Universidad de las Ciencias Informáticas

“Facultad 3”

Título: Desarrollo del subsistema “Administración y Gobierno” para el proyecto Informatización de los Tribunales Populares Cubanos.

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

Autores: *Daylenis Sánchez Delgado*
Juan Carlos Grenot Jardines

Tutor: *Ing. Handy Hernández Dalmau*
Asesor Técnico: *Ing. Arianna Leyva Campos*

La Habana Curso 2011-2012

"Dediquémonos, con modestia y sin fanfarria, cada cual en el puesto que le corresponde, al cumplimiento diario y estricto del deber. ¡Ésa es la clave!"

Raúl Castro Ruz

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____ _____

Daylenis Sánchez Delgado Juan Carlos Grenot Jardines

_____ _____

Ing. Handy Hernández Dalmau Ing. Arianna Leyva Campos

DATOS DE CONTACTO

Ing. Handy Hernández Dalmau: (hdalmau@uci.cu) Graduado de Ingeniero en Ciencias Informáticas en el año 2008 en la Universidad de las Ciencias Informáticas (UCI). Profesor interno. Jefe de capacitación y tutores, coordinador del Proyecto de Informatización de los Tribunales Populares Cubanos.

Ing. Arianna Leyva Campos: (alcampos@uci.cu) Graduada de Ingeniera en Ciencias Informáticas en el año 2010 en la Universidad de las Ciencias Informáticas (UCI). Profesor interno. Ha desempeñado los siguientes roles: analista, programadora, planificadora y jefa de subsistema del Proyecto de Informatización de los Tribunales Populares Cubanos. Ha cursado todos los cursos obligatorios del diplomado de Docencia e Investigación Universitaria y de Gobierno Electrónico.

Daylenis:

A mi mamá y mi papá, por haber confiado tanto en mí, por todo su sacrificio y por ayudarme a realizar mis sueños. A mi abuela, por ser mi cómplice, mi consejera y por su amor incondicional.

A mi novio Luis Ernesto, por ser lo más lindo que me ha pasado en la vida, por compartir cada segundo conmigo, por ayudarme en todo, por estar ahí siempre que lo necesito, por ser mi persona.

A mis hermanos Orlando, José Alberto y Danay, por su infinito cariño.

A Tata y su esposo Samuel, que es más que mi prima, es mi hermana, por aguantarme todo y ayudarme siempre. A toda mi familia, a mi tía María Caridad, a mis tías Celia, Yaqueline, Dávida e Iracé, por su fe en mí. A mis tíos Orlando y Arbelio. A todos mis primos, en especial a Arianna, Kirenia, Katia, Vivian, Arbelito, Joenny, Heidi y Alejandro. A los más pequeños de la familia, Javier, Claudia, Liset, Maykro y Marlito.

A la familia de mi novio, que es también la mía, por acogerme como una más de los suyos, a mis suegros Luis y Maritza, a Magalys, Rocío, Mercedes.

A mis vecinos, Graciela, Milagro, Luis, Oscar, Aidita, Nidia, por ayudarme a convertirme en la joven que soy hoy.

A mis amigos de siempre, Aina, Diane, Nany, Baby, Katia, Lupita, Yaimé, Ariannita, Ángel, Juan Manuel, Israel, gracias por regalarme tantos momentos felices. A mis amigos de la Félix, Julio, Marcel, Miriam, Elaine, Glenda, Enrique, Ailén, Ariel, Yohana, por todos los momentos especiales compartidos, gracias por hacerme sentir en familia.

A los amigos que he conocido durante estos 5 años, a mis compañeros de clase, en especial a Eilianys, David, Dayana, Liván, Yasmany, Reinier, Pedro Frank, José, Rafael y Gustavo. A Ray Williams, José Ángel, Luis Felipe, Manuel Álvarez, por su ayuda y amistad. A mis profesores y maestros en especial a Consuelo, Evelin, Yadamí y Leidili. A mis tutores , en

especial a Arianna, por tantas noches de desvelo, por sus consejos, su amistad y su cariño.

A mis compañeros y profesores del proyecto TPC, en especial a Tony, Manuel, Raidel, Isabel, Alex, Maykel, Daimi, Darian, Yoslenis, Martin, Dayron y Yosvani.

A todos de corazón gracias…

Juan Carlos:

A mi familia y en especial a mis padres ya que sin la ayuda y amor incondicional de ellos no hubiera sido posible la realización de este sueño que es el de poder ser un ingeniero informático, todo esto se lo debo a ellos.

A mi familia del 9109 que vienen desde 3ro conmigo, que doy gracias a la vida por haberlos conocido, ustedes son un ejemplo de que es tener una buena amistad.

A mis amigos del aula, de la facultad, de otras facultades y a otras personas que ya no se encuentran en la universidad debido a una u otra razón.

A los profesores que me impartieron clases durante estos 5 largos años.

A los profesores y alumnos del proyecto que me han ayudado en todo este tiempo a mi tutor Handy, a la gente de mi módulo Ariannita, Daylenis y Raidel.

A las tías del edificio por haberme ayudado tanto en todos los sentidos.

En fin a todas las personas que laboran en esta escuela y me han ayudado les agradezco de corazón a todos.

Daylenis:

A mi familia, en especial a mis padres y mi abuelita Olga, a la luz de mi vida: Luis Ernesto, a mis amigos de siempre, y al realizador de todos nuestros sueños: Fidel Castro Ruz.

Juan Carlos

Dedico el presente trabajo a mis padres ya que son las personas más importantes para mí, por brindarme su amor, comprensión y cariño infinitos.

Resumen

En la presente investigación se describe el desarrollo del subsistema Administración y Gobierno del Sistema de Informatización de Tribunales (SIT), el cual surge ante la necesidad de automatizar los procesos de vencimiento de términos y configuración del calendario de señalamientos de actos judiciales de las materias: Económico, Civil, Penal, Administrativo y Laboral, así como la implementación de las funcionalidades relacionadas con la configuración de usuarios, roles, dominios, tribunales, nomencladores y servicios imprescindibles para el correcto funcionamiento del SIT. La implementación del subsistema incorpora un conjunto de beneficios a los Tribunales Populares Cubanos como son: la gestión de los términos de los trámites que se realizan en el tribunal alertando a los usuarios[1] cuando un trámite está a punto de declararse extemporáneo[2], mecanismos de búsqueda de información, notificación automática de un conjunto de eventos de interés para los usuarios, así como el ahorro de recursos materiales y humanos.

Para la realización del sistema se utilizó RUP[3] como metodología de desarrollo, UML como lenguaje de modelado, Visual Paradigm como herramienta CASE[4] para el modelado, PHP como lenguaje de programación, PostgreSQL como gestor de base de datos, Apache como servidor web y se utilizaron los marcos de trabajo ExtJS, Sauxe y Doctrine. Como resultado se obtuvo un subsistema funcional que está en fase de pruebas de liberación en el centro CALISOFT en la Universidad de las Ciencias Informáticas. El sistema fue presentado en la Jornada Científica Estudiantil en su XII edición alcanzando el resultado de Destacado.

Palabras clave: *subsistema Administración y Gobierno, SIT, vencimiento de términos, configuración.*

[1] Jueces, abogados y fiscales
[2] Impropio del tiempo en que sucede o se hace, fuera de lugar
[3] Rational Unified Process (Proceso unificado de Rational)
[4] Computer Aided Software Engineering (Ingeniería de software asistida por computadora)

Índice

Índice de Tablas

Índice de Figuras

Introducción

La ciencia y la tecnología son dos fenómenos inherentes al hombre desde su aparición. El primero, surge con la necesidad o capacidad del hombre de encontrar respuestas a todo lo que lo rodea. Los conocimientos de la ciencia le permitieron modificar su entorno, fabricar objetos y bienes materiales mediante el uso de técnicas, herramientas y teorías. De esta forma surge el segundo fenómeno, la tecnología: aprovechamiento práctico del conocimiento científico.

Como resultado de una de las disímiles aplicaciones de la ciencia, surge la Informática con el objetivo de hacer posible el tratamiento automatizado de la información. La automatización oportuna de procesos mediante el uso de la Informática favorece la disminución de los costos y el incremento de la productividad. Con el desarrollo de la informatización en todas las esferas de la sociedad y el perfeccionamiento acelerado de las Tecnologías de la Información y las Comunicaciones (TIC´s), la mayoría de las empresas e instituciones necesitan informatizar los procesos de trabajo que desarrollan para ganar competitividad, eficiencia y tiempo, por lo que entre los objetivos estratégicos de los Tribunales Populares Cubanos, en lo adelante TPC, se encuentra la automatización de sus procesos.

Los TPC constituyen un sistema de órganos estatales cuya función es impartir justicia dimana[5] del pueblo en sus tres instancias: suprema, provincial y municipal, es ejercida en nombre del pueblo por el Tribunal Supremo Popular quien posee la máxima autoridad judicial y sus decisiones son definitivas. (QUESADA, 1997).

En los TPC actualmente todos los procesos que se llevan a cabo se hacen manualmente y con gran dependencia del papel, por ejemplo el registro de documentos, el asiento en los libros, la creación y archivo de los expedientes; posibilitando que se cometan errores de escritura, tachaduras, borrones y pérdida de documentos. Por otra parte los reportes estadísticos se emiten en plazos muy largos para la alta dirección, por lo que no es posible tomar decisiones operativas basadas en información actualizada.

El control de los vencimientos de términos es clave para garantizar el principio “El derecho a un proceso sin dilaciones[6] indebidas”**.** En la Ley de los Procedimientos Civil, Administrativo, Laboral y Económico y la Ley Nº 5 del Procedimiento Penal se establecen términos procesales [7] para los actos, éstos tienen como fin que los litigios sean resueltos con la mayor rapidez posible. Actualmente en los tribunales cubanos es el secretario el encargado de estar al tanto del vencimiento de los términos procesales y de

[5] Se origina de, proceder una cosa de otra o derivarse de ella
[6] Retraso o demora de algo por un tiempo
[7] Circunstancias que constituyen dilación

su notificación al juez. Una vez notificado el vencimiento del término al juez por medio de diligencias éste procede según las disposiciones expresadas por la ley. Las partes también son responsables de conocer el tiempo que les fue impuesto por el tribunal para realizar sus trámites. Para realizar esta actividad los secretarios utilizan los siguientes datos: el término establecido por la ley para el acto procesal en cuestión y el calendario de días no hábiles. Esta tarea es tediosa y requiere un alto grado de desgaste humano por la cantidad de procesos que se atienden a diario. La concepción actual de esta actividad tiene numerosas limitaciones: primero, se enfoca en notificar solo cuando el término ya se ha vencido sin tener en cuenta los términos que están próximos a vencerse y de esta forma alertar a los implicados, segundo el cálculo manual para determinar cuándo se vence el término implica un gran cúmulo de información que debe ser analizada por el secretario lo que lo hace sensible a errores; y tercero se centra en el control del cumplimiento de los términos por las partes.

El cronograma de señalamientos de los actos públicos es otra de las tareas que resultan de gran dificultad actualmente en los tribunales, debido a que implica el análisis de un gran cúmulo de información de todas las materias. La complejidad de este proceso lo hace susceptible a errores humanos y provoca dilación en los procesos señalados.

La Universidad de las Ciencias Informáticas (UCI) es la encargada de informatizar los procesos de los TPC, por lo que el Centro de Gobierno Electrónico (CEGEL) de la Facultad 3 crea el proyecto Informatización de los Tribunales Populares Cubanos, donde se desarrolla la solución informática Sistema de Informatización de Tribunales (SIT), la cual fue dividida lógicamente en 7 subsistemas, 5 de ellos encargados de informatizar los procesos por materias (Administrativo, Civil, Penal, Laboral, Económico), el subsistema Común: encargado de centralizar el conjunto de actos procesales definidos como comunes para las materias, el restante: Administración y Gobierno, surge ante la necesidad de la existencia de un subsistema encargado de tratar la seguridad, configurar los usuarios con roles que le permitan regular las acciones según el cargo que posean los mismos, gestionar la creación de los tribunales, salas y asignarles las materias que se atienden en los mismos, además de agrupar un conjunto de información relevante para los procesos de los restantes módulos del SIT.

Analizando la situación de los TPC planteada con anterioridad, teniendo en cuenta que actualmente se han descrito los procesos de negocio que se comenzarán a informatizar y que fueron determinados los requerimientos del subsistema Administración y Gobierno, se identificó el siguiente **problema a resolver**:

¿Cuál es la especificación de los artefactos necesarios para la informatización de los requisitos identificados en el subsistema Administración y Gobierno del proyecto Informatización de los Tribunales Populares Cubanos?

Para dar respuesta al problema planteado se define como **objetivo general**: Desarrollar el subsistema Administración y Gobierno perteneciente al proyecto Informatización de los Tribunales Populares Cubanos, del cual se derivan como **objetivos específicos** los siguientes:

- Elaborar el marco teórico de la investigación.
- Obtener el Modelo de diseño.
- Obtener el Modelo de implementación.
- Validar los resultados obtenidos.

Se define como **objeto de estudio**: El proceso de desarrollo de software de sistemas informáticos para la gestión jurídica del cual se deriva como **campo de acción** el diseño e implementación del subsistema Administración y Gobierno perteneciente al proyecto Informatización de los Tribunales Populares Cubanos.

La investigación asume como **idea a defender**: Con el desarrollo del subsistema Administración y Gobierno se obtendrá la especificación de los artefactos necesarios para la informatización de los requisitos identificados.

Para dar cumplimiento a los objetivos se definen las siguientes **tareas investigativas**:

- ✓ Estudio del estado del arte de soluciones informáticas para la gestión judicial.
- ✓ Estudio del proceso de desarrollo de software y su metodología.
- ✓ Estudio de la plataforma de desarrollo y de las herramientas utilizadas para diseñar, implementar y probar el sistema (definidas en el Documento de Arquitectura del Proyecto ITPC[8]).
- ✓ Estudio y selección de los patrones de diseño más factibles para la solución propuesta.
- ✓ Realización de los Diagramas de clases del diseño.
- ✓ Realización de los Diagramas de secuencia.
- ✓ Realización de los Diagramas de componentes.
- ✓ Realización del Diagrama de despliegue.
- ✓ Realización del Modelo de datos.
- ✓ Implementación de los componentes.

[8]Informatización de los Tribunales Populares Cubanos

- ✓ Diseño de casos de prueba.
- ✓ Realización de pruebas al sistema.
- ✓ Análisis de los resultados de las pruebas realizadas al sistema.

