ya vienen firmados por el usuario autorizado (juez, secretario, abogado) de otra aplicación como por ejemplo la del sistema procesal.

Existen testimonios profesionales de que el funcionamiento de la aplicación no es correcto cuando no consigue entregar la comunicación telemática a su destinatario final, y que puede ocasionar graves problemas jurídicos.(4)

- ✓ **SEINSIR (España)**

SEINSIR es un sistema informático de gestión judicial que permite integrar nuevas tecnologías (tratamiento de imágenes, bases de datos documentales, conexiones telemáticas, arquitectura cliente-servidor a tres niveles, entre otras) en los productos desarrollados para las oficinas judiciales de cualquier instancia y los servicios comunes relacionados, incrementando su nivel tecnológico y productivo y mejorando el servicio al ciudadano. Además permite generar aplicaciones para dar solución a las necesidades de registro, control y seguimiento de los asuntos judiciales que se resuelvan en un órgano judicial de cualquier tipo. Esto es posible por el sofisticado sistema de parametrización que incorpora y que permite la definición externa de la colección de datos y su representación y los listados, estadísticas y consultas necesarias para el funcionamiento informatizado del órgano judicial.

Este sistema ha sido concebido básicamente a partir del conocimiento de las necesidades del ámbito judicial de la comunidad Latinoamericana y de España.(5)

- ✓ **GIAJ-TRAMIX (Argentina)**

El sistema está planteado con arquitectura cliente/servidor para los usuarios internos del Poder Judicial, y arquitectura webenable con motor Oracle, tecnología propietaria, para los usuarios externos (abogados y otros justiciables). Las funcionalidades previstas incluyen la posibilidad para los profesionales de ingresar a sus expedientes y efectuar el control de las actuaciones, notificarse y también incorporar escritos en forma remota, es decir a través de Internet, todo mediante el uso de cualquier dispositivo que permita esta navegación. (6)

Este sistema genera importantes ventajas, que se traducen en avances relacionados con la agilización de los procesos, incremento de la tarea de oficio favorecida por la incorporación de procesos automatizados y el descongestionamiento de los circuitos administrativos, factores que permitirían proporcionar una operatoria más eficiente y eficaz, disminuyendo también los costos.(6)

- ✓ **SISPROP (Cuba)**

Sistema desarrollado en la Universidad Central de las Villas “Marta Abreu”, para abarcar las instancias de los tribunales Supremo y Provincial en la sala de lo penal. El sistema fue un fracaso pues no tuvo una buena aceptación por los clientes ya que presenta algunas no conformidades como la falta de validaciones en los datos suministrados por los usuarios. Además fue concebido en el lenguaje de programación Visual Pascal, con la herramienta de desarrollo Delphi usando a SQL Server para su base de datos, por lo que lo hace incompatible con el software libre.(7)

- ✓ **SISECO (Cuba)**

Sistema utilizado en la sala de lo económico provincial de La Habana desarrollado en el 2002 para el área de la estadística. El mismo es una aplicación de escritorio que fue desarrollada en Delphi 6 y con SQL server 2000 para la base de datos. El tribunal no cuenta con su código fuente y los autores no se encuentran en Cuba.

Este sistema ha resuelto los problemas estadísticos, pero desde el año 2010 viene presentando serios problemas, como por ejemplo que con el aumento del tamaño de la base de datos el sistema demora un poco más de cuatro minutos después de ser

ejecutada la aplicación, a pesar de que la estación de trabajo donde se realizó la revisión contaba con 1GB de memoria RAM y con un procesador DualCore a 2.93 Ghz. Los expedientes sólo pueden ser radicados hasta el año 2010 porque el campo de datos donde se selecciona “año de radicación” fue llenado de forma manual hasta dicho año y al no contarse con el código fuente no se le ha podido adicionar más años. Una alternativa para resolver este problema fue la implementación de un parche que se realizó en el año 2010 y que logró aumentarle un año más a este campo, pero el creador del parche no se encuentra trabajando ya en el Tribunal Supremo y se niega a dar el código fuente de esta solución para poder trabajar en el año 2012, lo que ha causado que se retorne a hacer las estadísticas de forma manual. Al ser un sistema de gestión estadística no resuelve los problemas planteados anteriormente.

A partir del estudio realizado a las soluciones informáticas que gestionan los procesos judiciales en el mundo, se demuestra que ningún sistema es adaptable al sistema judicial de los TPC, por ser software propietario, y necesitar una licencia para su uso, por los requerimientos que exigen para su ejecución, y en caso de los desarrollados y utilizados en Cuba porque de manera general tienen una nula comunicación entre las instancias, no superan las barreras del papel, están arraigados a los registros tradicionales (libros), y se puede hacer cualquier acción sobre el expediente sin tener en cuenta el estado en que se encuentra el mismo.

1.4 Aplicaciones web

Las aplicaciones Web están estrechamente ligada con Internet y soportada bajo los principios de una arquitectura cliente-servidor mediante la cual se separa el cliente, que a menudo es una aplicación que usa interfaz gráfica que inicia el diálogo o solicita los recursos del servidor que es quien responde las solicitudes. El cliente y el servidor pueden actuar como una sola entidad y también como entidades separadas, realizando actividades o tareas independientes, además el cliente no necesita conocer la lógica del servidor, sólo su interfaz externa, los cambios en el servidor implican pocos o ningún cambio en el cliente y que un servidor da servicios múltiples de forma concurrente.(8)

Al desarrollar el sistema como aplicación web se están aprovechando muchas de las ventajas que esto brinda como: evitar los problemas de compatibilidad pues solo se

necesita un navegador actualizado, no ocupa espacio en el disco duro de los clientes porque la aplicación se encuentra en un servidor y se accede a él mediante una dirección, donde se cargarán los cambios que se le hagan a la aplicación sin necesidad de tenerlos en el cliente.(8)

1.5 Metodologías de desarrollo de software

Las metodologías de desarrollo de software imponen un proceso a cumplir disciplinadamente para el desarrollo de un software cualquiera, con el fin de hacerlo más predecible y eficiente. Además tiene como principal objetivo aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo. No existe una metodología universal, pues toda metodología debe ser adaptada a las características de cada software exigiéndose así que el proceso sea configurable. La seleccionada por la envergadura de la aplicación y los procesos que la forman fue la siguiente.(9)

1.5.1 Proceso Unificado de Desarrollo (RUP)

Es una metodología de desarrollo de software que está basado en componentes e interfaces bien definidas y junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. (9)

Las características principales de RUP son:

- Guiada por casos de uso ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso.
- Centrado en la arquitectura pues la arquitectura describe los elementos del modelo que son más importantes para la construcción del software, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.
- Iterativo e incremental ya que la implementación del proyecto se realiza en Iteraciones, con lo que se pueden definir objetivos por cumplir en cada iteración y así poder ir completando todo el proyecto iteración por iteración.

RUP puede especializarse para una gran variedad de sistemas de software, en diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto. Además no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.(9)

En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales, los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como flujos de apoyo.

1. Modelo del negocio
2. Requisitos.
3. Análisis y Diseño.
4. Implementación.
5. Prueba (Testeo).
6. Instalación o despliegue.
7. Administración del proyecto.
8. Administración de configuración y cambios.
9. Ambiente.

Además de estar compuesta por 4 fases:

1. Concepción o Inicio.
2. Elaboración.
3. Construcción.
4. Transición.

En la siguiente figura se resume la relación que existe entre las fases y los flujos de RUP.

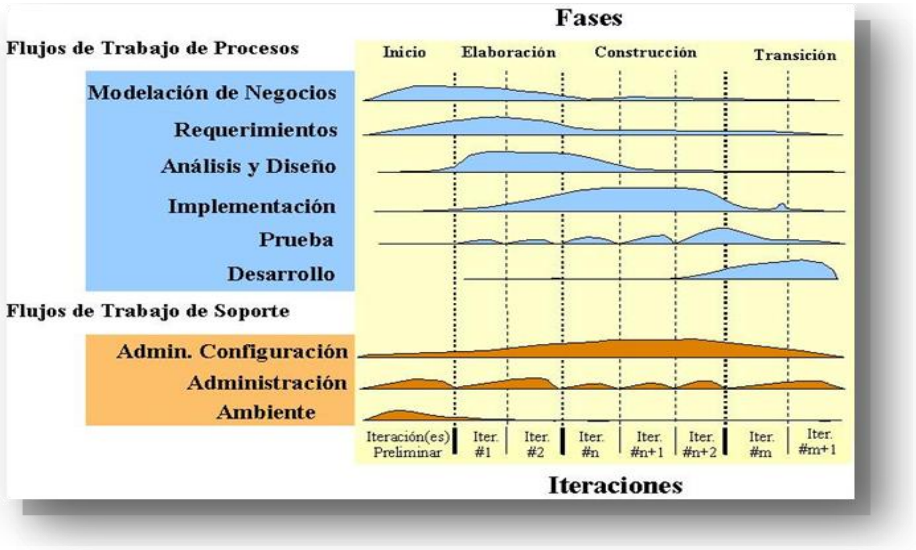


Figura 1. Metodología RUP

Esta investigación se centra en la **fase de construcción** que está compuesta por un ciclo de varias iteraciones, en las cuales se van incorporando sucesivamente los casos de uso, de acuerdo a los factores de riesgo del proyecto, de esta manera, se puede contar en forma temprana con versiones del sistema que satisfacen los principales casos de uso. Adentrándose en esta fase se obtiene el flujo de trabajo más importante de la misma y el cual es el principal objetivo de esta investigación, la **implementación,** que define cómo se organizan las clases y objetos en componentes del sistema, cuáles nodos se utilizarán y la ubicación en ellos de los componentes, así como la estructura en capas de la aplicación.

1.6 Arquitectura de software

En sus inicios, el desarrollo de software se definió como una tarea compleja para aquellos que se dedicaban al mundo de la tecnología. Con el paso de los años, se fue adquiriendo experiencias, desarrollándose nuevos métodos que mostraban la estructura, funcionamiento e interacción entre las partes del software. A todo esto, se le denominó Arquitectura de software.

Para confeccionar una Arquitectura de software se debe analizar detalladamente los objetivos y restricciones de la aplicación que se quiere desarrollar, pues estos son los

elementos fundamentales para la selección de las herramientas y tecnologías a usar en su desarrollo.

Para la confección de la arquitectura del proyecto TPC se examinaron un conjunto de características y objetivos que traería consigo el desarrollo del SIT, pues este debe garantizar fiabilidad, seguridad y eficiencia en la gestión de los procesos judiciales que se realizan en todos los tribunales de Cuba.

Para cumplir con las metas trazadas el equipo de arquitectos del proyecto concibió una arquitectura basada en un conjunto de marcos de trabajo, abordados más adelante, que posibilitan un desarrollo seguro, eficiente y controlado de los requisitos identificados que debe cumplir el producto final.

1.6.1 Marcos de trabajo

Un marco de trabajo es una estructura de software compuesta por componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework (o marco de trabajo) se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta, puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.(10)

A continuación se muestran los marcos de trabajo que fueron empleados para el desarrollo del sistema.

- ✓ **Zend Framework (ZF)**

ZF es un una herramienta de código abierto para desarrollar aplicaciones web y servicios web con PHP5 (Personal Home Page). Además es una implementación que usa código 100% orientado a objetos. La estructura de los componentes en este es algo único; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado.(11)

ZF ofrece un gran rendimiento y una robusta implementación patrón Modelo-Vista-Controlador, una abstracción de base de datos fácil de usar, y un componente de

formularios que implementa la prestación de formularios HTML, validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos.(11)

El departamento de tecnologías de la UCI en conjunto con la UCID (Unidad de Compatibilización Integración y Desarrollo de Software para la Defensa) constituyen el equipo de desarrollo de Zend Ext, el cual fue el resultado de la extensión de algunos componentes de ZF, con el objetivo de crear un marco de trabajo extensible y configurable centrado en el desarrollo de aplicaciones web, en la lógica del negocio y en las interfaces de usuario, alejando a los programadores de los detalles arquitectónicos, con soporte para entornos multi-entidad y para una arquitectura de sistema orientada a componentes.(12)

- ✓ **Doctrine**

Es un mapeador de objetos relacional (ORM por sus siglas en inglés) escrito en PHP que proporciona persistencia para objetos en este lenguaje. Está por encima de la capa de abstracción a la base de datos, uno de sus características es la posibilidad de escribir consultas a la base de datos a partir del tratamiento con objetos en PHP llamado Doctrine Query Language (DQL).(13)

- ✓ **ExtJS**

ExtJS es una librería Javascript que permite construir aplicaciones complejas e interactivas usando tecnologías como AJAX, DHTML (HTML dinámico) y DOM (Modelo de Objetos del Documento por sus siglas en inglés). Incluye un alto rendimiento, interfaces de usuario personalizables, bien diseñado y extensible modelo de componentes, una interfaz intuitiva y fácil de utilizar con licencias de código abierto y comercial. Ofrece la opción de crear validadores personalizados para las entradas de datos en los formularios de nuestra aplicación.(14)

- ✓ **Sauxe v2.0**

El Marco de Trabajo Sauxe se ha estructurado de manera tal que facilite la reutilización de los diferentes componentes y que sea de fácil entendimiento para todos los

programadores que desarrollen sobre el mismo. Sauxe utiliza el framework ExtJS v2.2, para implementar la capa de presentación, se apoya en Zend Ext v1.7.9 para el desarrollo de la lógica del negocio y para la gestión de los datos utiliza Doctrine v1.7.1. Este utiliza como patrón arquitectónico Modelo-Vista-Controlador (MVC). (12)

Sauxe implementa mecanismos de autenticación y autorización, mecanismos de mensajería y control de excepciones, gestión y configuración dinámica de precondiciones, poscondiciones y validaciones de variables, gestión y configuración de flujos de trabajo, visualización de las funcionalidades de un sistema, entre otras funcionalidades. (12)

Las herramientas empleadas en la construcción de Sauxe son herramientas libres, cumpliendo con la independencia tecnológica por la que tanto aboga el país, como desventaja, este marco de trabajo sólo maneja como sistema gestor de base datos el PostgreSQL 8.3 u 8.4, aunque consta de una gran organización de todos los elementos del proyecto, favoreciendo el desarrollo de componentes de forma organizada, pues cada componente tendrá una estructura bien definida en las diferentes capas (Capa de Presentación, Capa de Control o Negocio, Capa de Acceso a Datos, Capa de Datos, Capa de Servicios) que define en su arquitectura y facilita de esta manera las labores de mantenimiento del sistema a desarrollar. Su estructura es la siguiente:

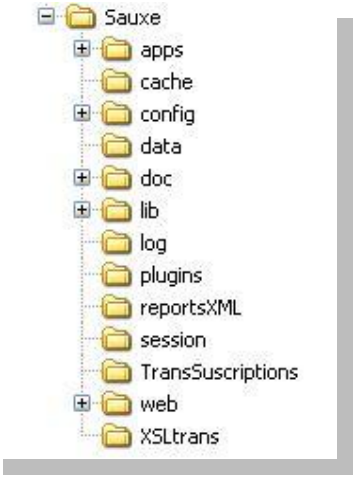


Figura 2. Estructura de las carpetas de Sauxe

Este marco de trabajo brinda un conjunto de ventajas significativas que han permitido el trabajo exitoso en el proyecto TPC ya que:

- Permite configurar y gestionar de manera dinámica la caché del marco de trabajo.
- Posibilita configurar y gestionar de manera dinámica la integración de las aplicaciones o dominios de soluciones que se instancien o construyan con la misma. (Países, Provincias, Municipios)
- Posee un mecanismo de abstracción de la capa relacional de persistencia, este mecanismo de mapeo relacional de objeto permite además, contar con un mecanismo de SQL Helper.
- Tiene un mecanismo de configuración y gestión dinámica de las características de concurrencia sobre entidades o dominios de la solución lógica que se modela, de manera que se puede configurar el esquema de concurrencia que se desee sobre las entidades o estructuras de entidades del esquema de persistencia de una solución específica.
- Ostenta un mecanismo de administración de transacciones, de manera que las operaciones de modificación sobre el modelo de dominio sean transaccionales por defecto.
- Brinda un componente (ZendExt_Exception) para el tratamiento de excepciones muy eficiente y de fácil manejo.
- Brinda un componente para validar reglas del negocio (ZendExt_Validation).
- Posibilita lograr fácilmente la interoperabilidad entre subsistemas y componentes mediante la utilización de un componente que ofrece Sauxe llamado ZendExt_IoC.
- Contiene un generador dinámico de reportes incluido.
- Fue desarrollado en la universidad.
- Trae integrado un componente para la seguridad **Acaxia**, el cual realiza el control acceso mediante un modelo RBAC.
- Tiene un mecanismo de administración y configuración dinámica de trazas de la solución.(12)

1.6.2 Patrones de arquitectura

Atendiendo a los objetivos y restricciones de la solución, se optan por los siguientes estilos o patrones que tipifican la arquitectura del sistema:

✓ **Patrón Modelo-Vista-Controlador (MVC)**

Este patrón separa conceptualmente la representación visual de la aplicación, las acciones que intercambian datos y el modelo de negocio y su dominio. En el sistema se concreta con la identificación de 3 elementos diferentes: **la vista** implementada en Java Script o HTML[1] reside del lado del cliente en tiempo de ejecución, **el modelo** que junto **al controlador** reside del lado del servidor, la interacción entre la vista y el controlador se realiza a través de una solicitud AJAX[2] y XML[3] o peticiones HTML y la respuesta dada por el controlador puede encontrarse en JSON[4], XML o HTML según corresponda la solicitud.(15)

✓ **Patrón Cliente – Servidor**

Se caracteriza por la existencia de uno o más nodos servidores donde se encuentra el servicio que se expone y varios nodos clientes que consumen este servicio. En el sistema, este estilo responde al hecho de que se trata de una aplicación web y se concreta con un servidor de bases de datos (PostgreSQL v8.4), un servidor web (con función de servidor de aplicaciones, Apache v2.0) y varios clientes que acceden al sistema a través de un navegador. (15)

1.6.3 Patrones de diseño

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

[1] Hipertext Markup Language.
[2] JavaScript asíncrono.
[3] Xtensible Markup Language por sus siglas en inglés.
[4] Acrónimo de JavaScript Object Notation.

implementadas de esta funcionalidad común (frameworks) y que nos permiten centrarse en desarrollar sólo la funcionalidad específica requerida por cada aplicación y además, da mejor imagen de profesionalidad y calidad.(16)

Los patrones GRASP[5] describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Este grupo contiene patrones como:

- Experto.
- Creador.
- Alta cohesión.
- Bajo acoplamiento.
- Controlador.
- Polimorfismo.
- Fabricación pura.
- Indirección.
- No hables con extraños (Ley de Demeter).(16)

Los patrones GOF[6] son 23 y se clasifican según su propósito en Creacionales, Estructurales y de Comportamiento y según su ámbito en Objeto y de Clase. Los mismos tienen como características que:(16)

- Son soluciones concretas, o sea, proponen soluciones a problemas concretos y no a las teorías genéricas.
- Son soluciones técnicas porque indican resoluciones técnicas basadas en Programación Orientada a Objetos (POO).
- Se utilizan en situaciones frecuentes ya que se basan en la experiencia acumulada al resolver problemas reiterativos.
- Favorecen la reutilización de código.(16)

[5] Patrones Generales de Software para Asignación de Responsabilidades por sus siglas en inglés
[6] Grupo de los Cuatro (GOF siglas de Gang of Four en inglés)

1.6.4 Lenguajes de programación

Un lenguaje de programación es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. (17)

Está formado de un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama programación.(17)

Como esta investigación es sobre el desarrollo de una aplicación web, existen lenguajes de programación que responden del lado del cliente y otros del lado del servidor, que permiten informatizar todas las funcionalidades con que debe contar el sistema.