Para llevar a cabo las tareas investigativas se emplearon diversos métodos científicos teóricos y empíricos. Los **métodos teóricos** utilizados fueron:

- ✓ **Histórico Lógico:** Se aplica al realizar el estudio a nivel nacional e internacional del desarrollo de sistemas similares al que se quiere implementar, así como de la evolución de los procesos de gestión de los Tribunales Populares Cubanos.
- ✓ **Analítico-Sintético**: Este método se emplea para organizar y sintetizar toda la información obtenida del estudio del estado del arte de los diferentes sistemas informáticos para la gestión judicial. La utilización de este método permite comprender mejor toda la información anterior, la cual será de gran utilidad para lograr un adecuado desarrollo del sistema.
- ✓ **Modelación**: Es usado para realizar los diagramas necesarios en el desarrollo del sistema que mejorarán el entendimiento de los procesos de gestión de los Tribunales Populares Cubanos.

Y como **métodos empíricos** se utilizan los siguientes:

- ✓ **Medición:** Es el procedimiento que se realiza con el objetivo de obtener información numérica acerca de una propiedad o cualidad del objeto, donde se comparan magnitudes medibles y conocidas. En la presente investigación es utilizado para medir la calidad del producto resultante mediante la realización de pruebas al sistema.

El contenido del presente trabajo está distribuido en 3 capítulos:

En el **Capítulo 1**, Fundamentación teórica, se realiza un estudio del estado del arte relacionado con los sistemas informáticos para la gestión judicial, además se exponen los conceptos fundamentales relacionados con el tema de la investigación y se describen los lenguajes, las herramientas y metodología a utilizar para el desarrollo de la aplicación.

En el **Capítulo 2**, Propuesta de solución, se presenta todo lo relacionado al diseño e implementación del sistema, se modelan los Diagramas de clases del diseño e implementación, Diagramas de interacción, Diagrama de despliegue, Diagrama de componentes, Modelo de datos, se presentan además los Estándares de codificación utilizados en la implementación.

Capítulo 3, Validación de los resultados, en este capítulo se registran los resultados de las pruebas realizadas al sistema.

Además, el trabajo incluye Resumen, Conclusiones, Recomendaciones, Bibliografía referenciada, Bibliografía consultada y Glosario de términos.

Se espera como **posible resultado** de la investigación:

- ✓ Modelo de diseño.
- ✓ Modelo de datos.
- ✓ Modelo de implementación.
- ✓ Diseño de casos de prueba.

Capítulo 1: Fundamentación teórica

Introducción

En el presente capítulo se realiza un estado del arte de los sistemas informáticos para la gestión judicial similares al que se desarrollará y además se caracterizan las tecnologías y metodologías que se utilizan durante el desarrollo del sistema.

1. Definiciones relacionadas con el objeto de estudio y el campo de acción

¿Qué es Administración?

“La administración es "el conjunto de las funciones o procesos básicos (planificar, organizar, dirigir, coordinar y controlar) que, realizados convenientemente, repercuten de forma positiva en la eficacia y eficiencia de la actividad realizada en la organización.” (Díez de Castro Emilio Pablo, 2004)

“La administración es "el proceso de diseñar y mantener un entorno en el que, trabajando en grupos, los individuos cumplan eficientemente objetivos específicos”. (Heinz, 2004)

¿Qué tipo de funcionalidades debe realizar un módulo de administración?

Por lo general diferentes tipos de aplicaciones necesitan la gestión o configuración de los usuarios, las estructuras donde operan sus funcionarios, gestión de grupos de usuarios, definición de dominios, regulación de acciones, control de acceso restringido para cada estructura, entre otras funcionalidades.

¿Qué es Informática Jurídica?

La Informática Jurídica es una ciencia que forma parte de la Informática, es la especie en el género, y se aplica sobre el Derecho, para dar tratamiento lógico y automático a la información legal. Es una ciencia que estudia la utilización de aparatos o elementos físicos electrónicos, como la computadora, en el Derecho; es decir, la ayuda que este uso presta al desarrollo y aplicación del Derecho. (Ecu-Red, 2011)

La Informática Jurídica estudia el tratamiento automatizado de:

- ✓ Las fuentes del conocimiento jurídico a través de los sistemas de documentación legislativa, jurisprudencial y doctrinal (Informática Jurídica Documental).
- ✓ Las fuentes de producción jurídica, a través de la elaboración informática de los factores lógico-formales que concurren en el proceso legislativo y en la decisión judicial (Informática Jurídica Decisional).
- ✓ Los procesos de organización de la infraestructura o medios instrumentales con los que se gestiona el Derecho (Informática Jurídica de Gestión). (Ecu-Red, 2011)

¿Qué es Informática Jurídica de Gestión?

Rama de la Informática Jurídica encaminada a organizar y controlar la información jurídica de documentos, expedientes, libros, etc., ya sea mediante la aplicación de programas de administración que permitan crear identificadores y descriptores para la clasificación de dicha información. Este tipo de informática es conocida como de administración y/o control, es utilizada en tribunales, estudios jurídicos, notarías, entre otras; se utiliza sobre todo para llevar el seguimiento de trámites y procesos con el objeto de mantener actualizada la información y llevar un buen control de la misma. (Ecu-Red, 2011)

2. Subsistema Administración y Gobierno

Los sistemas informáticos actuales son cada vez más flexibles y adaptables en dependencia de las necesidades de los clientes, por lo general están compuestos por varios módulos independientes que realizan determinadas funciones y todos ellos unidos por una estructura organizativa llamada módulo de administración para que el sistema funcione de forma ordenada, segura y consistente. Por esta razón generalmente la mayoría de los sistemas cuentan con un rector informático, el cual se encarga de permitir el control de la estructura organizativa del sistema, garantizando eficiencia en todos sus módulos y procesos, posibilitando la centralización de algunos procedimientos y configuraciones globales que garantizan el buen funcionamiento del mismo. Manejando diferentes conceptos en dependencia de las necesidades finales del cliente, por ejemplo: usuarios, puestos de trabajo, gestión de configuración de archivos y variables, control del flujo de eventos, entre muchos otros. (Pérez, 2008)

En el subsistema Administración y Gobierno se agrupan aquellas funcionalidades relacionadas con la configuración del sistema, la gestión de usuarios y gestión de fechas de trabajo a los jueces legos, calendario de señalamientos con los servicios de vencimiento de términos, calendario de días no hábiles, configuración de roles de acceso y restricción de acciones a usuarios, conjunto de dominios y gestión de los tribunales con sus salas y materias que atienden. Además incluye los nomencladores donde se definen un conjunto de datos necesarios para llevar a cabo los procesos de los TPC, como por ejemplo el tipo de sujeto, tipo de moneda, tipo de delito, tipo de organismo, motivo de subsanación de demanda, entre otros. El subsistema se encarga también de brindar servicios a los restantes módulos del SIT. Soporta la creación (gestión) de tribunales en las diferentes instancias. Los tribunales a su vez contendrán usuarios, cada uno con roles definidos que podrán ser modificados en los respectivos tribunales para los que fueron creados.

Permite además la gestión de las reglas del calendario de días no hábiles por las que se regirán todos los tribunales del país, así como el calendario de señalamientos donde se va a permitir el señalamiento de los

actos judiciales públicos y serán visibles a todos los módulos junto a la fecha de vencimiento de los mismos.

El subsistema debe garantizar un proceso de autenticación que es el encargado de verificar que alguien o algo es quien o lo que dice ser y se realizará de la siguiente manera: en redes de equipos públicos y privados la autenticación se lleva a cabo comúnmente a través de contraseñas de inicio de sesión. Este proceso se lleva a cabo de la siguiente manera. Todos los clientes son autenticados cuando se conectan a un servidor, deben escoger primero la estructura (tribunal o sala) por la que trabajarán en el sistema, después de eso el cliente es de confianza y a su vez cuando un cliente ejecuta alguna acción, ésta es chequeada, entra una política específica, donde cada uno de los controles y acciones son permitidos o denegados para ese usuario en dependencia de sus permisos. Esta especificidad asume que el usuario está autenticado tanto tiempo como dure la conexión.

Otro de los procesos llevados a cabo por el subsistema es la asignación y creación de roles, una vez creado un rol se le especifica a qué subsistema y funcionalidades puede acceder, y luego ese rol es asignado al usuario especificando además a cuál estructura (tribunales y salas) tiene permiso con ese rol. Permite además la activación y desactivación de usuarios, que será crítica para la seguridad del sistema ya que se mantendrá el registro de todos los usuarios que han trabajado en el mismo. Por ejemplo si un usuario cambiara de centro de trabajo el mismo sería desactivado en lugar de eliminar sus datos y las operaciones que realizó.

El desarrollo del subsistema Administración y Gobierno dentro de la solución informática SIT para los TPC les proporcionarían un conjunto de beneficios como los que a continuación se listan:

- ✓ La gestión de los términos de los trámites que se realizan en el tribunal alertando a los usuarios (jueces, abogados y fiscales) cuando un trámite está a punto de declararse extemporáneo.
- ✓ Existencia de un calendario de señalamientos de actos públicos donde los usuarios podrán comprobar la planificación de los mismos.
- ✓ Existencia de un calendario de días no hábiles, el cual hará más sencillo el proceso del cálculo del vencimiento de términos.
- ✓ La notificación automática de un conjunto de eventos de interés para los usuarios.
- ✓ Mecanismos de búsquedas de información.
- ✓ Configuración y asignación de fechas laborales en un tribunal para los Jueces Legos.
- ✓ Gestión, manejo y fácil acceso a información especializada con la consulta a los nomencladores.
- ✓ Ahorro de recursos materiales y humanos.

Teniendo en cuenta lo anteriormente planteado la implementación de un sistema informático que optimice los procesos que se realizan en los TPC supondrá una mejora significativa en los servicios que estos brindan repercutiendo positivamente en la sociedad cubana.

3. Estado del arte de sistemas informáticos relacionados con el campo de acción

El proyecto del cual es parte esta investigación centra su desarrollo en la rama de la Informática Jurídica de Gestión por las aspiraciones y características del mismo, pues su principal objetivo es automatizar los procedimientos judiciales en cada instancia de los tribunales cubanos.

La introducción de las TIC´s en la administración de justicia puede permitir una justicia de mayor calidad y, al mismo tiempo, abierta, transparente y próxima al ciudadano. En la actualidad se promueve el uso de la informática en la rama del derecho, y aunque su avance es lento, existen numerosas soluciones informáticas en el mundo que hacen uso de sistemas informáticos para la gestión procesal, la mayoría de estas aplicaciones están enfocadas en la agilización de la tramitación procesal, sin embargo hasta el momento no existe una solución informática que facilite íntegramente el desarrollo de un proceso judicial similar a los que tienen lugar en los TPC debido a las diferencias en las leyes de cada país.

3.1. Sistemas estudiados en el mundo

Infolex, Software de gestión jurídica (España): Este software es realizado por la empresa Jurisoft la cual es líder en el sector de la Informática Jurídica. La misma se dedica a la creación de software aplicado al mundo del derecho lo que le permite conocer perfectamente las peculiaridades técnicas y legales que diferencian a la información jurídica y económica. Uno de sus productos es Infolex el cual es líder desde 1988 y decano del software de Gestión Jurídica. Este software trabaja en varios módulos: expedientes, seguimiento de expedientes de actuaciones, escritos, parte diario, procesos diarios y gestión de cobros, minutación-facturación, listín electrónico, agenda de despacho, contabilidad, tributación, listados e informes, bases de datos, herramientas. A pesar de las numerosas funcionalidades de este sistema, no es posible su utilización en los TPC debido a que los procesos que maneja no se ajustan a los que se necesitan informatizar, fue diseñado y construido para España por lo que sus procesos se rigen por las legislaciones de dicho país. Sin embargo es válido destacar que se tomaron ideas del mismo para la solución a desarrollar, por ejemplo, la forma en que implementan la seguridad, Infolex permite la creación de un esquema de seguridad dentro del despacho y el control de acceso de los usuarios a las distintas funciones del programa a partir de una clave personal, lo que garantiza la confidencialidad e integridad de la información que se maneja, aspecto de gran importancia para los clientes.

Lexnet, Sistema de gestión de notificaciones telemáticas (España): El funcionamiento de Lexnet se basa en un sistema de correo electrónico seguro, con firma electrónica, a través del cual el usuario recibe las notificaciones emitidas por el juzgado y presenta los escritos por vía telemática. Posteriormente, y a través del mismo sistema, recibe un resguardo electrónico en el que se acredita que la transmisión se ha efectuado correctamente y se le comunica la fecha efectiva de la presentación de dicho escrito en la oficina judicial o juzgado correspondiente. Su uso se regula en el Real Decreto 84/2007, de 26 de enero, sobre implantación en la administración de justicia del sistema informático de telecomunicaciones Lexnet para la presentación de escritos y documentos, el traslado de copias y la realización de actos de comunicación procesal por medios telemáticos.(Hernández, 2010)

Dentro del marco funcional que comprende la realización de actos de comunicación, la presentación de escritos con traslado de copias y de escritos iniciadores, el sistema Lexnet posee las siguientes funcionalidades: presentación de documentos y adjuntos al órgano judicial, envío de notificaciones por parte del órgano judicial, emisión de acuses de recibo, acceso autenticado y seguro mediante navegador web, acceso e intercambio de documentos mediante servicios web, gestión de carpetas de usuario, búsqueda y traza de mensajes. Este sistema se basa solo en la gestión de notificaciones y documentos, funcionalidades necesarias en la solución a implementar, pero que por sí solas no resuelven los problemas de los TPC, sin embargo aportan ideas a la solución como el intercambio de información mediante servicios web.

Gedex, software jurídico (España): que realiza el seguimiento completo de los expedientes de un despacho, bufete o departamento jurídico. Uno de los más utilizados en España y Latinoamérica; sus inicios se remontan al año 1996 y actualmente cuenta con un gran número de licencias vendidas a diferentes despachos jurídicos. Gestiona expedientes así como sus contactos asociados, almacena, recupera y gestiona la documentación legal. Ofrece un sistema de contraseñas, con el que puede limitar el acceso a la información, ocultar expedientes y contactos a ciertos pasantes o empleados. El sistema Gedex está básicamente enfocado al seguimiento de los expedientes y la documentación en sentido general, la manera de realizar los procesos no se ajusta a cómo se hace en Cuba ya que este sistema tienes sus fundamentos en las legislaciones de España y no se ajusta a las cubanas, es decir el flujo por el que deben tramitar los expedientes no es el mismo, razón ésta por la que no es factible para ser usado en los TPC.

Lex-Doctor, Sistema para la gestión jurídica (Argentina): Este completo sistema es utilizado en la amplia mayoría de los estudios jurídicos y asesorías letradas informatizados de la Argentina desarrollado

por Sistemas Jurídicos SRL, empresa que tiene un importante posicionamiento en el segmento de los sistemas aplicados a la actividad jurídica en Latinoamérica. Su tecnología de administración de datos, permite manejar grandes volúmenes de información, y organizar grupos de trabajo operando en redes de gran cantidad de terminales.

Principales funcionalidades:

- ✓ Gestión de expedientes: Procesos judiciales, extrajudiciales, mediaciones y todo tipo de expediente de índole jurídico. Partes, letrados, agendas, pruebas, gestiones, movimientos, audiencias, vencimientos. Manejo total de la procuración.
- ✓ Gestión económica: Cuentas por cliente, por expediente, y otros tipos de cuenta definibles por el usuario. Actualizaciones y liquidaciones, con conversión de monedas. Liquidaciones individuales o masivas.
- ✓ Gestión documental: Confección automatizada de escritos, cédulas, oficios, mandamientos, telegramas, cartas, y otros documentos, con posibilidad de confección seriada. Almacenamiento de documentos creados por el sistema y por otros programas instalados en la PC.
- ✓ Reportes y estadísticas: Confección de listados e informes, con diseños programados o propios, y con posibilidad de exportar a distintos formatos. Evaluación estadística gráfica.
- ✓ Acceso remoto: Opcionalmente, permite operar a través de Internet; así, disponiendo de una conexión y una PC, se puede trabajar en línea con la oficina desde cualquier lugar.

Lex-Doctor a pesar de ser un sistema con mucho prestigio y aceptación presenta diferentes inconvenientes para su utilización por parte de los TPC puesto que el mismo está desarrollado sobre software propietario, impidiendo las posibilidades de modificación y adaptación al medio nacional, además centra su trabajo sobre la gestión documental fundamentalmente, y para su utilización es necesario comprar la licencia de uso.

3.2. Sistemas estudiados en Cuba

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

SisProp: Sistema para la tramitación de procesos penales: Sistema informático desarrollado en la provincia de Villa Clara en el 2008. La propuesta inicial fue que abarcara la instancia suprema y provincial de la materia penal, desarrollándose solamente la tramitación de los procesos en la última instancia. Facilita la tramitación de los procesos penales, con agilidad y precisión, en las entidades del sistema de

tribunales populares del país. Dos de las características fundamentales del sistema son su seguridad dada la naturaleza de la información que se maneja, y la flexibilidad con respecto a cualquier modificación que pudiera sufrir la Ley Procesal Penal.

Principales deficiencias:

- ✓ No capta ningún dato de la fase judicial de la tramitación y decisión del tribunal.
- ✓ No aporta estadística, ni información alguna.
- ✓ No posee una exhaustiva validación de los datos.
- ✓ No se trabajó con un NIP[9] de cada proceso.
- ✓ Programado en Delphi, corre sobre SQL Server por lo que no es compatible con el software libre en el que se está programado los sistemas generales de cada materia judicial en la solución informática SIT.

Este sistema desarrollado en Cuba, no cumple con las expectativas para las que fue creado, no es posible integrarlo a la solución informática SIT debido a que las tecnologías de desarrollo no son compatibles con las definidas para el SIT, ya que éstas son más avanzadas. Esta aplicación al no registrar ningún dato de la tramitación no emite informes estadísticos, quedando por debajo de las expectativas que se tienen con la creación del SIT.

SisEco: Sistema Económico: Sistema informático desarrollado en Ciudad de la Habana en el 2002, concebido para el área de la estadística en el procedimiento Económico. El sistema cuenta con funcionalidades muy básicas y es lento en cuanto a tiempo de respuesta. En él se insertan los documentos radicados manualmente y las salvas diariamente se guardan en disquetes. La secretaria de estadística recoge el libro de radicación de escritos (LRE) e inserta los datos en la aplicación y cada cinco años borra la información, quedando solamente asentada en los libros.

Principales deficiencias:

- ✓ Inserta documentos radicados manualmente. La secretaria de estadística recoge el libro LRE2 e inserta los datos en la aplicación y cada cinco años borra los datos de los años anteriores quedando solamente la información asentada en los libros.
- ✓ Las salvas se guardan en disquetes. (Juiz, 2009)

Al igual que la anterior, esta aplicación no cumplió las expectativas iniciales de su creación. Es lento, característica que no va aparejada con la estadística en tiempo real. Prácticamente no informatiza nada, pues la radicación se realiza de forma manual, las salvas se guardan en disquetes técnica que ha

[9]Número de Identificación Permanente o General.

quedado obsoleta en nuestros días, la información almacenada es borrada cada cinco años, algo incomprensible para un sistema judicial para el cual mantener registros es primordial, ya que la información es el mecanismo principal para la justicia, por lo que tampoco es posible integrar la solución al SIT.

4. Fundamentación de las herramientas, metodología y tecnologías a utilizar

4.1. Metodología de Desarrollo de Software: RUP (Proceso Unificado de Rational)

Desarrollar un software con calidad depende de un sinnúmero de actividades y etapas, donde el impacto de elegir la mejor metodología para un equipo, en un determinado proyecto es trascendental para el éxito del producto. (Figueroa, 2008)

Las Metodologías de Desarrollo de Software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental cuando se desarrolla un producto de software. Existen numerosas metodologías, las mismas se pueden clasificar en dos grandes grupos:

- ✓ Las metodologías orientadas al control de los procesos, estableciendo rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán. Estas metodologías son llamadas Metodologías pesadas.
- ✓ Las metodologías orientadas a la interacción con el cliente y el desarrollo incremental del software, mostrando versiones parcialmente funcionales del software al cliente en intervalos cortos de tiempo, para que pueda evaluar y sugerir cambios en el producto según se va desarrollando. Estas son llamadas Metodologías ligeras/ágiles.

Dado la naturaleza y magnitud del proyecto del que forma parte esta investigación fue definido en el documento de arquitectura del proyecto TPC la utilización de RUP como metodología de desarrollo de software aprovechando las ventajas que ofrece la metodología como la mitigación temprana de posibles riesgos, temprana retroalimentación, aplicación del conocimiento adquirido de una iteración en las restantes, progreso visible en las primeras etapas y principalmente gestión de la complejidad, al ser éste un proyecto de cuantiosos procesos complejos. RUP es uno de los procesos más generales existentes en la actualidad y el equipo de trabajo está familiarizado con la utilización del mismo, incluye artefactos, entre ellos documentos solicitados por el cliente y necesarios en el proyecto ITPC debido al constante cambio de personal. Los proyectos realizados utilizando RUP cuentan con cuatro fases: inicio, elaboración, construcción y transición.

4.2. Herramienta de Ingeniería de Software Asistido por Ordenadores (CASE) para el modelado

4.2.1. Visual Paradigm 6.4

Visual Paradigm es una herramienta CASE que se utiliza para realizar el modelado de diagramas usando como lenguaje de notación UML entre otros. Proporciona muchas facilidades tales como: la generación de código para varios lenguajes, ingeniería inversa, generación de informes. Es la herramienta ideal para un entorno de software libre, también permite la creación de diagramas en un ambiente totalmente visual. Con una clase de diseño bien especificada, Visual Paradigm puede generar código en lenguajes como PHP, que es el que se utilizará en el desarrollo del SIT.

Tiene una interfaz muy intuitiva y es de fácil aprendizaje para los desarrolladores. Permite además hacer descripción de los casos de uso dando una gran variedad de plantillas predeterminadas con la posibilidad de personalizarlas. Con Visual Paradigm los analistas pueden generar la documentación necesaria de los artefactos obtenidos hasta el momento en el proyecto. Fue definido utilizar Visual Paradigm 6.4 como herramienta CASE en el proyecto TPC principalmente porque la UCI tiene licencia para su uso y entre las ventajas que ofrece actualmente cumple con las políticas de migración a software libre en Cuba, ya que es una herramienta multiplataforma que se puede utilizar tanto en Linux como en Windows.

4.3. Lenguaje de programación para el desarrollo Web: PHP 5.2.5

PHP (acrónimo de Hypertext Preprocessor) posee una amplia comunidad de desarrollo la cual implementa constantemente mejoras en su código, en la UCI este lenguaje es utilizado por muchos programadores debido a las diversas ventajas que provee y que son de gran beneficio para los mismos. Es un lenguaje “del lado del servidor” (esto significa que PHP funciona en un servicio remoto que procesa la página web antes de que sea abierta por el navegador del usuario) especialmente creado para el desarrollo de páginas web dinámicas. Puede ser incluido con facilidad dentro del código HTML. Entre sus principales características se destacan: que es gratuito, se integra de manera sencilla con múltiples gestores de bases de datos y tiene un gran número de funciones previamente definidas como por ejemplo funciones para el trabajo con fechas, arreglos y cadenas de texto, fueron utilizadas en la implementación del subsistema Administración y Gobierno, entre otras.

Principales Ventajas que ofrece:

- ✓ Muy fácil de aprender.
- ✓ Se caracteriza por ser un lenguaje rápido.
- ✓ Es un lenguaje multiplataforma: Linux, Windows, entre otros.
- ✓ Capacidad de conexión con la mayoría de los manejadores de base de datos como PostgreSQL.
- ✓ Capacidad de expandir su potencial utilizando módulos.

- ✓ Posee documentación en una página oficial la cual incluye descripción y ejemplos de cada una de sus funciones.
- ✓ Es libre, por lo que se presenta como una alterativa de fácil acceso para todos.

4.4. Sistema gestor de base de datos: PostgreSQL 8.4

PostgreSQL es un Sistema de Gestión de Base de Datos de Objetos Relacional (ORDBMS). Es ampliamente considerado como una de las alternativas de sistemas de base de datos de código abierto que se ha ganado una sólida reputación de confiabilidad, integridad de datos y corrección. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multi-hilos para garantizar la estabilidad del sistema. Es un sistema seguro y puede soportar grandes volúmenes de datos. (PostgreSQL, 2008)

Características fundamentales:

- ✓ Transacciones: Permiten el paso entre dos estados consistentes manteniendo la integridad de los datos.
- ✓ Integridad referencial: PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.
- ✓ Bloqueos de tablas y filas: PostgreSQL ofrece varios modos de bloqueo para controlar el acceso concurrente de los datos en tablas.
- ✓ Restricciones (constraints) y disparadores (triggers): Tienen la función de mantener la integridad y consistencia de la base de datos.
- ✓ Múltiples tipos de datos predefinidos: Como todos los manejadores de base de datos, PostgreSQL implementa los tipos de datos definidos para el estándar SQL3 y aumenta algunos datos.
- ✓ Soporte de tipos y funciones de usuarios: PostgreSQL soporta operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario. Incorpora una estructura de datos de arreglo.
- ✓ Interfaz con diversos lenguajes, entre ellos PHP.

4.5. Marcos de trabajo

Un marco de trabajo (framework) es una estructura de soporte definida mediante la cual otro proyecto de software puede ser desarrollado y organizado. Puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Son diseñados con el intento de facilitar el desarrollo de software, permitiendo al programador abstraerse de un conjunto de detalles de bajo nivel para proveer un sistema funcional.

4.5.1. Zend Framework 1.5.0

Conocido también como Zf, Zend Framework es un marco de trabajo de código abierto para desarrollar tanto aplicaciones como servicios web en PHP5. Usa una implementación completa orientada a objetos. Sus componentes fueron desarrollados con una baja dependencia unos de otros, lo que posibilita al programador el uso de estos por separado. Ofrece un alto rendimiento y una robusta implementación del patrón Modelo Vista Controlador (MVC). Incluye una gama de componentes, que le permiten al programador realizar toda la abstracción de la base de datos, la autentificación de usuarios, el acceso a los servicios web, entre otras. Es un marco de trabajo extensible, reutilizable y que puede ser integrado fácilmente con otros marcos de trabajos y librerías de clases. Tiene una buena documentación y una amplia comunidad de desarrolladores.

A partir de la extensión de algunos componentes de Zend Framework surge ZendExt, desarrollado por el Departamento de Tecnología y la UCID (Unidad de Compatibilización Integración y Desarrollo de Software para la Defensa), con el objetivo de crear un marco de trabajo extensible y configurable centrando el desarrollo de las aplicaciones, en la lógica del negocio, 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.

La unión de ZendExt, Doctrine y ExtJS dio lugar al marco de trabajo Sauxe. Los 51 componentes de Zend Framework conforman un potente y extensible marco de trabajo de aplicaciones web al combinarse entre sí. De los 51 Sauxe utiliza solamente los siguientes:

Componentes			
1. Zend_Cache	3. Zend_Controller	5. Zend_Log	7. Zend_Session
2. Zend_Config	4. Zend_Loader	6. Zend_Registry	8. Zend_View

Tabla 1 Componentes de Zend Framework que utiliza Sauxe

4.5.2. Sauxe 2.0

Sauxe es un marco de trabajo que contiene un conjunto de componentes reutilizables. Provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. (Nemury y otros, 2008)

Contiene la librería JavaScript ExtJS, que es utilizada en la capa de presentación, por la gran gama de componentes que se pueden reutilizar para agilizar el proceso de desarrollo y mostrarle al usuario una interfaz más amigable y funcional. Está integrado al Generador de Reportes Dinámicos (GDR) desarrollado por el Centro de Datos de la UCI.