1.6.4.1 Lenguaje de programación del lado del cliente

Son aquellos lenguajes de programación capaces de ser interpretados directamente por el navegador sin necesidad de un pre-tratamiento. En la aplicación estos fueron los utilizados:(8)

- ✓ **JavaScript**

Es un lenguaje de programación interpretado y muy fácil, pues no hace falta tener conocimientos avanzados de programación para poder hacer un programa en JavaScript. Maneja objetos dentro de la página web los cuales facilitan la programación de páginas interactivas, a la vez que se evita la posibilidad de ejecutar comandos que puedan ser peligrosos para la máquina del usuario, tales como el formateo de unidades y la modificación archivos. Es un lenguaje dinámico que responde a eventos en tiempo real como presionar un botón, pasar el puntero del mouse sobre un determinado texto o el simple hecho de cargar la página o caducar un tiempo. Se utiliza esencialmente del lado del cliente, implementado como parte del navegador web permitiendo mejoras en la seguridad significativas puesto que es muy eficiente en los temas de validaciones.(18)

- ✓ **CSS**

Las Hojas de estilo en cascada, en inglés Cascading Style Sheets (CSS), fueron diseñadas y desarrolladas por la World Wide Web Consortium (W3C). Una CSS es el tipo de documento que utiliza un navegador web para redefinir las propiedades de los distintos elementos y las etiquetas en el código HTML. Permite dar formato a los documentos de forma global, le proporcionan al diseñador de páginas web definir un conjunto de ampliaciones HTML especiales y aplicarlas al documento, provee la especificación e intercambio de los fondos para textos y documentos, así como sus tipos y tamaños de fuente. Las definiciones del formato de un documento se pueden colocar en archivos separados y aplicarlas a un grupo de documentos. Posibilitan además aplicar un formato modificado a archivos HTML ya existentes, con los que se puede aplicar a un documento diferentes estilos de orígenes, constituyendo una herramienta poderosa para el diseño de documentos HTML.(19)

- ✓ **HTML**

El HTML es el lenguaje con el que se escriben las páginas web. Es un lenguaje de hipertexto que permite escribir texto de forma estructurada, y que está compuesto por etiquetas, que marcan el inicio y el fin de cada elemento del documento.(20)

Un documento hipertexto no sólo se compone de texto, puede contener imagen, sonido, video y otros, por lo que el resultado puede considerarse como un documento multimedia. Estos deben tener la extensión html o htm, para que puedan ser visualizados en los navegadores.(20)

1.6.4.2 Lenguaje de programación del lado del servidor

Los lenguajes de lado del servidor son aquellos reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él. Los utilizados son:(21)

- ✓ **PHP v5.2.5**

PHP es un lenguaje interpretado de propósito general ampliamente usado, diseñado especialmente para desarrollo web y que puede ser incrustado dentro de código HTML.

Generalmente se ejecuta en un servidor web, tomando el código PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. En resumen:

- Es un lenguaje multiplataforma.
- Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- El código fuente escrito en PHP es invisible al navegador y al cliente porque es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- Tiene una capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con PostgreSQL.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- Tiene manejo de excepciones.(22)

1.7 Herramientas de desarrollo del Software

Para el desarrollo del SIT, el equipo de arquitectura seleccionó un conjunto de aplicaciones y herramientas para utilizarlas en el desarrollo de todas las fases y flujos de la metodología escogida. Para el proceso de la implementación del software se inclinaron por las siguientes:

1.7.1 Herramientas CASE

Las herramientas CASE[7], son el conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores para aumentar la calidad del software reduciendo el esfuerzo, el costo y el tiempo. Además de estructurar la documentación asociada a los artefactos generados, facilitan el desarrollo de software desde la

[7] Ingeniería de Software Asistida por Ordenador por sus siglas en inglés.

planificación, pasando por el análisis y el diseño hasta la generación del código fuente de los programas.

Para la fase de implementación se seleccionaron algunas para apoyar el trabajo de los desarrolladores.(23)

✓ **Visual Paradigm v6.4**

Es una herramienta UML (Unified Modeling Language en inglés) profesional que soporta el ciclo de vida completo de desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Le permite dibujar a los desarrolladores todos los tipos de diagramas que necesitan utilizar para la construcción del software, como el Diagrama de Componentes y el de Despliegue. Dentro de sus características fundamentales que sirvieron para su selección están:

- **Multiplataforma:** Soportado en plataformas Java para sistemas operativos Windows, Linux (es la que utiliza el equipo de desarrollo) y Mac.
- **Generación de documentos:** Comparte y genera los diagramas y diseños en formatos como PDF, HTML y Microsoft Word.
- **Ingeniería de código:** Permite generación de código e ingeniería inversa en lenguajes como Java, C++, CORBA, IDL, PHP, XML Schema, Ada, Python, C#, VB .NET, ODL, Flash Action Script, Delphi, Perl y Rugby.
- **Integración con entornos de desarrollo:** Apoyo al ciclo de vida completo de desarrollo del software: análisis, diseño e implementación en IDE como NetBeans.
- **Modelado de bases de datos:** Generación de bases de datos, conversión de diagramas entidad-relación a tablas de base de datos, mapeos de objetos y relaciones, ingeniería inversa desde gestores de bases de datos.(24)

Teniendo en cuenta las características mencionadas se decidió utilizar el Visual Paradigm por ser además un producto de calidad, que soporta aplicaciones web en varios idiomas, además de tener compatibilidad entre todas sus ediciones. Es fácil de utilizar, en este caso, sería para el diseño de los diagramas correspondientes a la fase de implementación.

1.7.2 Servidor Web

Un servidor Web es un programa que está diseñado para transferir hipertextos, páginas Web o páginas HTML: textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música. Además sirve de contenido estático a un navegador, carga un archivo y lo sirve a través de la red al navegador de un usuario. Este intercambio es mediado por el navegador y el servidor que hablan el uno con el otro mediante HTTP[8].

- ✓ **Servidor HTTP Apache v2.0**

Apache es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, entre otros), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Es utilizado para que la aplicación pueda ejecutarse en las estaciones de trabajo.(25)

1.7.3 Sistema gestor de bases de datos

Los sistemas de gestión de bases de datos (SGBD) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipularlos, garantizando la seguridad e integridad de los mismos.(26)

- ✓ **PostgreSQL v8.4**

Es un sistema de gestión de bases de datos relacional orientada a objetos, libre y multiplataforma, publicado bajo la licencia BSD (Berkeley Software Distribution). Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una sola empresa sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es

[8] Hypertext Transfer Protocol

denominada el PGDG (PostgreSQL Global Development Group). Tiene la extraordinaria potencialidad de permitir que mientras un proceso escribe en una tabla, otros accedan a la misma sin necesidad de bloqueos esto es posible gracias a un sistema denominado Acceso Concurrente Multiversión.(27)

1.7.4 Herramienta de desarrollo colaborativo

Para el trabajo de desarrollo de un software entre un conjunto determinado de personas se necesitan herramientas que permitan dar información de su contenido para continuar con el trabajo, teniendo en cuenta que se cohesione el trabajo del equipo de desarrollo, para el SIT se escogió el siguiente:

- ✓ **SVN (SubVersion) v1.5**

Es un sistema de control de versiones, que mantiene los registros de todos los cambios que se han realizado a los archivos de un software, lo que permite el trabajo de distintos desarrolladores en un mismo proyecto, esta herramienta es muy usada por los programadores de software libre. Existen dos razones fundamentales para su uso:

- Gestiona las modificaciones durante el desarrollo.
- Permite que varias personas trabajen sobre los mismos ficheros.(28)

A pesar que el equipo de desarrollo se divide el trabajo, pueden ocurrir conflictos sobre los mismos ficheros, que esta herramienta resuelve eficientemente, además genera las trazas sobre los ficheros que han sido modificados, agregados y eliminados, para un mejor control de lo que va sucediendo con el software.

1.7.5 IDE de desarrollo

Un IDE[9] proporciona un marco de trabajo amigable para una gran cantidad de lenguajes de programación, donde es posible que un mismo entorno de desarrollo tenga la posibilidad de utilizar varios lenguajes de programación. Eclipse, ZendStudio y Netbeans

[9] IDE: Entorno de Desarrollo Integrado o Integrated Development Environment en inglés.

son entornos de desarrollo integrados que pueden utilizar PHP como lenguaje de programación.

✓ **Netbeans v7.0.1**

Por ser un proyecto de código abierto, el equipo de arquitectura decidió usar como IDE el Netbeans en su versión 7.0.1 por soportar lenguajes dinámicos como PHP y Java Script, además tiene integración con el subversión, es multiplataforma, tiene una interfaz muy amigable, tiene todas las herramientas para crear aplicaciones profesionales ya sean de escritorio, empresariales, web, móviles y aplicaciones SOA, no sólo en Java sino también en C/C++ y Ruby.(15)

1.7.6 Navegadores

Un navegador web o explorador web es una aplicación que permite al usuario recuperar y visualizar documentos de hipertexto, comúnmente descritos en HTML, desde servidores web de todo el mundo a través de internet. Cualquier navegador actual permite mostrar o ejecutar gráficos, secuencias de vídeo, sonido, animaciones y programas diversos además del texto y los hipervínculos o enlaces. En este caso el explorador usado es Mozilla Firefox.

✓ **Mozilla Firefox**

Es un navegador web libre, multiplataforma y está disponible en varias versiones de Microsoft Windows, GNU/Linux y algunos sistemas basados en Unix. Incluye navegación por pestañas, corrector ortográfico, búsqueda progresiva, marcadores dinámicos, un administrador de descargas y un sistema de búsqueda integrado.

Seguidamente se mencionan algunas características de Firefox frente a Internet Explorer:

- **Rapidez:** La velocidad de Firefox e Internet Explorer (IE) es, en términos generales, casi la misma. Aunque si se manejan muchas páginas web a la vez, se notará una mayor estabilidad con Firefox que con IE.
- **Compatibilidades:** Se ofrecen características DHTML (HTML dinámico) para crear interfaces de usuario eficaces para aplicaciones web. Se proporciona también integración con la plataforma .NET, simplifica la integración del código de

servidor y cliente y permite que las aplicaciones llamen a funciones del servidor de manera asincrónica.

- **Seguridad y control de privacidad:** Las mejoras no son excesivamente significativas en cuanto a la apariencia externa, pero el cambio interno es importante. Una muestra de ello son las mejoras en seguridad respecto a las versiones anteriores, que tapan gran cantidad de agujeros de seguridad.(29)

Junto al navegador utilizado para la ejecución de la aplicación, se utiliza para la implementación uno de los complementos del mismo, que facilita significativamente en el desarrollo del sistema, por las diferentes prestaciones que brinda.

✓ **Firebug**

Es un paquete de utilidades con el que se puede analizar (revisar velocidad de carga, estructura DOM), editar, monitorizar y depurar el código fuente, CSS, HTML y JavaScript de la página web de manera instantánea, de mucha ayuda a los desarrolladores, en la prueba de la implementación local que están realizando. Entre sus características principales se encuentran:

- Desarrollo de CSS.
- Explora el DOM.
- Busca errores.
- Desarrollo de HTML.
- Línea de comandos para JavaScript.
- Depuración de JavaScript.
- Registro de eventos JavaScript.
- Monitor de Red.
- Layout CSS.

1.8 Pruebas de software

Al concluirse la implementación del sistema, se debe realizar un conjunto de pruebas que garanticen la calidad del producto final, antes de ser entregado al cliente. Las mismas evalúan a la aplicación comprobando que cumpla con los requisitos que inicialmente se plantearon como necesidades del cliente por parte de los analistas del proyecto. Una prueba eficiente es aquella que detecta al menos un error, por lo que a la aplicación se le realizarán las siguientes:

✓ **Pruebas de caja negra**

Las pruebas de caja negra se refieren a las pruebas que se llevan a cabo sobre la interfaz del software. 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 un resultado correcto, así como que la integridad de la información externa se mantiene. Las pruebas de caja negra permiten encontrar errores tales como:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.(30)

✓ **Pruebas de caja blanca**

Las pruebas de caja blanca se basan en el diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivarlos. Mediante la prueba de la caja blanca se pueden obtener casos de prueba que:

- Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
- Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
- Ejecuten todos los bucles en sus límites operacionales.
- Ejerciten las estructuras internas de datos para asegurar su validez. (30)

1.9 Conclusiones parciales

Al finalizar este capítulo se llega a la conclusión de que ningún sistema de gestión procesal de los existentes en el mundo, ni los utilizados en Cuba resuelve los casos de uso definidos por los analistas del proyecto TPC. Además que la metodología, las herramientas 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.
- Modelo-Vista-Controlador como patrón de arquitectura.
- Visual Paradigm v6.4 como herramienta CASE para el modelado.
- PHP v5.2.5 como lenguaje de programación.
- Netbeans v7.0.1 como entorno de desarrollo.
- SAUXE v2.0 como framework de desarrollo.
- PostgreSQL v8.4 como servidor de Base de Datos.
- Apache v2.0 como servidor Web.
- Para el control de versiones el SubVersion v1.5.
- Mozilla Firefox como navegador para ejecutar la aplicación con Firebug como complemento para la ayuda en la implementación.

Por último, con las pruebas que se le realizarán al sistema se garantizará que la aplicación cumpla con la implementación de los casos de uso planteados por el equipo de análisis del proyecto TPC.

Capítulo 2: Descripción de la solución propuesta

2.1 Introducción

El procedimiento Diligencias Previas se inicia cuando la parte promovente realiza una demanda en la sala de lo económico de los TPC trayendo consigo la formación de un nuevo expediente por cada una de ellas. Diariamente se radican en la sala un promedio de 30 expedientes por cada secretaria que trabaja en la misma, provocando que todos no puedan ser gestionados en el término que establece la ley.

Para agilizar el procedimiento Diligencias Previa, sería de gran ayuda contar con una solución informática que asegure el control de este proceso y permita la seguridad de los datos que se manejan, o sea, la disponibilidad, integridad y confidencialidad de la información.

En este capítulo se lleva a cabo una descripción de la propuesta del sistema, sus principales funcionalidades, se demostrará el uso de algunos patrones de diseño definidos en el capítulo anterior así como el modelado de los diagramas de componentes y de despliegue. Comprenderá además la definición de los estándares de codificación, la implementación y la representación de la estrategia de integración entre los componentes que están relacionados con el sistema.

2.2 Principios del sistema

Durante la etapa de inicio se definieron un conjunto de principios que el sistema debía cumplir que determinarán si su implementación fue efectiva o no, siendo estos los principales parámetros a medir para comprobar su resultado:

- Los datos deben ser captados por quien los genera.
- Ningún dato debe teclearse más de una vez.
- Deben validarse los datos obligatorios.
- El sistema debe mostrarle al usuario correspondiente el listado de los procesos que tiene en trámite, permitiéndole además la opción de filtrar según los estados en que pueda estar el expediente.
- Realizar la generación automática de todas las resoluciones (providencias, autos y sentencias) así como los oficios, citaciones, despachos y notificaciones.

- Los números de radicación de los expedientes, autos, resoluciones y sentencias deben ser asignados automáticamente por el sistema.
- El sistema debe permitir el acceso a las funcionalidades definidas para cada rol y llevar el control de la consecutividad del proceso en cada uno de los expedientes.

2.3 Arquitectura del SIT

El desarrollo del SIT está basado en una arquitectura en **n capas**, la que tiene como principal ventaja que en el caso de que exista algún error o la necesidad de algún cambio obligatorio, solo es necesario cambiar el nivel en cuestión, sin afectar el correcto funcionamiento del resto del sistema. Además se emplea el patrón arquitectónico Modelo–Vista–Controlador (MVC), siendo de los más utilizados en las aplicaciones web, permitiendo una organización e integración como se refleja en la figura 3. (15)

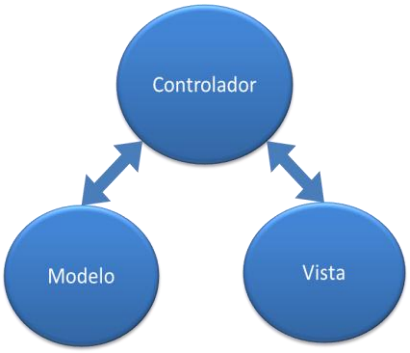


Figura 3. Integración del Patrón Modelo-Vista-Controlador

El marco de trabajo Sauxe permite una estructura de organización para cada una de las capas definidas anteriormente. En el paquete de clases que está ubicado en la dirección “TPC/apps/economico/models”, se encuentran las clases del modelo que representan la lógica del negocio, como se muestra en la figura 4. En el paquete “bussines” (negocio) se encuentran las clases “Model” (modelo) que son las encargadas de gestionar los objetos de las clases entidades del modelo de datos. En el paquete “domain” (dominio) se encuentran las clases que heredan de sus respectivas clases base que se encuentran en el paquete “*generated*” que también pertenece al paquete “domain”.

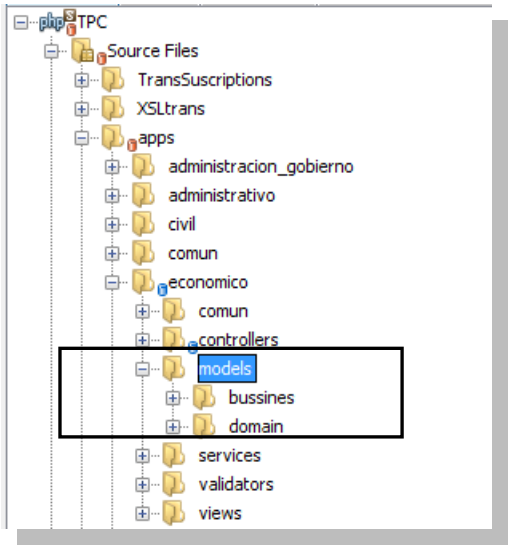


Figura 4. Capa del Modelo

Los controladores que se encuentran en la dirección "TPC/apps/economico/controllers", son los encargados de realizar las peticiones al modelo en caso de ser necesario y enviar una respuesta a la vista correspondiente.

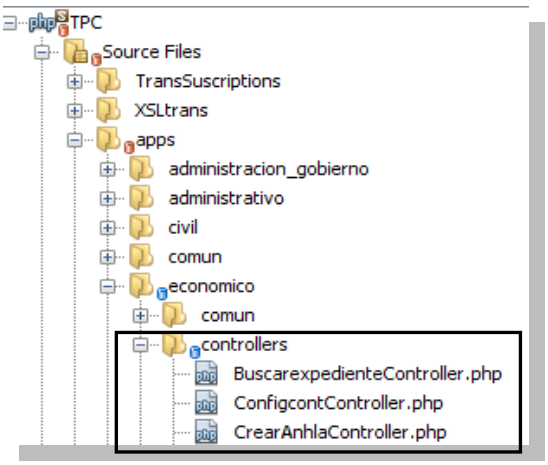


Figura 5. Capa del Controlador

La forma de integrarse el Controlador con la Vista, es mediante los archivos con extensión .phtml que se encuentran en el paquete "TPC/apps/economico/views", donde se define la dirección a los archivos .js y .css siendo los primeros los correspondientes a cada controlador y que se encuentran en el paquete

"TPC/web/economico/views/js" y los segundos (CSS), definidos por el marco de trabajo Sauxe que no han sido modificados para lograr una uniformidad en la Vista.

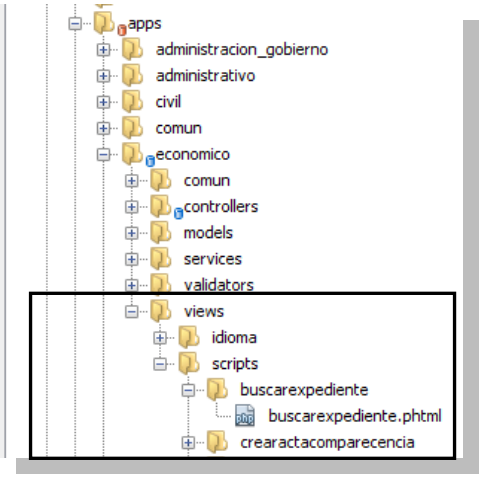


Figura 6. Capa de la Vista paquete apps

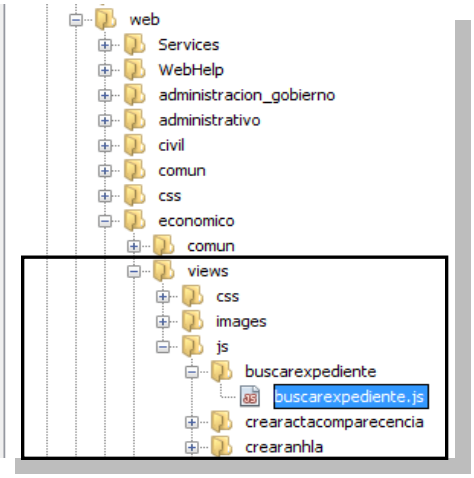


Figura 7. Capa de la Vista paquete views

2.4 Patrones utilizados

Durante el desarrollo del capítulo 1 se realizó un estudio de los patrones de diseño GRASP y GOF, que durante este capítulo se demostrará la utilización de alguno de ellos que se pusieron de manifiesto durante la implementación:

✓ **Patrón bajo acoplamiento**

Se puede decir que existe bajo acoplamiento cuando la dependencia entre las clases de un sistema es muy poca, con hacer cambios en alguna de las clases no se tenga que reprogramar nuevamente ni redefinir los objetos que son utilizados en ellas. En el desarrollo del procedimiento Diligencias Previas se pone de manifiesto el uso de este patrón: en el momento en que se consume un servicio web que brinda otro subsistema, no se conoce por parte del consumidor como es que está confeccionado el mismo, pero si los datos que devuelve. Cuando se construye un servicio web se realizan llamadas a diferentes métodos implementados dentro del subsistema, que de no existir el servicio, provocaría una alta dependencia entre las clases del sistema.

✓ **Patrón inversión de control (IoC)**

La inversión de control es un patrón de diseño pensado para permitir un menor acoplamiento entre componentes de una aplicación y fomentar así la reutilización de los mismos. Este patrón es utilizado por el marco de trabajo Sauxe para crear y brindar servicios mediante un fichero **ioc-general.xml** que se representa en la figura 8. Cada módulo del proyecto cuenta con un esquema propio en la base de datos y cuando se necesite acceder al esquema de otro, se crea y brinda un servicio para que el que lo necesite lo consuma.