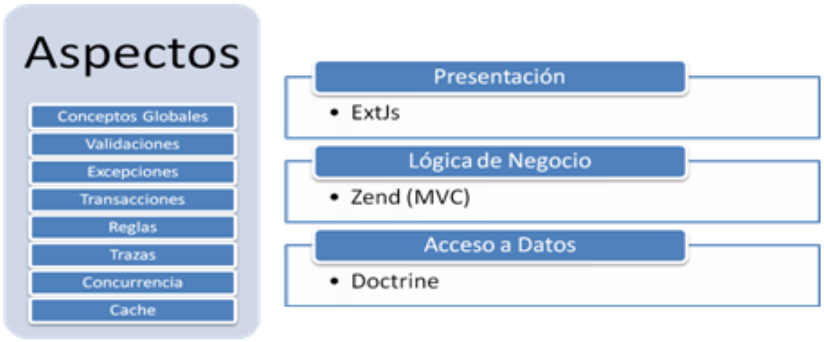


Figura 1 : Arquitectura de Sauxe

El marco de trabajo Sauxe tiene definido para su uso una serie de herramientas específicas, como sistema gestor de base datos PostgreSQL 8.3 u 8.4 solamente, siendo ésta una limitación. Los lenguajes de programación están definidos, del lado del servidor PHP y del lado del cliente JavaScript. Todas estas herramientas son libres siguiendo el paradigma de independencia tecnológica por el cual apuesta el país. La arquitectura de la plataforma tiene varias ventajas en diferentes tipos de escenarios arquitectónicos. Permite configurar y gestionar de manera dinámica la caché del marco de trabajo y la integración de las aplicaciones o dominios de soluciones que se instancien o construyan con la misma. Posee mecanismos de abstracción de la capa relacional de persistencia y de administración de transacciones. Las operaciones de modificación sobre el modelo de dominio son transaccionales por defecto, lo que constituye una ventaja. Incluye además la capacidad de definir familias de excepciones. Presenta una arquitectura basada en capas. Está compuesto básicamente por cinco niveles o capas: Capa de Presentación, Capa de Control o Negocio, Capa de Acceso a Datos, Capa de Datos y Capa de Servicios. Sauxe en su versión 2.0 está formado por los componentes:

Componentes		
ZendExt_App	ZendExt_MVC	ZendExt_Nomencladores
ZendExt_Aspect	ZendExt_Exception	ZendExt_Trace
ZendExt_ADT	ZendExt_FastResponse	ZendExt_Validation
ZendExt_Cache	ZendExt_GlobalConcept	ZendExt_Portal
ZendExt_ExpImp	ZendExt_IoC	ZendExt_TransactionManager

Tabla 2 Componentes que integran el marco de trabajo Sauxe

4.5.3. ExtJS 2.2

ExtJS es una librería JavaScript para la programación del lado del cliente. Es una potente herramienta para crear las interfaces de usuarios de una aplicación web. Incluye un conjunto de clases, y métodos que facilitan su vinculación con el lenguaje utilizado del lado del servidor. Ofrece gran cantidad de controles (widgets) para crear interfaces de usuario complejas. Sauxe incluye PHP-Ext, librería de código abierto que permite potenciar la capa de interfaz de usuario de JavaScript en las aplicaciones. Para ello ofrece una serie de librerías para integrar ExtJS en el sistema a desarrollar. Entre las posibilidades que ofrece se encuentran la creación de formularios, combos, paneles de tablas o menús, componentes utilizados en la implementación de la solución. Además ayuda a la comunicación entre el cliente y el servidor mediante JSON (notación de objetos de JavaScript) y XML (lenguaje de marcas extensible).

Tiene un sistema dual de licencia: Comercial y Código Abierto (Open Source). En tiempo de ejecución carga y crea todos los objetos HTML a través del uso de DOM (Modelo de Objetos del Documento).

Ventajas de utilizar ExtJS:

- ✓ Código reutilizable.
- ✓ Independiente o adaptable a diferentes marcos de trabajo.
- ✓ Orientada a la programación de interfaces de tipo escritorio en la web.
- ✓ Funciones comunes como validación, combos editables, ventanas deslizables (con minimizar y maximizar) y tablas editables, son muy fáciles de implementar.
- ✓ Implementación basada en patrones de diseño.
- ✓ Amplia librería de componentes gráficos fácilmente extensibles.

4.5.4. Doctrine 1.2.1

Doctrine es un potente y completo sistema ORM para PHP 5.2 o mayor, con una capa de abstracción de la base de datos (DBAL del inglés Database Abstraction Layer) incorporada. Entre otras ventajas ofrece la posibilidad de exportar una base de datos existente a sus clases correspondientes y también a la inversa, es decir convertir clases a tablas de una base de datos. Su principal ventaja radica en poder acceder a la base de datos utilizando la programación orientada a objetos (POO). Doctrine utiliza el patrón Active Record para manejar la base de datos, tiene su propio lenguaje de consultas de datos (DQL) y trabaja de manera rápida y eficiente. Es fácilmente integrado a los principales frameworks de desarrollo utilizados actualmente.

4.6. Entorno de Desarrollo Integrado (IDE): NetBeans 7.0.1

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

Pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. NetBeans es un IDE especialmente diseñado para el desarrollo de aplicaciones en Java, pero que acepta otros lenguajes de programación. Consta de una gran base de usuarios y una comunidad en constante crecimiento, lo que le ha permitido, al igual que muchos otros sistemas libres, el progreso paulatino de sus prestaciones y la eliminación de errores de programación (bugs) que pudiesen existir.

Ventajas del uso del NetBeans:

- ✓ La plataforma NetBeans permite que las aplicaciones se desarrollen a partir de un conjunto de módulos o componentes de software.
- ✓ NetBeans IDE es fácil de instalar y de uso instantáneo, se ejecuta en varias plataformas incluyendo Windows y Linux.
- ✓ NetBeans IDE es la herramienta ideal para el desarrollo de software con PHP, AJAX y JavaScript.

4.7. Servidor web: Apache 2.0

Los servidores web son indispensables para la publicación y explotación de un sistema que esté desarrollado para su uso en la web. Estos servidores contienen la lógica para la interpretación de las peticiones de los usuarios a través de los protocolos específicos, que hace posible la interacción dinámica entre el cliente y sus acciones sobre el sistema en que esté trabajando. Apache se caracteriza por ser un servidor ligero, altamente configurable y de amplia explotación, según Netcraft, empresa dedicada a la realización de encuestas a nivel global y estudios sobre el tráfico en internet, el mayor por ciento de los servidores web actuales son servidores Apache.

Principales características de Apache:

- ✓ Tecnología gratuita de código fuente abierta.
- ✓ Multiplataforma, funcionando tanto en Windows como en Linux.
- ✓ Soporte para interfaz de entrada común (CGI del inglés Common Gateway Interface).
- ✓ Soporte para varios lenguajes, entre ellos PHP.
- ✓ Soporte para el protocolo HTTP.
- ✓ Estable.

4.8. Patrones de arquitectura

4.8.1. Modelo Vista Controlador (MVC)

Modelo–Vista–Controlador es un patrón de arquitectura de software. Separa conceptualmente la representación visual de la aplicación, las acciones que intercambian datos y el modelo de negocio y su dominio. En el SIT se concreta con la identificación de 3 elementos diferentes: la vista implementada en

Java Script o HTML reside del lado del cliente en tiempo de ejecución, el Controlador, y 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 o peticiones HTML y la respuesta dada por el controlador puede encontrarse en JSON, XML o HTML según corresponda la solicitud. (Productiva, 2011) La finalidad del MVC es mejorar la reusabilidad por medio del desacople entre la vista y el modelo.

Ventajas de utilizar MVC

- ✓ Es posible tener diferentes vistas para un mismo modelo (ej. representación de un conjunto de datos como una tabla).
- ✓ Es posible construir nuevas vistas sin necesidad de modificar el modelo subyacente.
- ✓ Proporciona un mecanismo de configuración a componentes complejos muchos más tratable que el puramente basado en eventos (el modelo puede verse como una representación estructurada del estado de la interacción).(Sebastián, 2010)

4.8.2. Modelo Cliente-Servidor

La tecnología Cliente/Servidor es el procesamiento cooperativo de la información por medio de un conjunto de procesadores, en el cual múltiples clientes, distribuidos geográficamente, solicitan requerimientos a uno o más servidores centrales. (TIC, 2011)

Es un modelo basado en la idea del servicio, en el que el cliente es un proceso consumidor de servicios y el servidor es un proceso proveedor de servicios. Además esta relación está establecida en función del intercambio de mensajes que es el único elemento de acoplamiento entre ambos. Los tres elementos fundamentales sobre los cuales se desarrollan e implantan los sistemas Cliente/Servidor son: el proceso cliente que es quien inicia el diálogo, el proceso servidor que pasivamente espera a que lleguen peticiones de servicio y el middleware[10] que corresponde a la interfaz que provee la conectividad entre el cliente y el servidor para poder intercambiar mensajes. Se caracteriza por existir un nodo (o más) servidor donde reside el servicio que se expone y varios clientes que consumen dicho servicio, en el SIT este estilo responde al hecho de que se trate de una aplicación web y se concreta con un servidor de bases de datos (PostgreSQL) un servidor web (con función de servidor de aplicaciones, Apache 2) y varios clientes que acceden al sistema a través de un navegador. (Productiva, 2011)

[10]Software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones.

4.8.3. Multicapas

La arquitectura multicapas está basada en tres tipos de capas, que son: Presentación o Interfaces, Reglas de Negocios y Datos.

Figura 2: Patrón de arquitectura Multicapas

La Primera Capa (Presentación o Interfaces), es la página inicial que permite al usuario ver el diseño del programa.

La Segunda Capa (Reglas de Negocios), es la capa intermedia del programa donde se maneja las transacciones y reglas del negocio, actuando como intermediario entre las interfaces del usuario y la otra capa que es la de los datos.

La Tercera Capa (Datos), esta es la capa más importante del programa, ya que maneja la información basada en una plataforma potente permitiendo así una consistencia en la información. (Perdomo, 2008)

La arquitectura multicapas se caracteriza por organizar los componentes del sistema en capas con responsabilidades bien definidas y donde las capas del nivel más alto invocan los servicios de las del nivel inferior. (Productiva, 2011)

Conclusiones parciales

En este capítulo se realizó un estudio de sistemas que realizan procesos similares a los que se desarrollan en la investigación, se exponen algunos ejemplos de los mismos, lo que contribuye con conocimientos e ideas para la realización del sistema. Del estudio del estado del arte se concluye que hasta el momento no existe una solución informática que facilite íntegramente el desarrollo de un proceso judicial similar a los que tienen lugar en los TPC, por lo que es necesario implementar un sistema que responda a las necesidades de los mismos.

Se realizó un análisis de las tecnologías informáticas a emplear a lo largo del desarrollo de la solución del problema, las cuales se listan a continuación y fueron previamente definidas en el Documento de

Arquitectura del proyecto Tribunales Populares Cubanos.

- ✓ Lenguaje de programación: PHP 5.2.5
- ✓ Marco de trabajo: Sauxe 2.0
- ✓ Herramienta CASE para el modelado: Visual Paradigm 3.4
- ✓ Servidor web: Apache 2.0
- ✓ Sistema Gestor de Base de Datos: PostgreSQL 8.4
- ✓ IDE de desarrollo: NetBeans 7.0.1

Capítulo 2: Propuesta de solución

Introducción

En el presente capítulo se propone la solución técnica de la investigación. Se presenta el Modelo de diseño conformado por los Diagramas de clases y los Diagramas de secuencia (Diagramas de interacción), el Modelo de implementación con los Diagramas de componentes y el Diagrama de despliegue. Se exponen además los patrones de diseño utilizados en la solución propuesta, los estándares de codificación definidos para la implementación y el diseño de la base de datos mediante los modelos físico y lógico.

1. Estándares de codificación

- Identación[11]

El contenido siempre se identará con tabs, nunca utilizando espacios en blanco.

- Cabecera del archivo.

Es importante que todos los archivos .php inicien con una cabecera específica que indique información de la versión, autor de los últimos cambios, etc.

/* * Componente para gestionar los sistemas.
* @package Administración y Gobierno SIT
* @copyright TPC Cuba
* @author Daylenis Sánchez Delgado
*@modify 9-04-2012
* @version 1.0-0 */

- Comentarios en las funciones.

Todas las funciones deben tener un comentario, antes de su declaración, explicando qué hacen.

```
/**
 * Description
 * Método para cargar el DPA(División Política Administrativa) de un país o provincia
 *  en dependencia del id que le pasen por parámetro )
 */
function cargardpaAction() {

    $idpais = $this->_request->getPost('idndpa');
    $resultado = array();
    $dpa = new NdpaModelExt;
```

Figura 3: Ejemplo de descripción de los métodos

[11] En el contexto de informática significa mover un bloque de texto hacia la derecha insertando espacios o tabuladores para separarlo del texto adyacente, lo que en el ámbito de la imprenta se ha denominado siempre como sangrado o sangría.

- Clases.

Las clases serán colocadas en un archivo .php aparte, donde sólo se colocará el código de la clase. El nombre del archivo será el mismo de la clase y siempre empezará en mayúscula. En lo posible, procurar que los nombres de clase 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.

Estilo y reglas de escritura de código PHP

- Nombres de variables.

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación Pascal-Casing[12]. Los nombres de las variables deben ser descriptivos y concisos, no usar ni grandes frases ni pequeñas abreviaciones. Siempre es mejor saber qué hace una variable con sólo conocer su nombre. Esto se aplica para los nombres de variables, funciones, argumentos de funciones y clases.

  - ✓ En las funciones, es importante que el nombre denote su función inmediatamente. Cosas como ImprimirDatos están bien, pero estaría mejor ImprimirDatosUsuario. De igual manera, en los argumentos de las funciones es necesario saber inmediatamente qué se está usando. Es mejor CrearUsuario($nick, $email) que Crear($n, $e).
  - ✓ No se debe dañar la legibilidad del código por pereza, aplicar el sentido común y no crear funciones de más de 4 palabras.

- Siempre incluir las llaves.

En todo momento cuando se vaya a codificar un bloque de instrucciones, éste debe ir encerrado entre llaves, incluso cuando conste de una sola línea.

- ¿Dónde colocar las llaves?

Todos los corchetes van en una línea propia.

- Precedencia de operadores.

Lo mejor es siempre usar paréntesis para estar seguro de la precedencia de los operadores. Básicamente, la idea es no codificar operaciones complejas y estar seguros que los compañeros de equipo con menos habilidad comprendan todo sin problemas.

[12] Pascal-Casing es como la notación húngara (ej. intEdad) pero sin prefijos. En este caso, los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula.

- Cadenas de texto entre comillas.

PHP tiene dos formas de construir cadenas de texto, con comillas simples y con comillas dobles. La diferencia es que si se usan comillas dobles y se coloca dentro del texto el identificador de una variable, el compilador lo interpretará y reemplazará por su valor. Por ésta razón siempre se han de usar comillas simples a menos que se necesite hacer la interpolación de variables que permiten las dobles. Hay casos especiales donde es mejor usar comillas dobles, por ejemplo cuando se usan caracteres de escape, así que se puede romper ésta regla cuando sea para mejorar la lectura del código.

- No utilizar variables sin inicializar.

Si no se tiene control sobre el valor de una variable, se debe verificar que esté inicializada.

Nomenclatura según el tipo de clases.

- Clases controladora: Las clases controladoras después del nombre llevan la palabra:"Controller" (ej. GestionarUsuarioController).
- Clases de los modelos.

Business (Negocio): Las clases que se encuentran dentro de la carpeta business después del nombre llevan la palabra: "Model" (ej. GestionarUsuarioModel).

Domain (Dominio): Las clases que se encuentran dentro de la carpeta domain el nombre que reciben es el de la tabla en la Base de Datos (ej. Usuario)

Generated (Dominio bases): Las clases que se encuentran dentro de la carpeta generated el nombre comienza con la palabra: "Base" y seguido el nombre de la tabla en la Base de Datos (ej. BaseUsuario).

- Clases del framework.

Como parte del marco de desarrollo de Zend existe el Zend_Loader (Cargador) que, además del cumplimiento de ciertas normas para la nomenclatura de las clases, garantiza que a partir de una ruta de inclusión, este sea el responsable de la inclusión de los recursos requeridos en el proceso. Ejemplo: Los nombres de las clases contienen la dirección donde se encuentran, de la siguiente forma, si tenemos una clase llamada Condado en la siguiente estructura Cuba/VC/SantaClara/Condado.php entonces un identificador de la misma debe ser Cuba_VC_SantaClara_Condado lo que garantiza que a través de una casuística particular el cargador localice estos recursos.