Sauxe a través del patrón IoC, define la creación de clases, para de ellas brindar los servicios necesarios. En la siguiente figura se muestra la clase VisorServices ubicada en el paquete apps/económico/services donde se crea la función insertarDocumento.

```
3  class VisorServices {
4
5      public function insertarDocumento($body, $nombre, $definitivo) {
6          try
7          {
8              $mydocument = new Dedocumentogenerado();
9              $a = getdate();
10             $fechacreado = $a['year'] . '-' . $a['mon'] . '-' . $a['mday'];
11             $mydocument->nombre = $nombre;
12             $mydocument->documento = $body;
13             $mydocument->fechacreacion = $fechacreado;
14             $mydocument->definitivo = $definitivo;
15             DdocumentogeneradoModelEco::insertardedocumentogeneradoE($mydocument);
16             $_SESSION['iddocumentogenerado'] = $mydocument->iddocumentogenerado;
17         }
18         catch (Exception $es)
19         {
20             $msg = $es->getTraceAsString();
21             echo"{'codMsg':3,'mensaje': '.$msg.'}";
22         }
23     }
```

Figura 8. Representación de la clase VisorServices

```
1  <economico src="economico">
2
3      <VerDetallesExpediente reference="">
4          <inyector clase="VerdocumentosService" metodo="verDetallesExpediente" />
5          <prototipo>
6              <parametro numExpediente="tipo" tipo="string" />
7              <resultado tipo="array" />
8          </prototipo>
9      </VerDetallesExpediente>
10
11     <BuscarDocumentos>
22
23     <VerDocumento>
33
34     <ObtenerProcedimientos>
41
42     <ListarTipo>
49
50     <insertardocumento>
58
59     <actualizarDocumento>
66
67     <admitirEscritoPoscondiciones>
75
76     <insertarANHL>
84
85     <vistapreviasusanacion>
93
```

Figura 9. Fichero ioc-general.xml

La siguiente figura muestra la función ListarPaisesAction() en la clase RegistrarEscritoPromocionalController.php del módulo Económico, la cual consume el servicio ListarPaises brindado por el módulo Común.

```
function ListarPaisesAction() {
    $integrator = ZendExt_IoC::getInstance();
    echo json_encode($integrator->moduloComun->ListarPaises());
}
```

Figura 10. Función ListarPaisesAction()

✓ **Patrón singleton**

Este patrón es el encargado de garantizar que una clase sólo tenga una instancia y proporciona un punto de acceso global a ella, la única instancia debería ser extensible mediante herencia y los programadores deberían ser capaces de usar una instancia extendida sin modificar su código. En el sistema se pone de manifiesto este patrón al integrarse con los servicios web de otros módulos, en la figura 9 se pone de manifiesto la utilización de esta solución en el CU: RegistrarEscritoPromocional para un mejor entendimiento de la misma.

✓ **Patrón controlador**

El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, recibiendo los datos del usuario y enviando a las distintas clases según el método solicitado. Este patrón sugiere que la lógica del negocio debe estar separada de la capa de presentación y así aumentar la reutilización de código y a la vez tener un mayor control, además dividir los eventos del sistema en el mayor número de controladores, para poder aumentar la cohesión y disminuir el acoplamiento. En la siguiente figura se muestra un ejemplo de un controlador para un mejor entendimiento del mismo.

```php
<?php
/*
 * @package TPC
 * @copyright TPC
 * @autor David Estévez Díaz
 * @autor Pedro Enrique Romero Manfugas
 * @version 1.0.0
 */

class NotificarController extends ZendExt_Controller_Secure {

    function init() {...}

     function NotificarAction() {...}

     public function ComboPartesDPAction() {...}

     public function GetPartesNotificar($numero){...}

     public function ListarExpedientesAction() {...}

     public function CargarStoreParteAction() {...}

     public function NotificaryaAction() {...}

     public function VerDocumentoHtmlAction() {...}
}
?>
```

Figura 11. Clase controladora NotificarController

2.5 Estándares de codificación

Para una mejor organización del código fuente se determinó utilizar estándares de codificación que permiten una implementación uniforme, pues si en cualquier momento otro desarrollador necesita el código fuente para comprender la lógica de la implementación realizada, este sea de fácil entendimiento, además estos estándares definen el formato de código para los siguientes puntos:

- ✓ **Indentación**

Por indentación se entiende mover un bloque de texto hacia la derecha insertando espacios o tabuladores para separarlo del texto adyacente. La indentación se utiliza para mejorar la legibilidad del código fuente por parte de los programadores, teniendo en cuenta que los compiladores o intérpretes raramente consideran los espacios en blanco entre las sentencias de un programa.

En la indentación del código fuente del SIT se utilizaron los **tabs**, y no los espacios en blanco por ser más cómodos y precisos. En la figura 12 se muestra un ejemplo de un código indentado.

```php
function BuscarPersonaNaturalAction() {
    $formulario = json_decode(stripcslashes($this->_request->get('formulario')));
    $primerNombre = $formulario->primerNombre;
    $segundoNombre = $formulario->segundoNombre;
    $primerApellido = $formulario->primerApellido;
    $segundoApellido = $formulario->segundoApellido;
    $carnet = $formulario->carnet;
    $start = $this->_request->getPost("start");
    $limit = $this->_request->getPost("limit");
    $idPromoventeGrid = $this->_request->getPost("idpromovente");
    $idPdeudorGrid = $this->_request->getPost("idpdeudor");
    $idPromoventeNew = $_SESSION['idPromovente'];
    $idPresuntoDeudorNew = $_SESSION['idPresuntoDeudor'];
    $idPersonaPromovente = $idPromoventeNew + $idPromoventeGrid;
    $idPersonaDeudora = $idPresuntoDeudorNew + $idPdeudorGrid;
    $arregloPersonasEscogidas = array();
    $contador = 0;
    if ($idPersonaPromovente != 0)
        $arregloPersonasEscogidas[$contador++] = $idPersonaPromovente;
    if ($idPersonaDeudora != 0)
        $arregloPersonasEscogidas[$contador] = $idPersonaDeudora;
    $resultado = DpersonanaturalExt::BuscarPersonaNatural($limit, $start, $primerNombre,
          $segundoNombre, $primerApellido, $segundoApellido, $carnet, $arregloPersonasEscogidas);
    $natural = array();
    for ($i = 0; $i < count($resultado->items); $i++) {
        $natural[$i]['ci'] = $resultado->items[$i]['pasaporte'];
        $natural[$i]['idpersona'] = $resultado->items[$i]['idpersona'];
        $natural[$i]['primernombre'] = $resultado->items[$i]['primernombre'];
        $natural[$i]['segundonombre'] = $resultado->items[$i]['segundonombre'];
        $natural[$i]['primerapellido'] = $resultado->items[$i]['primerapellido'];
        $natural[$i]['segundoapellido'] = $resultado->items[$i]['segundoapellido'];
        if ($natural[$i]['ci'] == null)
            $natural[$i]['ci'] = $resultado->items[$i]['ci'];
    }
    echo json_encode(array('total' => count($natural), 'datos' => $natural));
    return;
}
```

Figura 12. Función BuscarPersonaNaturalAction()

- ✓ **Cabecera del archivo**

Es importante que todos los archivos .php inicien con una cabecera específica que indique información de la versión y autor de los últimos cambios. Es de cada equipo decidir si se quieren o no agregar más datos.

En la figura 13 se muestra un ejemplo de cómo debe quedar la cabecera de los archivos:

```php
<?php
/*
 * @package TPC
 * @copyright TPC
 * @autor David Estévez Díaz
 * @autor Pedro Enrique Romero Manfugas
 * @version 1.0.0
 */
```

Figura 13. Cabecera de clase .php

✓ **Comentarios en las funciones**

A todas las funciones se le debe escribir un comentario, antes de su declaración, describiendo que realizan en su implementación para que ningún programador tenga que analizar el código de una función para conocer su utilidad. Con el nombre de la clase y el comentario que la acompaña debe bastar para esto.

✓ **Clases**

Las clases serán colocadas en un archivo .php, donde sólo se colocará el código de la clase. El nombre del archivo será el mismo que el de la clase y siempre empezará en mayúscula. Mientras sea posible, tratar que los nombres de las clases tengan una sola palabra. Las clases siguen las mismas reglas de las funciones, por tanto, debe colocarse un comentario antes de la declaración de la clase explicando su utilidad.

✓ **Rutinas y Métodos**

- Los métodos comenzarán su nombre con mayúscula y si es más de una, cada palabra comenzará con mayúscula.
- Un buen nombre para una función o método es aquel que describe todo lo que la rutina hace.
- Es recomendable que los nombres de los métodos comiencen con un verbo, seguido del objeto al que afecta. Por ejemplo: RechazarEscrito, SubsanarEscrito, InsertarPersona.
- Cuando existan grupos de funciones que realicen operaciones similares con pequeñas diferencias, se deberá establecer un sistema de creación de nombres coherente donde además del objeto que afecta se especificará en qué lugar del negocio actúa. Por ejemplo: PasarDefinitivoDesistimiento, PasarDefinitivoSubsanación.

✓ **Números de métodos y grados de responsabilidad**

El número de métodos de una clase debe ser inferior a 40. Esta regla debe aplicársele a todas las clases del proyecto.

✓ **Nombre de variables**

- Excepto las constantes, todas las instancias y variables de clase o método empezarán con minúscula. Las palabras internas que lo forman (si son

compuestas) empiezan con su primera letra en mayúsculas (datosExpediente) utilizando el estilo lowerCamelCase.

- No se utilizarán nombres de variables que puedan ser ambiguos. Por ejemplo “col” es un nombre ambiguo ya que puede ser columna o color.
- Las variables booleanas deben tener nombres que sugieran respuestas o contenidos de tipo S/N, por ejemplo: encontrado, correcto, realizado.
- Los nombres de las variables booleanas deben ser positivos, por ejemplo: encontrado en lugar de noEncontrado.
- Se introducirán variables booleanas temporales cuando en una estructura de control (if, case, while) la expresión sea excesivamente compleja.

2.6 Modelo de implementación

El modelo de implementación está conformado por los **diagramas de despliegue y componentes**, que son artefactos generados durante el flujo de trabajo de implementación ya que es en este flujo de trabajo donde se describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue.

2.6.1 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 los elementos que lo conforman. El uso más importante de estos diagramas es mostrar la estructura de alto nivel del modelo, especificando los subsistemas de implementación y sus dependencias.

En la figura 14 se muestra el diagrama de componentes para el sistema.