2. Diseño

En la fase de diseño se modela el sistema enfocado al soporte de todos los requisitos, tanto funcionales como no funcionales, creándose así una entrada apropiada para las actividades de implementación. El

propósito del diseño es especificar una solución que trabaje y pueda ser fácilmente convertida en código fuente y construir una arquitectura simple y fácilmente extensible. (Ivar Jacobson, G. B. y. J. R, 2000)

2.1. Propósitos del diseño

- ✓ Adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia y tecnologías de interfaz de usuario.
- ✓ Crear una entrada apropiada y un punto de partida para actividades de implementación, capturando los requisitos o subsistemas individuales, interfaces o clases.
- ✓ Descomponer los trabajos de implementación en partes más manejables que pueden ser llevadas a cabo por diferentes equipos de desarrollo.
- ✓ Capturar las interfaces entre los subsistemas antes en el ciclo de vida del software, lo cual es muy útil cuando se utiliza interfaces como elementos de sincronización entre diferentes equipos de desarrollo.

2.2. Arquitectura del SIT

La arquitectura de software es el conjunto de técnicas metodológicas desarrolladas con el fin de facilitar la programación. Se refiere a un grupo de abstracciones y patrones que nos brindan un esquema de referencia útil para guiarnos en el desarrollo de software dentro de un sistema informático. Establece todos los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos y necesidades del sistema. Se necesita una arquitectura robusta, que guíe el proceso de desarrollo y que defina de manera abstracta los componentes que lleven a cabo alguna tarea, sus interfaces y la comunicación entre ellos.

El desarrollo del SIT responde a un estilo arquitectural en capas (n-layer) basándose en una distribución jerárquica de las responsabilidades para proporcionar una división efectiva de los problemas a resolver, de forma tal que cada capa contiene la funcionalidad relacionada solo con las tareas de esa capa, las capas inferiores no tienen dependencias de las capas superiores y la comunicación entre ellas está basada en una abstracción que les proporciona un bajo acoplamiento. Se hace uso del patrón arquitectónico Modelo–Vista–Controlador (MVC), el cual permite la reutilización e independencia entre las capas, además permite que se puedan realizar cambios en las mismas sin tener que modificar las otras capas, facilita la estandarización, la utilización de los recursos y la administración.

Capa de Presentación o Vista: En esta capa se emplea las facilidades que brinda el marco de trabajo ExtJS para la construcción de interfaces amigables a la vista de los usuarios. ExtJS centra su desarrollo en el uso de JavaScript, CSS y AJAX. Esta capa se comunica únicamente con la capa de negocio.

Capa de Control o Negocio: En esta capa se emplea el patrón de arquitectura MVC. De forma vertical al modelo descrito hasta este momento, estarán los aspectos fundamentales del sistema así como el encargado del tratamiento de la seguridad a nivel de aplicación Web.

Capa de Acceso a Datos o Modelo: En esta capa estará presente el Mapeador de Objetos Relacionales (ORM, Object Relational Mapping) Doctrine, como marco de trabajo de persistencia para la comunicación con el servidor de datos mediante el protocolo PDO (PHP Data Object), también estará un persistidor de configuración que es el encargado de comunicarse vía XML con los ficheros de configuración del sistema denominado respuesta rápida (Fast Response).

Capa de Datos: En esta capa estará ubicado como servidor de base de datos PostgreSQL y un conjunto de Ficheros de Configuración de la arquitectura tecnológica.

Capa de Servicio: En esta última capa se encuentran todos los subsistemas que prestan y consumen servicios entre sí. Esta gran estructura descrita, se comunica con unas series de servicios Web mediante protocolos SOAP, e IoC, que interactúan con el sistema proporcionándole un conjunto de funcionalidades tanto de seguridad como de negocio.

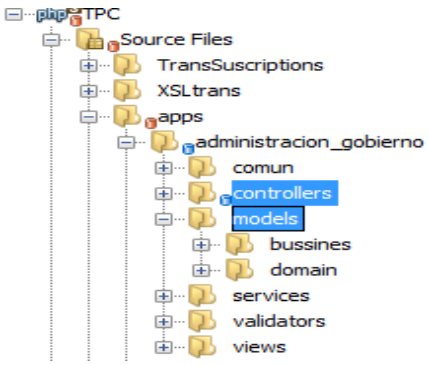


Figura 4: Estructura del paquete apps, donde se encuentran las clases controladoras y las del modelo, aplicando el patrón MVC

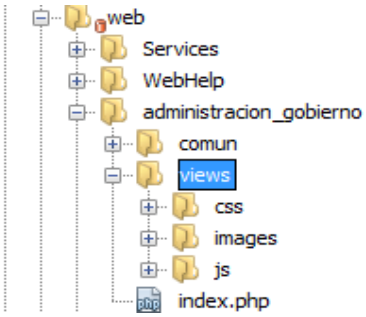


Figura 5: Estructura del paquete web, donde se encuentran las clases de la vista, aplicando el patrón MVC.

2.3. Patrones de diseño utilizados

2.3.1. 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. Se utiliza entre otras cosas para desacoplar las clases de sus dependencias posibilitando que las mismas puedan ser reemplazadas o actualizadas con muy pocos o casi ningún cambio en el código fuente de sus clases. Cuando se desea escribir clases que dependan de clases cuyas implementaciones no son conocidas en tiempo de compilación, además se desea testar las clases aisladamente sin sus dependencias y se quiere desacoplar sus clases de ser responsables de localizar y gestionar el tiempo de vida de sus dependencias. (Guillerón, 2009)

Este patrón es utilizado por el marco de trabajo Sauxe para crear y brindar servicios. Cada módulo del proyecto cuenta con un esquema propio en la base de datos y ninguno puede acceder directamente al otro, en el caso que un módulo necesite acceder al esquema de otro, lo que se hace es crear y brindar un servicio para que el que lo necesite lo consuma.

En las siguientes figuras 6, 7 y 8, se muestra el servicio para obtener datos de usuarios desde su creación en la clase GestUsuarioService, su declaración en el ioc-general.xml del módulo Administración y Gobierno y su consumo en el módulo Seguridad.

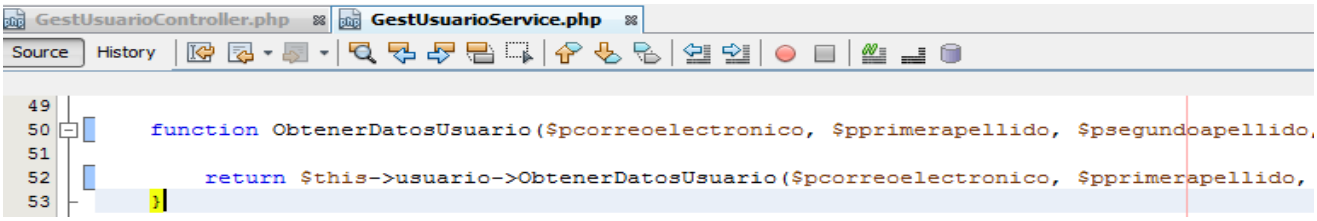


Figura 6: Representación de la clase GestUsuarioService, ubicada en el paquete apps\administracion _gobierno\services, donde se crea la función ObtenerDatosUsuario(…)

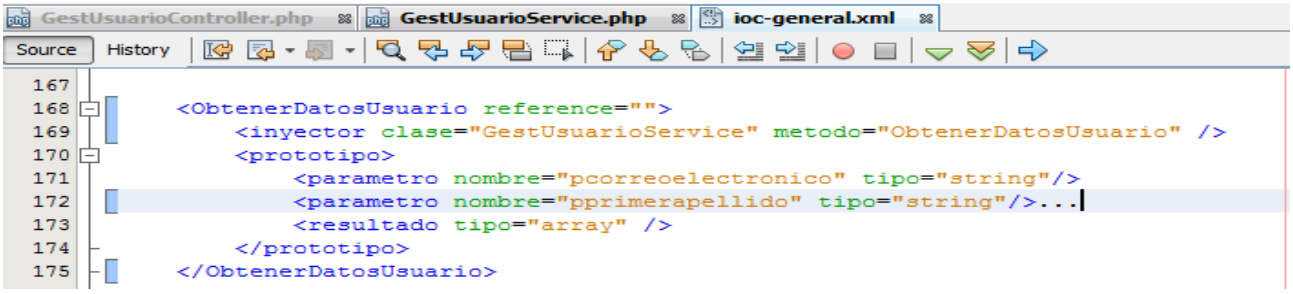


Figura 7: Declaración del servicio en el XML de configuración ioc-general.xml del módulo Administración y Gobierno

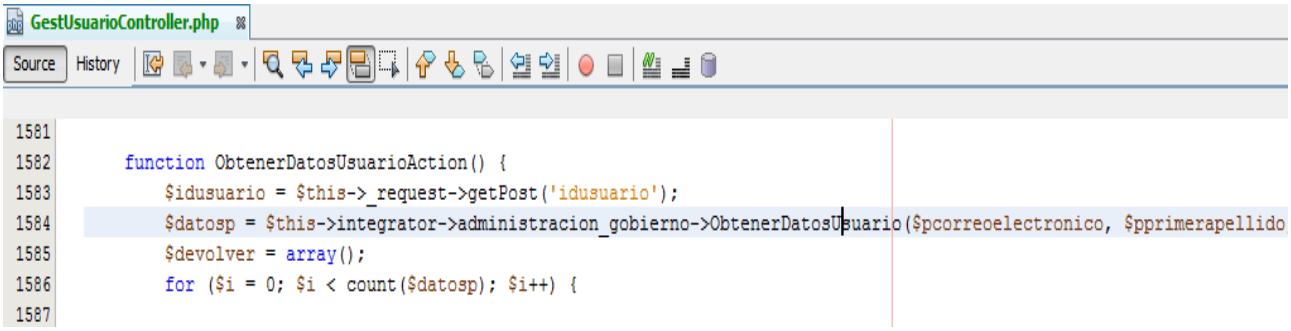


Figura 8: Consumo del servicio ObtenerDatosUsuario por la clase GestusuarioController del módulo Seguridad

2.3.2. Patrones GRASP[13]

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. (Gutierrez, 2007) Las responsabilidades están relacionadas con las obligaciones de un objeto en cuanto a su comportamiento. Básicamente, estas responsabilidades son de los siguientes dos tipos:

- ✓ Conocer: conocer los datos privados encapsulados, conocer los objetos relacionados, conocer las cosas que puede derivar o calcular.
- ✓ Hacer: hacer algo él mismo, como crear un objeto o hacer un cálculo, iniciar una acción en otros objetos, controlar y coordinar actividades en otros objetos.

Los patrones GRASP utilizados en la solución son los siguientes:

Experto: Este patrón se tiene en cuenta para asignar una responsabilidad al experto en información (la clase que tiene la información necesaria para la realización de la asignación).

[13]General Responsibility Assignment Software Patterns (Patrones de software generales de asignación de responsabilidades)

Bajo acoplamiento: El acoplamiento mide la fuerza con que una clase está conectada a otra, de esta forma una clase con bajo acoplamiento debe tener un número mínimo de dependencia con otras clases. Este patrón se tiene en cuenta para asignar una responsabilidad permitiendo que el acoplamiento permanezca bajo. Este patrón se evidencia en el marco de trabajo Sauxe ya que dentro de la capa modelo, las clases de abstracción de datos son las más reutilizables y no tienen asociaciones con las clases de la capa de la vista ni con el controlador, dependiendo solo de sus clases bases las que contienen los atributos correspondientes a su tabla en la base de datos.

```php
<?php

class Dlego extends BaseDlego {

    public function setUp() {
        parent::setUp();

        $this->hasOne('Dpersonanatural', array('local' => 'idpersona', 'foreign' => 'idpersona'));
    }

}
?>
```

Figura 9 Clase Dlego, perteneciente a la capa del modelo y que depende de la clase BaseDlego

Controlador: Soluciona el problema: ¿quién debería ser el responsable de gestionar un evento de entrada al sistema?, este patrón se tiene en cuenta para realizar las asignaciones en cuanto al manejo de los eventos del sistema y definir sus operaciones. Se encarga de que se utilice la misma clase controlador para todos los eventos del sistema en el mismo escenario de caso de uso. (Gutierrez, 2007)

En la estructura que propone Sauxe este patrón se evidencia en las clases controladoras que se encuentran en la carpeta: TPC/app/administracion_gobierno/controllers, dentro de esta carpeta se encuentran todas las clases controladoras del subsistema y así mismo para los demás módulos del SIT, dichas clases son las responsables de implementar todas las funcionalidades pertenecientes a una interfaz determinada, además son las que reciben los datos del usuario y los utilizan de acuerdo a la acción solicitada.

```
15  class GestUsuarioController extends ZendExt_Controller_Secure {
16
    function init() {...}
20
21  function GestUsuarioAction() {...}
24
25  function FichaUsuarioAction() {...}
31
32  function CargarRolesUsuarioAction() {...}
38
39  public function GetDatosUsuario($idusuario) {...}
75
76  function CargarUsuarioAction() {...}
196
197 function ObtenerDatosUsuarioAction() {...}
248
```

Figura 10 Clase GestUsuarioController encargada de implementar las funcionalidades de la interfaz de gestión de usuarios.

2.3.3. Patrones GOF[14]

Instancia única (Singleton): Su propósito es garantizar que una clase sólo tiene una única instancia, proporcionando un punto de acceso global a la misma. Sauxe que usa como base Zend Framework tiene una instancia única del controlador frontal disponible mediante este patrón para lograr una vía de entrada única a las solicitudes, otro ejemplo de su aplicación en el SIT se evidencia en la Figura 12 en la clase ZendExt_Event. Entre sus ventajas destacan que es fácilmente modificable para permitir más de una instancia y, en general, para controlar el número de las mismas (incluso si es variable), además se reduce el espacio de nombres (frente al uso de variables globales).

```
class ZendExt_Event {
        /**
         * Instancia del Xml de eventos
         * @var SimpleXml
         */

        private static $_instance;

        /** * Constructor */
        private function __construct() {
                $this->_xml = ZendExt_FastResponse::getXML('events');
                $this->_ioc = ZendExt_IoC::getInstance();
        }
        /**
         * Retorna una instancia de ZendExt_Event para el singleton
         * @return ZendExt_Event
         */
        static function getInstance () {
                if (self :: $_instance == null)
                        self :: $_instance = new self ();
                return self :: $_instance;
        }
```

Figura 11 Ejemplo de implementación del patrón Instancia única en la clase ZendExt_Event

[14] Gang of Four (La banda de los cuatro)

Fachada (Facade): Su propósito es proporcionar una interfaz unificada de alto nivel que, representando a todo un subsistema, facilite su uso. La "fachada" satisface a la mayoría de los clientes, sin ocultarlas funciones de menor nivel a aquellos que necesiten acceder a ellas. (Madrid., 2010) Este patrón simplifica los accesos a las clases de la capa de acceso a datos proporcionando un objeto que todas las clases de capas superiores utilizarán para acceder a las clases contenidas en la capa del modelo. Define una interfaz de más alto nivel que permite usar el sistema más fácil. El objetivo de la aplicación de este patrón es reducir la dependencia entre clases. Se utilizará una clase intermediaria entre las clases controladoras de Zend Framework y las de acceso a datos de Doctrine, la que brindará, de las operaciones de acceso a datos, solo las que necesiten los controladores para su funcionamiento, lo que reduce la dependencia de estos entre las múltiples clases de acceso a datos existentes en el sistema.