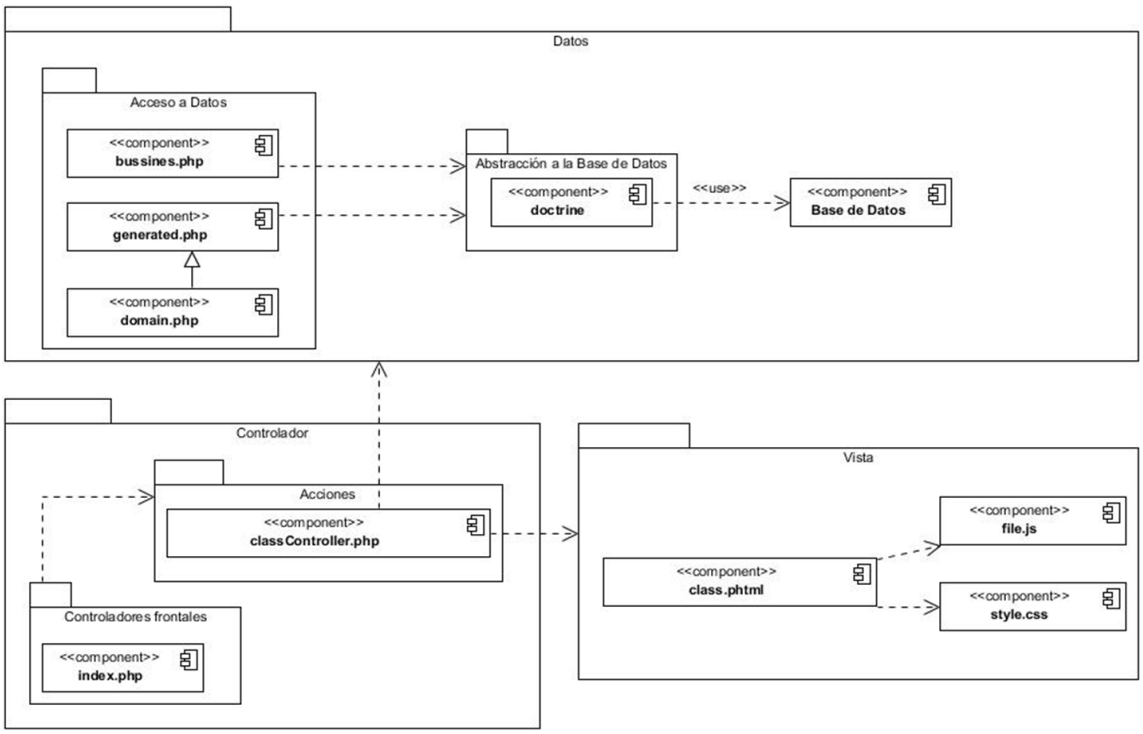


Figura 14. Diagrama de componentes

2.6.2 Diagrama de despliegue

El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo.

Podemos observar lo siguiente sobre el modelo de despliegue:

- Cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo de hardware similar.
- Los nodos poseen relaciones que representan medios de comunicación entre ellos, tales como internet, intranet, bus y similares.

El propósito del modelo de despliegue es capturar la configuración de los elementos de procesamiento y las conexiones entre estos elementos en el sistema. Permite el mapeo de procesos dentro de los nodos, asegurando la distribución del comportamiento a través de aquellos nodos que son representados.

En la figura 15 se muestra el diagrama de despliegue que se propone para el sistema.

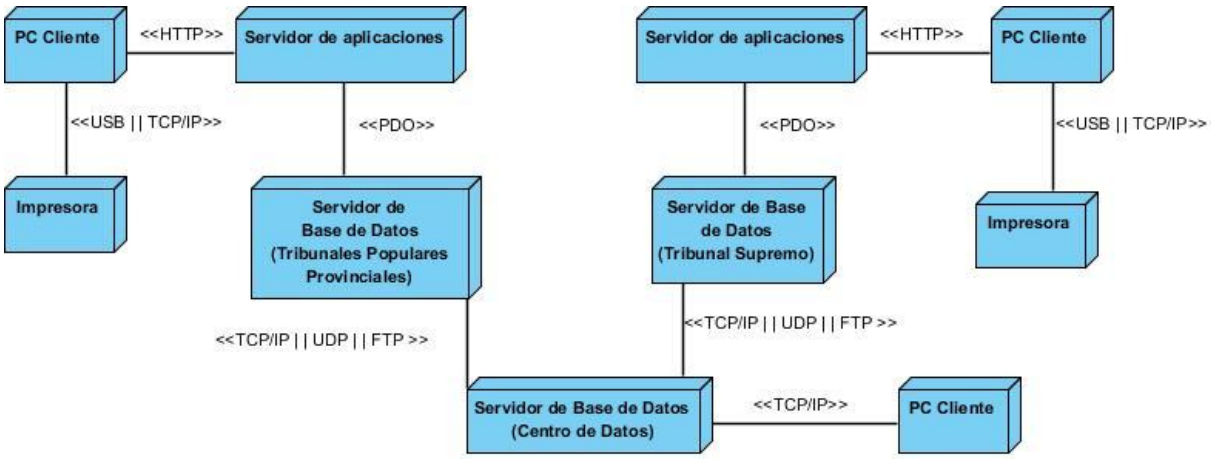


Figura 15. Diagrama de despliegue

✓ **Descripción de los nodos físicos**

Un nodo es "un elemento físico que existe en tiempo de ejecución y que representa un recurso computacional, que en general tiene al menos una memoria y a menudo capacidad de procesamiento".

Cada uno de los nodos que componen el diagrama de despliegue correspondiente a la aplicación tiene su relevancia para su correcto funcionamiento, definiéndolos de la siguiente manera:

- **Nodo PC – Cliente:** se encontrará el sistema operativo Linux y el navegador web Mozilla Firefox mediante el cual los clientes tendrán acceso al sistema y harán uso del mismo.
- **Nodo Centro de Datos:** se encontrará la base de datos que tendrá la información de todos los tribunales, ya sean, el Supremo y las instancias Provinciales.
- **Nodo Servidor de bases de datos (Tribunal Supremo):** se encontrará la base de datos con la información del Tribunal Supremo Popular, teniendo este comunicación con el Centro de Datos para una vez al día de 12 a 6 de la mañana, que es considerado un horario no laboral, se sincronizará con el Centro de Datos para enviar la información diaria.
- **Nodo Servidor de bases de datos (Tribunales Populares Provinciales):** se encontrará la base de datos con la información de las instancias de los

Tribunales Provinciales Populares, cada una sincronizará la información con el Centro de Datos en horario no laboral para enviar la información diaria.

- **Nodo Servidor de aplicaciones:** se encontrará todo lo concerniente a la aplicación, donde estarán agrupados los archivos a través de los cuales el usuario logra acceder al sistema, además se encuentra contenida toda la información específica de cada registro, sus clases; así como almacena además la configuración general del sistema, las clases y librerías externas y los plugins de instalación de la aplicación.

2.6.3 Interfaces del sistema

En el desarrollo de la aplicación se desarrollaron interfaces sencillas y seguras, que permiten una fácil interacción con el sistema, estando este acorde con el personal que lo utilizará. En la figura 16 se muestra la interfaz principal de entrada del juez, donde se aprecian los distintos vínculos a todas las funciones que requiere la misma.

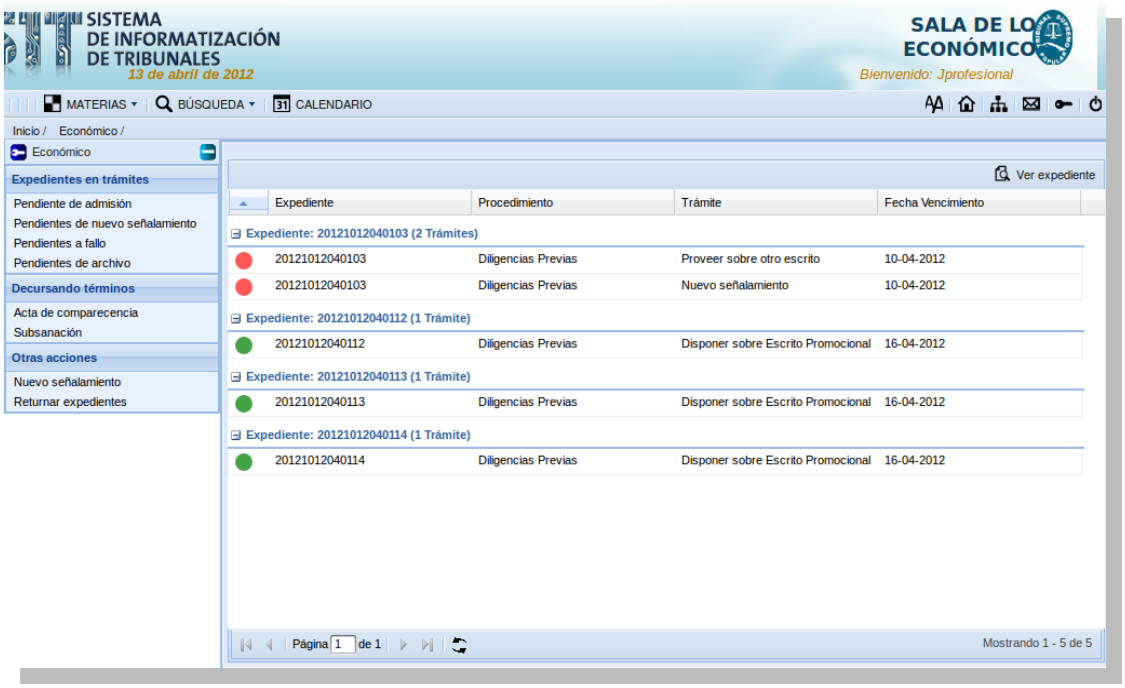


Figura 16. Pantalla inicial del juez

2.7 Conclusiones parciales

Los artefactos obtenidos en este capítulo son necesarios y de vital importancia para la construcción del sistema ya que posibilitan definirlo con suficiente detalle para permitir su interpretación y realización física.

El modelo de implementación permitió obtener una visión general de los componentes y subsistemas a implementar, facilitando que el proceso de implementación se realice de forma más simple y con una mayor calidad.

Capítulo 3: Validación de la solución

3.1 Introducción

Para la validación de la implementación del sistema, se realizaron un conjunto de pruebas de software que contribuyeron a determinar si se logró el objetivo propuesto por esta investigación, y conociendo que las pruebas de software consisten en la dinámica de la verificación del comportamiento de un programa en un conjunto finito de casos de prueba. Son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa u ordenador; probando el comportamiento del mismo.

En este capítulo se hace un análisis de los resultados obtenidos luego de realizarle a la implementación de la solución propuesta las pruebas de caja blanca y caja negra para comprobar su buen funcionamiento y su desempeño.

3.2 Prueba de caja blanca

Para asegurar adecuadamente el software conviene aplicar alguna prueba de tipo estructural (o de caja blanca), centrada en la implementación. Existen varios tipos de pruebas estructurales, como son aquellas que ejercitan todas las proposiciones del software bajo análisis, las que ejercitan todas las condiciones y las que ejercitan los caminos básicos. En este caso se realizará la prueba basada en un conjunto de caminos básicos. (31)

Para aplicar este método es necesario:

1. Convertir el código a un grafo de control.
2. Calcular el número ciclomático y obtener la base de caminos.
3. Preparar un caso de prueba para cada camino básico.

En la figura 17 se muestra el método BuscarPersonaNaturalAction() correspondiente a la clase controladora RegistrarEscritoPromocionalController.php el cual será utilizado como ejemplo para realizarle la prueba de camino básico.

```
function BuscarPersonaNaturalAction() {
    $formulario = json_decode(stripcslashes($this->_request->get('formulario'))); //1
    $primerNombre = $formulario->primerNombre; //1
    $segundoNombre = $formulario->segundoNombre; //1
    $primerApellido = $formulario->primerApellido; //1
    $segundoApellido = $formulario->segundoApellido; //1
    $carnet = $formulario->carnet; //1
    $start = $this->_request->getPost("start"); //1
    $limit = $this->_request->getPost("limit"); //1
    $idPromoventeGrid = $this->_request->getPost("idpromovente"); //1
    $idPdeudorGrid = $this->_request->getPost("idpdeudor"); //1
    $idPromoventeNew = $_SESSION['idPromovente']; //1
    $idPresuntoDeudorNew = $_SESSION['idPresuntoDeudor']; //1
    $idPersonaPromovente = $idPromoventeNew + $idPromoventeGrid; //1
    $idPersonaDeudora = $idPresuntoDeudorNew + $idPdeudorGrid; //1
    $arregloPersonasEscogidas = array(); //1
    $contador = 0; //1
    if ($idPersonaPromovente != 0)//2
        $arregloPersonasEscogidas[$contador++] = $idPersonaPromovente; //3
    if ($idPersonaDeudora != 0) //4
        $arregloPersonasEscogidas[$contador] = $idPersonaDeudora; //5
    $resultado = DpersonanaturalExt::BuscarPersonaNatural($limit, $start, $primerNombre,
         $segundoNombre, $primerApellido, $segundoApellido, $carnet, $arregloPersonasEscogidas);//6
    $natural = array(); //6
    for ($i = 0; $i < count($resultado->items); $i++) { //7
        $natural[$i]['ci'] = $resultado->items[$i]['pasaporte']; //8
        $natural[$i]['idpersona'] = $resultado->items[$i]['idpersona']; //8
        $natural[$i]['primernombre'] = $resultado->items[$i]['primernombre']; //8
        $natural[$i]['segundonombre'] = $resultado->items[$i]['segundonombre']; //8
        $natural[$i]['primerapellido'] = $resultado->items[$i]['primerapellido']; //8
        $natural[$i]['segundoapellido'] = $resultado->items[$i]['segundoapellido']; //8
        if ($natural[$i]['ci'] == null) //9
            $natural[$i]['ci'] = $resultado->items[$i]['ci']; //10
    }
    echo json_encode(array('total' => count($natural), 'datos' => $natural)); //11
    return; //11
}
```

Figura 17. Fragmento de código de ejemplo para prueba

El grafo de control a construir a partir del código mostrado debe cumplir un conjunto de reglas:

a) Existe un solo nodo inicial.
b) Existe un solo nodo final.
c) Cada nodo representa una secuencia de instrucciones, que puede ser vacía.
d) La relación entre los nodos es una relación de “el control pasa a”.
e) Un nodo puede tener uno o dos sucesores.
f) Un nodo puede tener uno o muchos antecesores.
g) Un nodo tendrá dos sucesores si existen dos caminos posibles, dependiendo de una condición.

h) Los casos de **for, while, until** y **case** se pueden reducir a grupo de nodos con dos sucesores.(31)

El grafo de control correspondiente al método mostrado en la figura 17 es el siguiente:

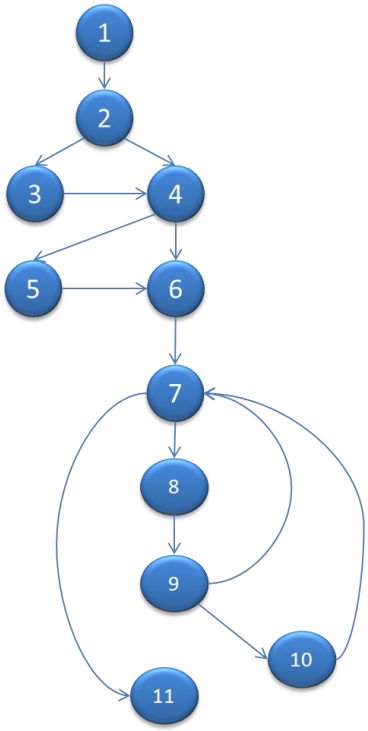


Figura 18. Grafo de control

Una manera de medir la complejidad del software es el uso de la métrica conocida como complejidad ciclomática. La misma se calcula a partir del grafo de control del software bajo análisis y mide el número de caminos independientes que pueden ocurrir en la ejecución, entre el inicio y el fin del software. **A** partir de un grafo de control **G=(N, V)**, donde **N** es un conjunto de nodos y **V** un conjunto de arcos entre los nodos, se calcula la complejidad ciclomática de varias maneras que resultan equivalentes.

El cálculo de la complejidad del grafo de la Figura 18 se realizó utilizando la forma clásica. La misma plantea que sean a (número de arcos) y n (número de nodos); entonces la complejidad ciclomática se define como **V(G)= a-n+2**. Por tanto la complejidad del grafo de control en cuestión es **V(G)= 14-11+2=5**.(31)

Sabiendo que la complejidad ciclomática es 5 entonces habrá 5 caminos básicos y que bastará probar estos para tener la seguridad de haber ejercitado todas las condiciones del código que se está probando.

Para obtener los caminos básicos se puede utilizar dos métodos conocidos como simplificado y general. El primero es bueno en algunos casos, pero el general resulta más flexible y centrado en las funciones importantes del módulo que se prueba.

En el método general se elige el primer camino como uno que tenga sentido funcional, es decir, que represente la operación normal del software. A partir de ese camino inicial se va variando una parte cada vez hasta completar los caminos necesarios. El procedimiento se puede describir como sigue:

1. Seleccionar un camino funcional que es el más importante a los ojos del probador y agregarlo a la base.
2. Fijar num-caminos en 1.
3. Mientras existan nodos de decisión con salidas no utilizadas y num-caminos < V(G):
   3.1. Seguir un camino básico hasta uno de tales nodos.
   3.2. Seguir la salida no utilizada y buscar regresar al camino básico tan pronto sea posible.
   3.3. Agregar el camino a la base e incrementar num-caminos en uno.(31)

Para el grafo de control en cuestión con complejidad ciclomática de 5, se tendrán los siguientes caminos:

Número	Camino
1	1-2-3-4-5-6-7-8-9-10-7-11
2	1-2-4-5-6-7-8-9-10-7-11
3	1-2-3-4-6-7-8-9-10-7-11
4	1-2-3-4-5-6-7-11
5	1-2-3-4-5-6-7-8-9-7-11

Tabla 19. Caminos básicos

Seguidamente se realizan los casos de pruebas que comprometerán la ejecución de cada camino básico identificado.

Camino	Condición de ejecución	Entrada	Resultado esperado	Resultado
1	Se ha seleccionado previamente una persona natural como promovente y otra persona natural como presunta deudora y se desea buscar una persona natural.	Alguno de los campos de filtro de búsqueda (nombre, segundo nombre, primer apellido, segundo apellido y ci/pasaporte) debe de tener valor.	Devuelva un arreglo de personas naturales donde no se incluyan las personas escogidas previamente	Positivo
2	Se ha seleccionado previamente una persona natural como presunta deudora y se desea buscar una persona natural.	Alguno de los campos de filtro de búsqueda (nombre, segundo nombre, primer apellido, segundo apellido y ci/pasaporte) debe de tener valor.	Devuelva un arreglo de personas naturales donde no se incluya la personas escogida previamente	Positivo
3	Se ha seleccionado previamente una persona natural como promovente y se desea buscar una persona natural.	Alguno de los campos de filtro de búsqueda (nombre, segundo nombre, primer apellido, segundo apellido y ci/pasaporte) debe de tener valor.	Devuelva un arreglo de personas naturales donde no se incluya la personas escogida previamente	Positivo
4	Se ha seleccionado previamente una	Alguno de los campos de filtro de	Devuelva un arreglo vacío.	Positivo

	persona natural como promovente y otra persona natural como presunta deudora y se desea buscar una persona natural.	búsqueda (nombre, segundo nombre, primer apellido, segundo apellido y ci/pasaporte) debe de tener valor.		
5	Se ha seleccionado previamente una persona natural como promovente y otra persona natural como presunta deudora y se desea buscar una persona natural.	El campo de filtro de búsqueda ci/pasaporte tiene que ser el de un pasaporte.	Devuelva un arreglo de personas naturales donde no se incluyan las personas escogidas previamente y todas las personas encontradas sean extranjeras.	Positivo