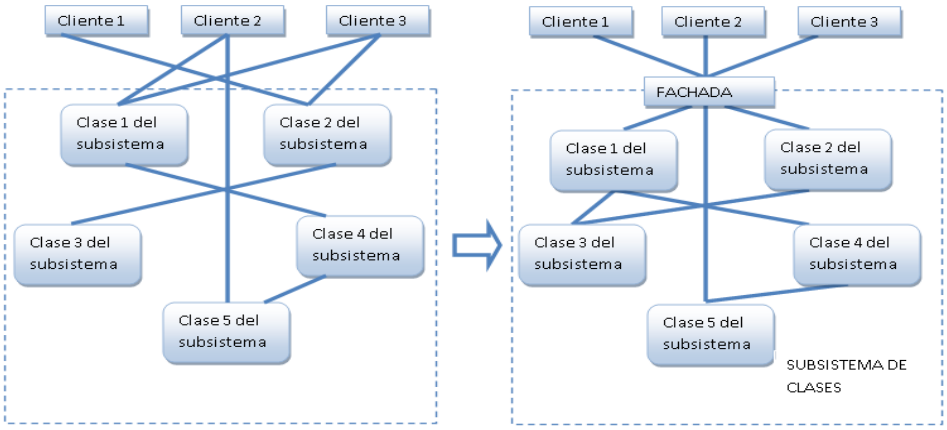


Figura 12: Estructura del patrón Fachada

2.4. Modelo de diseño

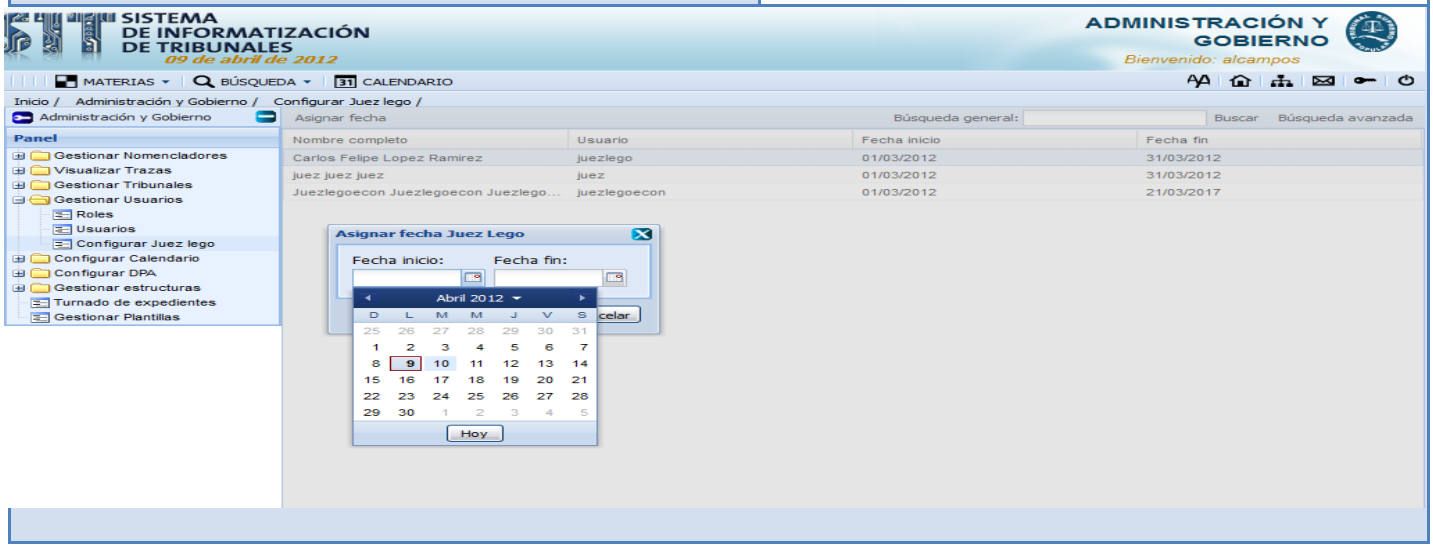
2.4.1. Realización de diagramas de clases del diseño

Una realización de casos de uso del diseño es una colaboración en el modelo de diseño que describe cómo se realiza un caso de uso específico, y como se ejecuta, términos de las clases de diseño y sus objetivos.

Tabla 3 Descripción de la sección Asignar Fecha del CU Configurar Juez Lego

Caso de Uso: Configurar Juez Lego.
Actor: Administrador.
Resumen: El caso de uso se inicia cuando el administrador necesita gestionar la fecha de trabajo de un juez lego asociado a un tribunal. Consiste en que el administrador busca el juez lego deseado y

selecciona la opción de asignar fecha a juez lego. El caso de uso termina con la actualización de un juez lego.	
Precondiciones: Debe ser inicializado por el administrador.	
Referencias: RF.19, RF.37, RF.38.	
Prioridad: Crítico	
Sección1: "Asignar Fecha"	
Acción del Actor	**Respuesta del Sistema**
1- El caso de uso inicia cuando el administrador selecciona un juez lego y solicita asignarle una fecha.	2- El sistema muestra una interfaz solicitando los siguientes datos: • Fecha de Inicio (Obligatorio). • Fecha Fin (Obligatorio).
3-Introduce los datos correspondientes y selecciona la opción "Aceptar".	4-El sistema valida que los datos introducidos son correctos y que no hay campos obligatorios vacíos. Terminando así el caso de uso.



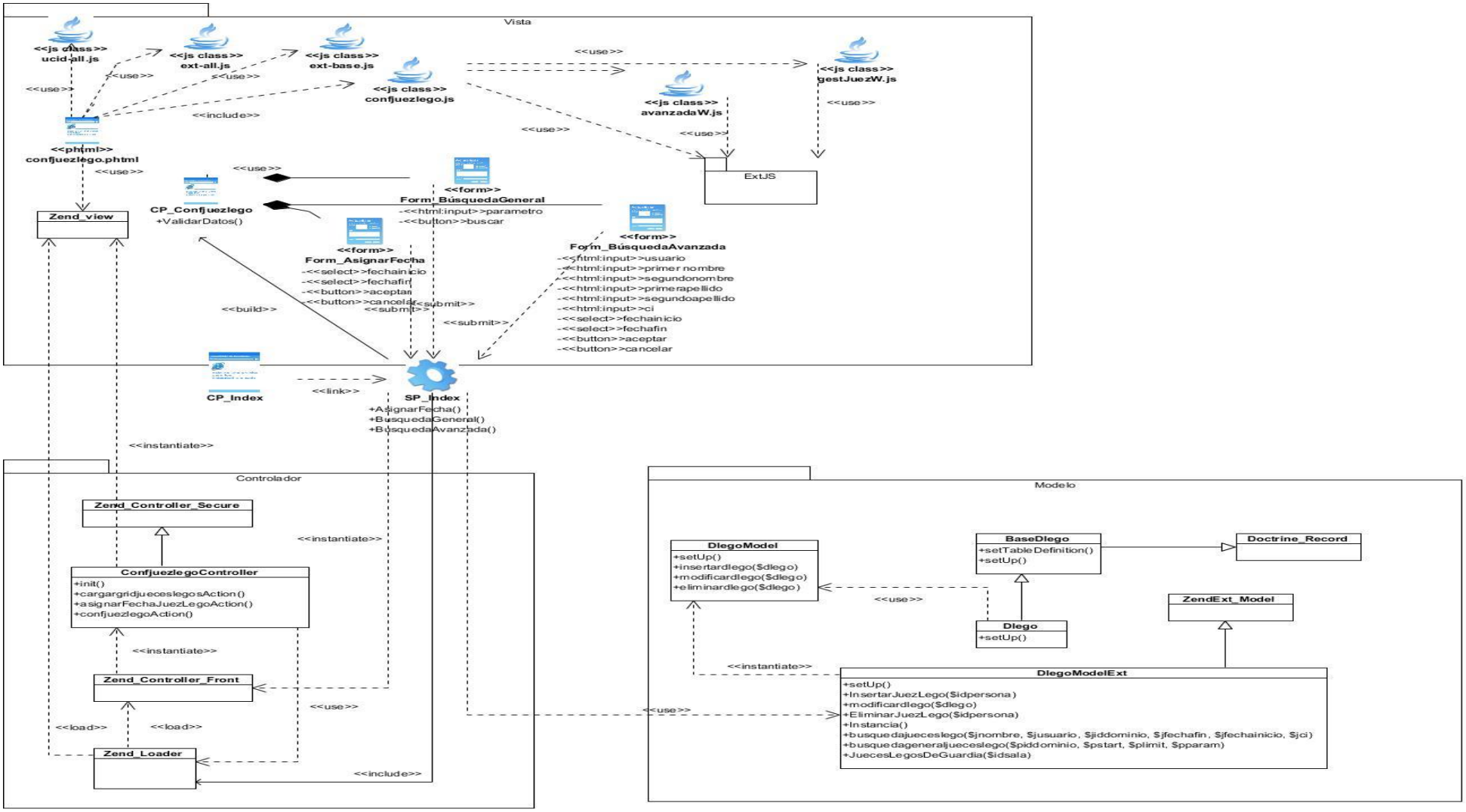


Figura 13: Diagrama de clases del diseño CU Asignar Fecha a Juez Lego

2.4.2. Realización de diagramas de interacción

El diagrama de interacción, representa la forma en como un cliente (actor) u objetos (clases) se comunican entre sí en petición a un evento. Esto implica recorrer toda la secuencia de llamadas, de donde se obtienen las responsabilidades claramente.

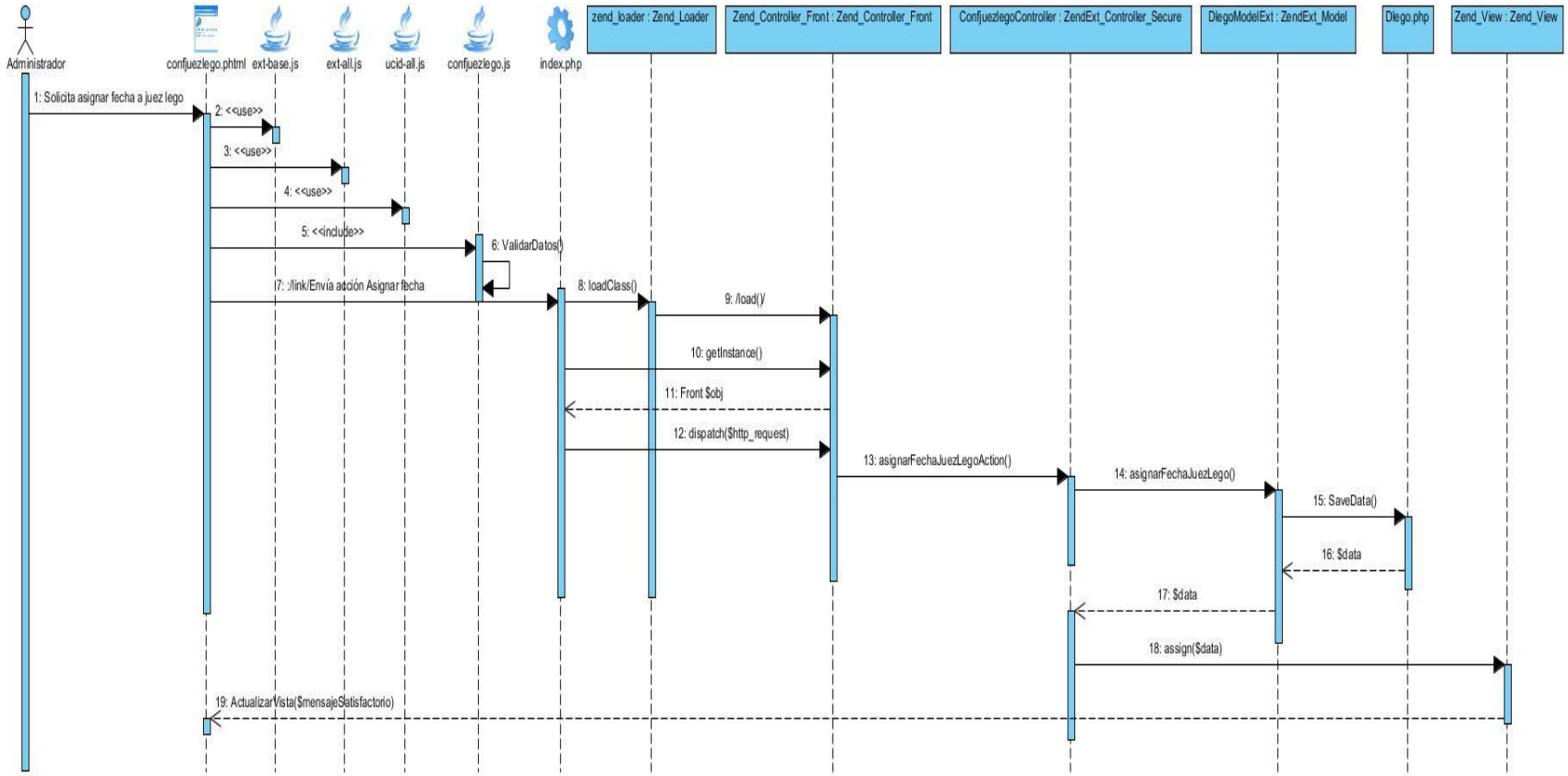


Figura 14: Diagrama de secuencia del CU Configurar Juez Lego, sección Asignar Fecha

3. Modelo de datos

El modelo de datos describe la representación lógica y física de los datos persistentes usados por la aplicación.