Tabla 20. Casos de prueba

3.3 Prueba de caja negra

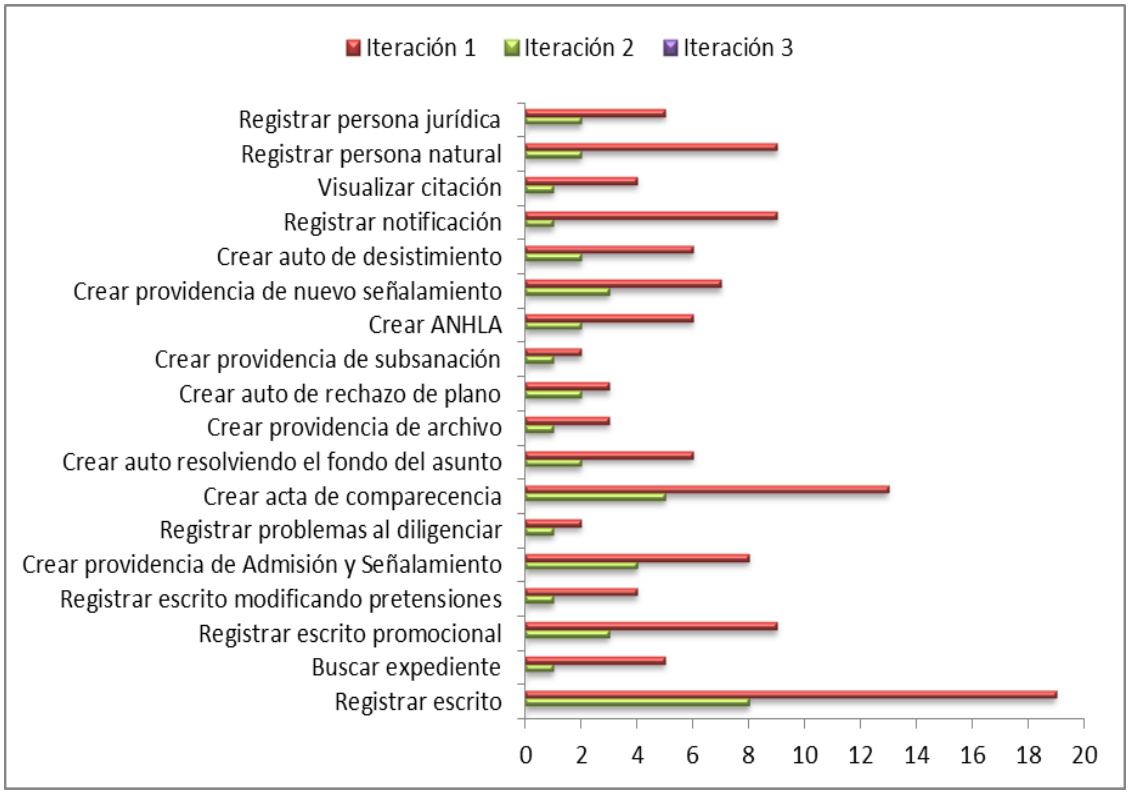
Las pruebas de caja negra son aquellas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de 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.(32)

La prueba de caja negra se centra principalmente en los requisitos funcionales del software y permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos. Estas permiten encontrar:

- Funciones incorrectas o ausentes.

- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.(32)

En la siguiente gráfica se muestran los resultados de las pruebas funcionales o de caja negra realizadas por el grupo de calidad del centro CEGEL al procedimiento Diligencias Previas del subsistema Económico para validar su implementación, llevándose a cabo 3 iteraciones donde se detectaron un conjunto de No Conformidades que fueron resolviéndose paulatinamente. En la misma se puede apreciar que en cada iteración disminuye la cantidad de inconformidades y que el caso de uso Registrar Escrito fue el que más problemas presentó por ser el más complejo, pero al igual que el resto en la última iteración fueron subsanadas todas las dificultades.



Gráfica 21. Iteraciones de pruebas realizadas a la aplicación vs No Conformidades detectadas

3.4 Conclusiones parciales

Con la realización de las pruebas a la aplicación desarrollada, se evidenció lo importante que son las mismas para darle un acabado al producto y poder darle el uso esperado. Con las pruebas de caja blanca se puede apreciar que el código implementado no tiene muchos errores y que la programación ha sido efectiva en cuanto a ello, los resultados esperados en cada uno de los casos de prueba fueron los obtenidos. Las pruebas de caja negra realizadas demuestran que las funcionalidades del sistema tienen correspondencia con las que se definieron inicialmente.

Conclusiones generales

Con la realización de este trabajo se arriban a las siguientes conclusiones:

- La fundamentación teórica utilizada en la ponencia posibilitó justificar la selección de la metodología, las herramientas, tecnologías y patrones utilizados.
- El modelo de implementación permitió generar los elementos necesarios para comprender la implementación del sistema y la estrategia para el despliegue del SIT.
- La validación del sistema mediante la aplicación de las pruebas de caja blanca y caja negra arrojaron resultados satisfactorios, demostrando el cumplimiento del objetivo planteado por esta investigación.

Recomendaciones

A pesar de concluir satisfactoriamente con la investigación y haber cumplido su objetivo principal, se recomienda:

- Realizar la prueba piloto de la aplicación en la sala de lo económico del Tribunal Provincial de La Habana.
- Realizar futuras mejoras al sistema según el avance de las tecnologías de desarrollo.

Bibliografía

1. GUILLÉN SUÁREZ, Angel Alexander. Ingeniería de Requisitos del procedimiento Diligencias Previas al proceso ejecutivo del Módulo Económico del Proyecto Tribunales Populares Cubanos (TPC). Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas. La Habana: Universidad de las Ciencias Informáticas, 2011.

2. BELLO. Diligencias Previas al Proceso Ejecutivo. 2006. S.l.: s.n.

3. MINISTERIO DE JUSTICIA DE ESPAÑA. Oficina Judicial - Ministerio de Justicia. In: [online]. 2010. [Accessed 18 febrero 2012]. Available from: http://oficinajudicial.justicia.es/portaloj/sistema_minerva.

4. Lexnet para Procuradores en los partidos judiacales de Huelva. In: [online]. [Accessed 18 febrero 2012]. Available from: http://www.phase.es/noticias/noticias-sobre-phase-informatica/128-lexnet-procuradores-de-los-tribunales-de-huelva-y-phase.html.

5. AVANTIUS WEB v1.4. Sistema de Gestión Procesal de Justicia - Proyectos - Servicios - Sobre Tracasa - Tracasa - Trabajos Catastrales, S.A. In: [online]. [Accessed 18 febrero 2012]. Available from: http://www.tracasa.es/servicios/servicios+proyectos_ficha-ISSI.aspx.

6. SUPERIOR TRIBUNAL DE JUSTICIA. Diariojudicial.com. In: [online]. noviembre 2008. [Accessed 18 mayo 2012]. Available from: http://www.diariojudicial.com/contenidos/2009/03/18/noticia_0004.html.

7. GONZÁLEZ MORELL, Msc. Daniel E. y GONZÁLEZ GUADARRAMAS, Msc. José R. Sistema para la Tramitación de Procesos Penales. S.l. Universidad Central Marta Abreu de Las Villas, 2008.

8. COLECTIVO DE PROFESORES P4. Programación en el cliente [online]. 2008. S.l.: s.n. Available from: http://eva.uci.cu/file.php/106/redisenno_P4/Semana_1/Guia_del_estudiante.pdf.

9. Metodologias de desarrollo de Software - EcuRed. In: EcuRed [online]. [Accessed 15 mayo 2012]. Available from: http://www.ecured.cu/index.php/Metodologias_de_desarrollo_de_Software.

10. Framework - EcuRed. In: [online]. [Accessed 18 mayo 2012]. Available from: http://www.ecured.cu/index.php/Framework.

11. LEOPOLDO, Carlos. Zend Framework, una introducción [online]. S.l. 2007. Available from: http://techtastico.com/post/zend-framework-una-introduccion.

12. GÓMEZ BARYOLO, Oiner, TENRERO CABRERA, Marianela y SILEGA MARTÍNEZ, Nemuris. Plantilla Registro de la Propiedad intelectual (Sauxe). La Habana. 2008.

13. DOCTRINE-PROJECT. Doctrine: ORM Open Source para PHP 5.2. 2008. S.l.: s.n.

14. EXTJS. Desarrollo en Web. In: [online]. 2008. Available from: http://desarrolloweb/2008/10/extjs.

15. NAVARRO DIAZ, Maikel. Arquitectura de software del proyecto TPC. La Habana. Universidad de las Ciencias Informáticas, 2010.

16. LARMAN, Carig. UML y Patrones. Introducción al análisis y diseño orientado a objetos. S.l.: Prentice Hall, [no date].

17. MICROSOFT. Conceptos básicos: funcionamiento de la programación. In: [online]. [Accessed 18 mayo 2012]. Available from: http://msdn.microsoft.com/es-es/library/ms172579(v=VS.80).aspx.

18. ORACLE. ¿Qué es JavaScript y en qué se diferencia de la tecnología Java? In: [online]. [Accessed 18 mayo 2012]. Available from: http://www.java.com/es/download/faq/java_javascript.xml.

19. MEYER, Eric A. Cascading Style Sheets 2.0 Programmer's Reference [online]. S.l.: Osborne/McGraw-Hill, [no date]. [Accessed 18 mayo 2012]. ISBN 0-07-213178-0. Available from: http://meyerweb.com/eric/books/css-progref/.

20. Lenguaje de Marcado de Hipertexto - EcuRed. In: [online]. [Accessed 18 mayo 2012]. Available from: http://www.ecured.cu/index.php/Html.

21. Lenguajes de programación del lado del servidor. In: [online]. [Accessed 18 mayo 2012]. Available from: http://tuto-mpweb.webcindario.com/2_3.html.

22. GUTMANS, Andi, SAETHER BAKKEN, Stig y RETHANS, Derick. PHP5 Power Programming [online]. S.l.: Prentice Hall, 2005. [Accessed 18 mayo 2012]. Available from: http://www.etnassoft.com/biblioteca/php5-power-programming/.

23. CASE Tools. In: [online]. [Accessed 18 mayo 2012]. Available from: http://case-tools.org/.

24. VISUAL PARADIGM INTERNATIONAL LTD. VP - UML Quick Start [online]. S.l.: s.n., 2012. [Accessed 18 mayo 2012]. Available from: http://www.visual-paradigm.com/product/vpuml/.

25. The Apache HTTP Server Project. In: [online]. [Accessed 18 mayo 2012]. Available from: http://httpd.apache.org/.

26. Sistema Gestor de Base de Datos - EcuRed. In: [online]. [Accessed 18 mayo 2012]. Available from: http://www.ecured.cu/index.php/Sistema_Gestor_de_Base_de_Datos.

27. PostgreSQL. In: [online]. [Accessed 18 mayo 2012]. Available from: http://www.postgresql.org.es/sobre_postgresql.

28. COLLINS-SUSSMAN, Ben, FITZPATRICK, Brian W. y PILATO, C. Michael. Control de versiones con Subversion [online]. S.l. [no date]. [Accessed 18 mayo 2012]. Available from: http://svnbook.spears.at/nightly/es/svn-book.html#svn-ch-1-sect-1.

29. Navegador Firefox. In: [online]. [Accessed 19 mayo 2012]. Available from: http://www.mozilla.org/es-ES/firefox/features/.

30. PRESSMAN, Roger S. Ingeniería de Software. Un enfoque práctico. España: Mc Graw Hill, 2005.

31. FERNÁNDEZ PEÑA, Juan Manuel. Pruebas de software [online]. S.l.: s.n., 2006. Available from: www.uv.mx/personal/jfernandez/files/2010/07/Cap3-Caminos.pdf.

32. Material_de_caja_b_y_caja_n [online]. S.l.: s.n. Available from: http://eva.uci.cu/file.php/158/Documentos/Recursos_bibliograficos/Libros_y_articulos_UD_2/Comun/Material_de_caja_b_y_caja_n.pdf.