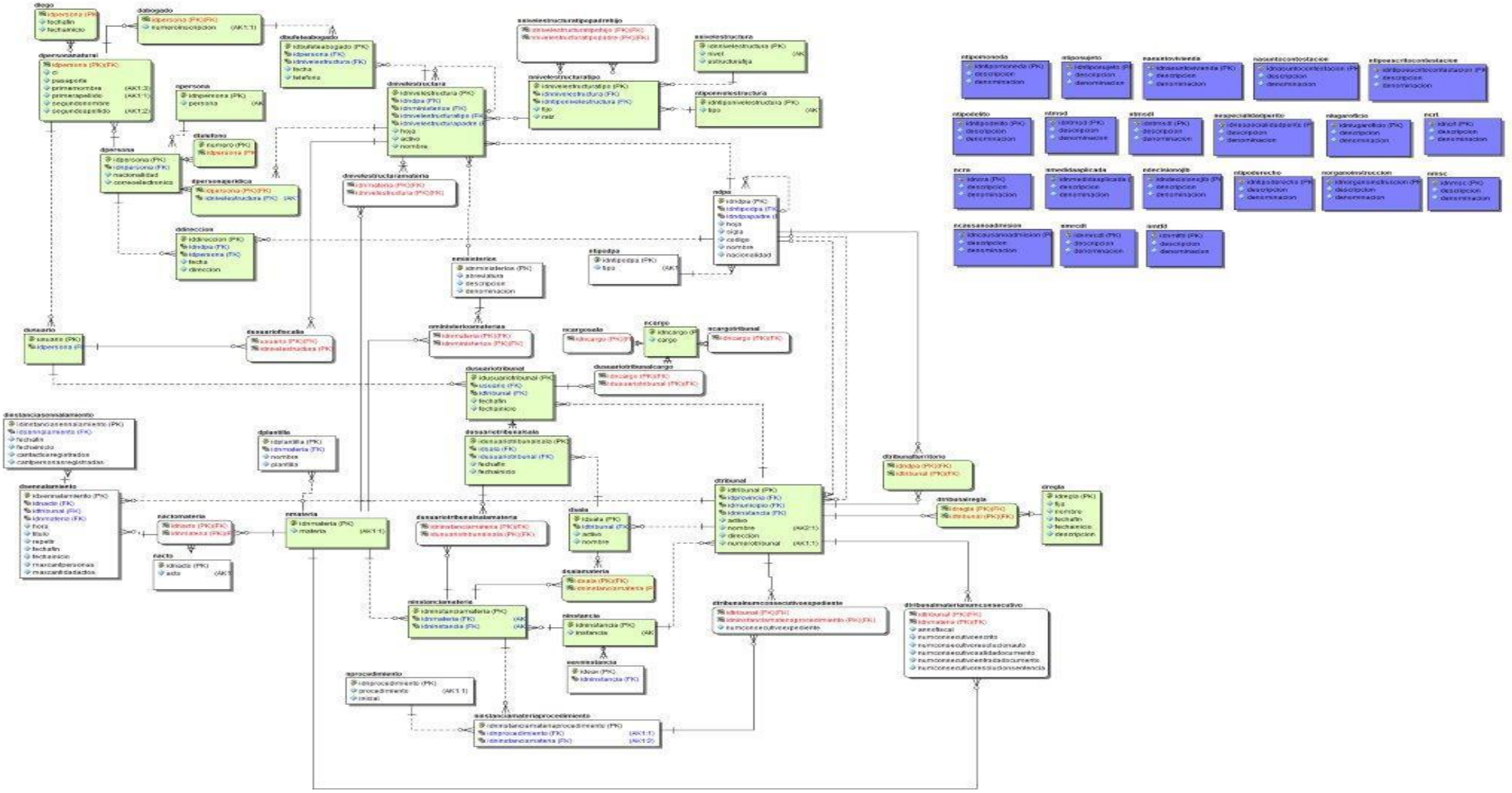


Figura 15: Modelo físico de datos esquema Administración y Gobierno

Diseño de la Base de Datos

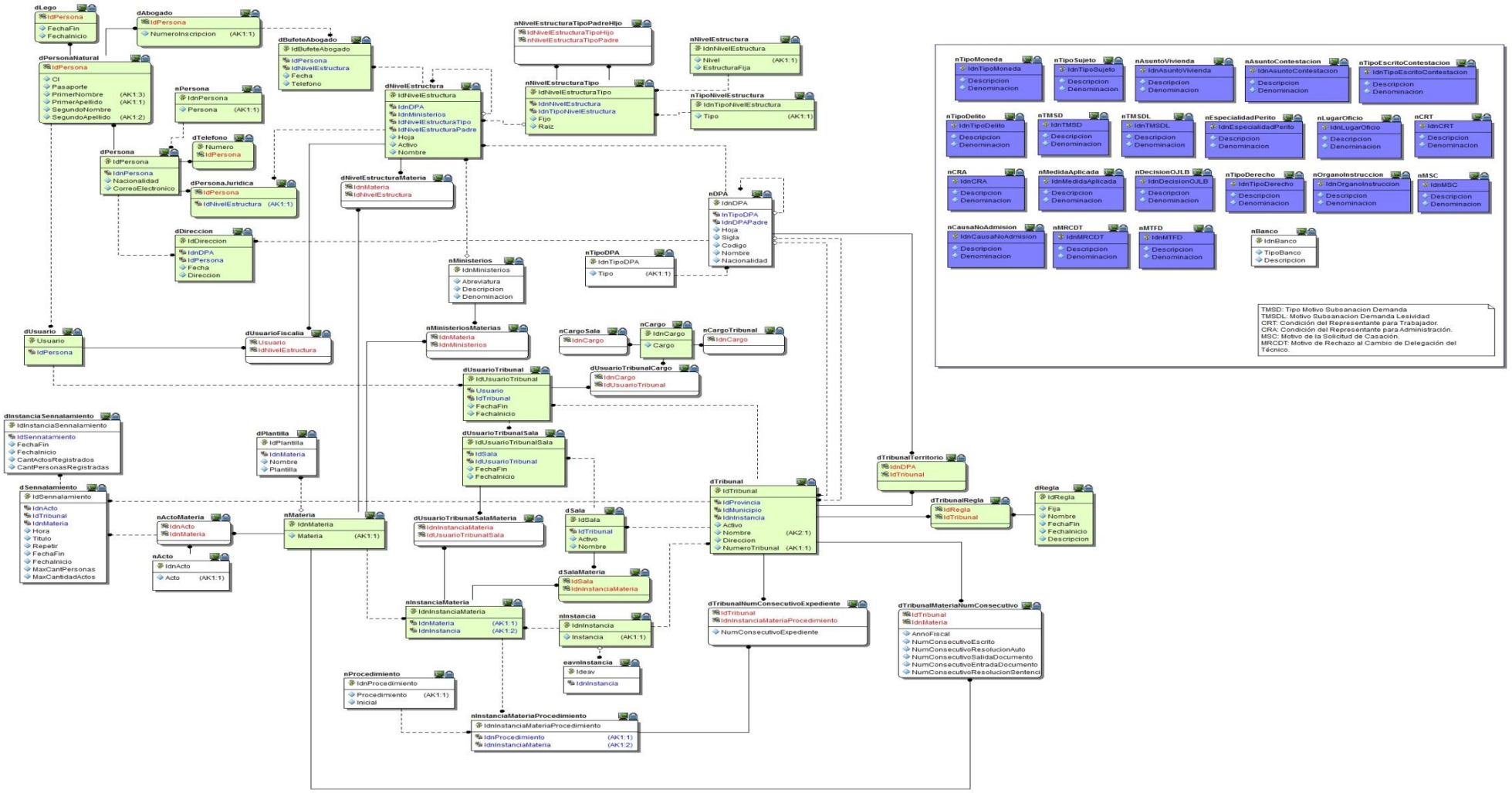


Figura 16: Modelo lógico de datos esquema Administración y Gobierno

4. Implementación

Este flujo de trabajo enmarca el comienzo de la fase de Construcción. Esta fase tiene como propósito dejar listo un producto de software en su versión operativa inicial (versión beta). A esta versión le concierne tener la calidad requerida para su uso y cumplir con los requerimientos del software determinados en el segundo capítulo.

Propósitos del flujo de implementación:

- ✓ Definir una organización del sistema en términos de Subsistemas de Implementación organizados en capas.
- ✓ Implementar los elementos de diseño en términos de “Elementos de Implementación” (ficheros fuente, binarios, ejecutables y otros).
- ✓ Probar los componentes desarrollados independientemente como unidades.
- ✓ Integrar los resultados producidos por desarrolladores independientes o equipos en un sistema ejecutable (Ivar Jacobson, G. B. y. J. R. ,2000).

4.1. Modelo de implementación

4.1.1. Diagrama de componentes

Los diagramas de componentes se representan como un grafo de componentes unidos por medio de relaciones de dependencia (compilación, ejecución), pudiendo mostrarse las interfaces que estos soporten. Cada componente representa una parte modular del sistema, despegable y reemplazable que encapsula implementación y un conjunto de interfaces y proporciona la realización de los mismos. Estos diagramas son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos. Dicho diagrama contiene los nodos que forman la topología hardware sobre la que se ejecuta el sistema y la distribución de las partes del sistema en ellos.

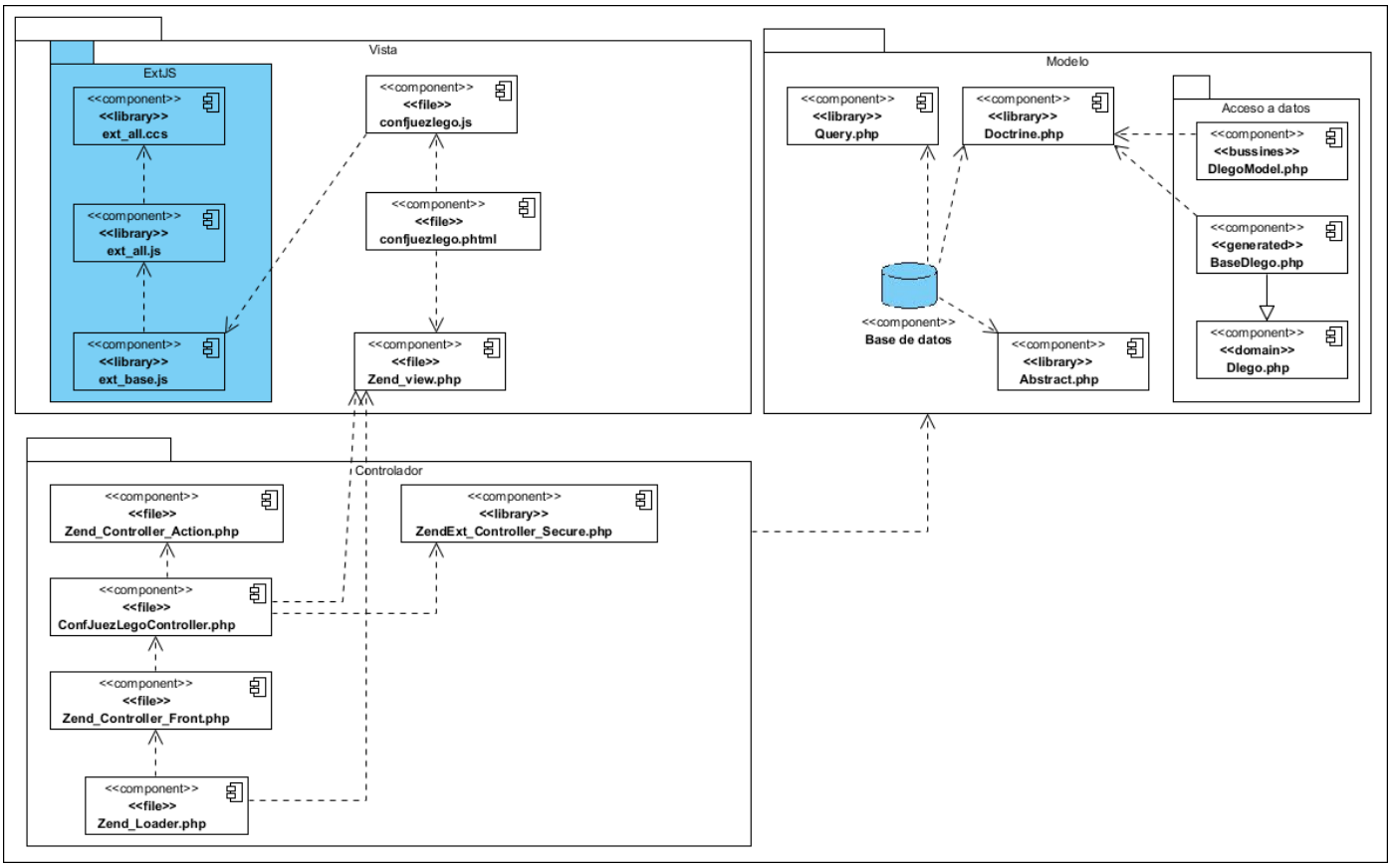


Figura 17 Diagrama de componentes del CU Configurar Juez Lego

4.1.2. Diagrama de despliegue

El Modelo de despliegue es utilizado para capturar los elementos de configuración del procesamiento y las conexiones entre esos elementos. También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos. El mismo está compuesto por:

- ✓ Nodos: Elementos de procesamiento con al menos un procesador, memoria, y posiblemente otros dispositivos.
- ✓ Dispositivos: Nodos estereotipados sin capacidad de procesamiento en el nivel de abstracción que se modela.
- ✓ Conectores: Expresan el tipo de conector o protocolo utilizado entre el resto de los elementos del modelo.(EcuRed, Marzo 2012)

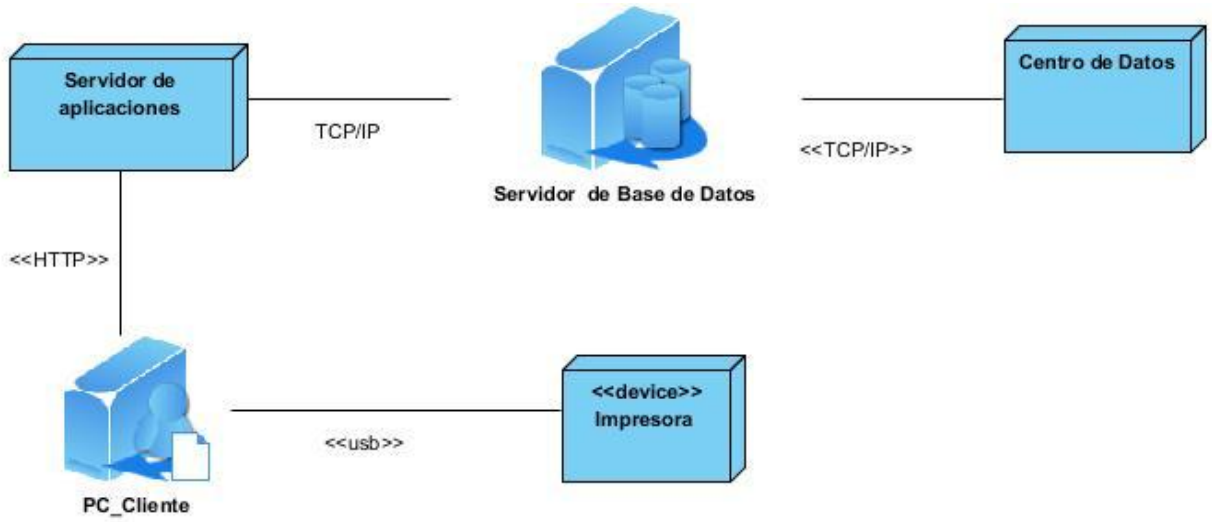


Figura 18: Diagrama de despliegue del SIT

4.1.3. Tratamiento de errores

En el sistema propuesto se evitan, minimizan y tratan los posibles errores, con el fin de garantizar la integridad y confiabilidad de la información. Para el tratamiento de errores se validan los datos que son introducidos por el usuario al sistema. Se valida que el formato de los datos sea el esperado y que no se omita información de importancia para la creación de tribunales, salas, usuarios. En estos casos se informa al usuario mediante un mensaje que ha realizado alguna operación incorrecta. Se muestran mensajes de advertencia para confirmar las acciones irreversibles, evitando la pérdida no deseada de datos. Cada usuario tendrá acceso restringido según su rol y la entidad en la que tenga permiso de trabajar con dicho rol por lo que se le permitirá solo ver las funcionalidades para las cuales obtenga permisos.

4.1.4. Seguridad

Para garantizar la seguridad de la información se creó el módulo de Administración y Gobierno el cual se encarga de definir roles de usuarios, que pueden ser: secretaria de gobierno, jefe de sala o sección, administrador, entre otros. El administrador es el encargado de los cambios y del buen funcionamiento del sistema, por tanto poseerá control total del mismo. El resto de los usuarios solo tendrá acceso a la información determinada por el administrador del sistema según el rol que posea.

Conclusiones parciales

En este capítulo se abordó lo referente a los flujos Diseño e Implementación. Se describió la arquitectura y los patrones que son utilizados en el desarrollo del sistema, se realizó el Modelo de datos y se entendió de una mejor forma el sistema realizado. Con la conclusión de este capítulo se obtuvo la implementación del módulo Administración y Gobierno del SIT. Fueron implementadas las clases obtenidas en el diseño. Se obtuvieron los Diagramas de componentes que describían el módulo a construir y el de despliegue que muestra su organización.

Capítulo 3: Validación de los resultados

Introducción

En este capítulo se evalúa el grado de calidad y fiabilidad de los resultados obtenidos con el desarrollo de este trabajo, teniendo por objeto la aplicación de algunas de las métricas empleadas internacionalmente para tal fin, específicamente las asociadas a las de la medición de la calidad del diseño e implementación de software como son los instrumentos de evaluación Tamaño Operacional de Clase y Relaciones entre Clases. Se realizarán además pruebas de Caja Blanca, de Caja Negra, pruebas Unitarias, pruebas de Aceptación y de Regresión con el objetivo de encontrar y corregir la mayor cantidad de errores del sistema en el menor tiempo posible.

1. Métricas de Software

Las métricas de software permiten medir de forma cuantitativa la calidad de los atributos internos del software lo que permite evaluar la calidad durante el desarrollo del sistema, están inspiradas en el estudio de la calidad del diseño orientado a objeto referenciadas por Pressman. Las métricas se centran en cuantificar tanto la complejidad, como la funcionalidad y eficiencia inmersa en el desarrollo de software. Inclina sus objetivos a mejorar la comprensión de la calidad del producto, a estimar la efectividad del proceso y mejorar la calidad del trabajo. Las métricas empleadas están diseñadas para evaluar los siguientes atributos de calidad:

- ✓ Responsabilidad, consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- ✓ Complejidad de implementación, grado de dificultad que tiene implementado un diseño de clases.
- ✓ Reutilización, grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- ✓ Acoplamiento, grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.
- ✓ Complejidad del mantenimiento, grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software.
- ✓ Cantidad de pruebas, número o grado de esfuerzo para realizar las pruebas de calidad del producto (componente, módulo, clase, conjunto de clases, etc.) diseñado.

Las métricas aplicadas al diseño para la evaluación de la solución propuesta son Tamaño Operacional de Clase (TOC) y Relaciones entre Clases (RC). El TOC está dado por el número de métodos asignados a

una clase y el instrumento de evaluación RC está dado por el número de relaciones de uso de una clase con otra.

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

Tabla 4 Atributos de calidad y modo en que afectan al TOC

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

Tabla 5 Atributos de calidad y modo en que afectan al instrumento RC

1.1. Resultado de la aplicación del instrumento de evaluación de la métrica Tamaño Operacional de Clase (TOC).

El instrumento de evaluación TOC fue aplicado a una muestra de 34 clases escogidas aleatoriamente del subsistema Administración y Gobierno, la Figura 19 muestra el porciento de la cantidad de procedimientos presentes en las clases agrupados en intervalos definidos, resultando 22 clases entre 1-5 procedimientos, 4 clases entre 6-10, 4 clases entre 11-15, 3 clases entre 21-25 y 1 clase entre 26-30 procedimientos.

Promedio de procedimientos por clase: 5

Responsabilidad: Baja: >= 5, 5 < Media < 10, Alta > 10

Complejidad: Baja: >= 5, 5 < Media < 10, Alta > 10

Reutilización: Alta: >= 5, 5 < Media < 10, Baja > 10

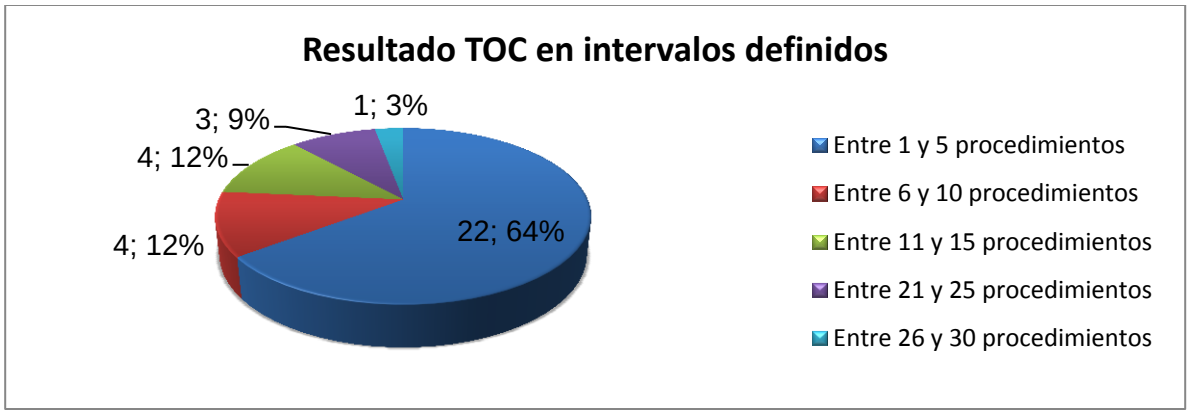


Figura 19 Representación en porciento agrupado en intervalos definidos de los resultados obtenidos tras aplicar el instrumento TOC.

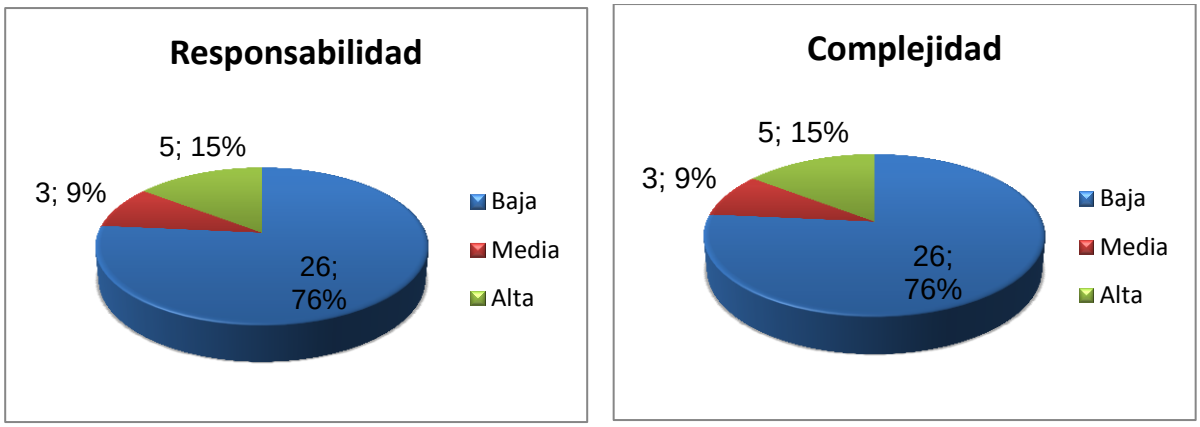


Figura 20: Representación de la incidencia de los resultados de la evaluación del instrumento TOC en los atributos: responsabilidad y complejidad.

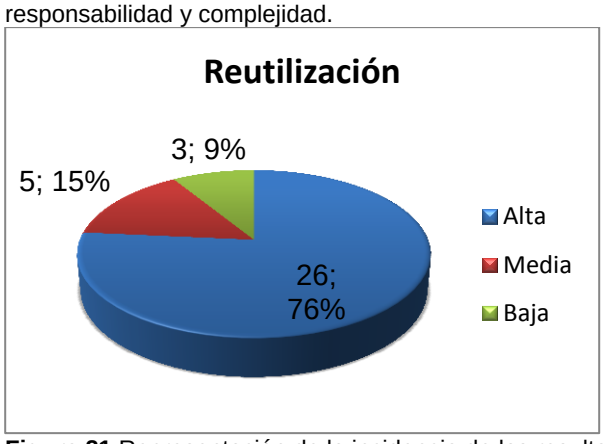


Figura 21 Representación de la incidencia de los resultados de la evaluación del instrumento TOC en el atributo reutilización.

Al hacer un análisis de los resultados obtenidos se concluye que la mayoría de las clases incluidas poseen de 1 a 5 procedimientos en su composición representando el 64% de la muestra. Debido a esto, el

resultado influye positivamente en los demás atributos de calidad puesto que ayuda a que el 76% de las clases posea una responsabilidad baja, es decir que no estén demasiado sobrecargadas, además puede observarse que el 76% del total de clases analizadas tiene una complejidad también baja y que se pueda realizar una explotación alta del atributo reutilización, estos resultados se ilustran en las figuras 20 y 21.

1.2. Resultado de la aplicación del instrumento de evaluación de la métrica Relaciones entre Clases (RC).

El instrumento de evaluación RC fue aplicado a una muestra de 12 clases del subsistema Administración y Gobierno seleccionadas aleatoriamente, la Figura 22 ilustra la cantidad de dependencias y el porciento que representan de la muestra seleccionada. Las figuras 22 y 23 muestran los resultados que arrojó la aplicación de la métrica en los atributos de calidad acoplamiento, complejidad de mantenimiento, reutilización y cantidad de pruebas.

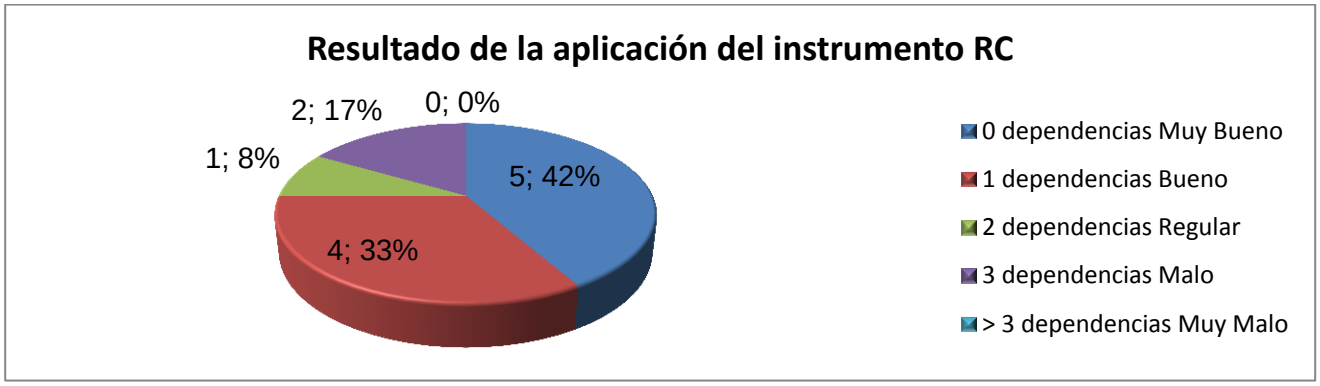
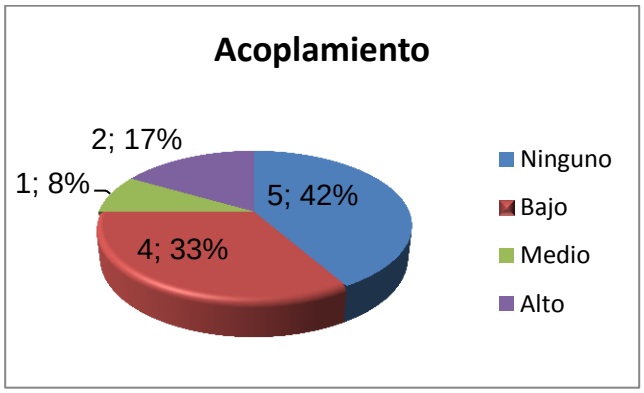


Figura 22: Representación en porciento de la aplicación del instrumento RC en intervalos definidos.



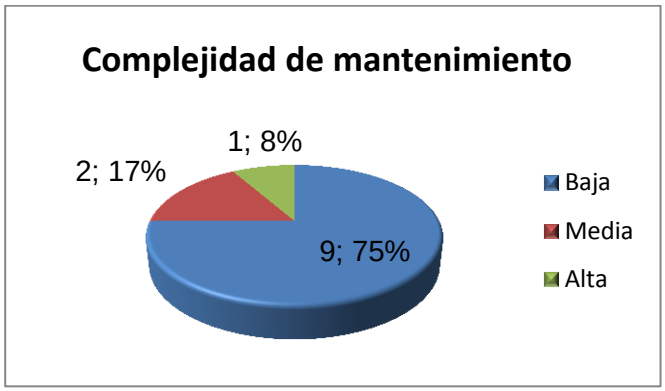


Figura 23: Representación de la incidencia de los resultados de la evaluación del instrumento RC en los atributos: acoplamiento y complejidad de mantenimiento.

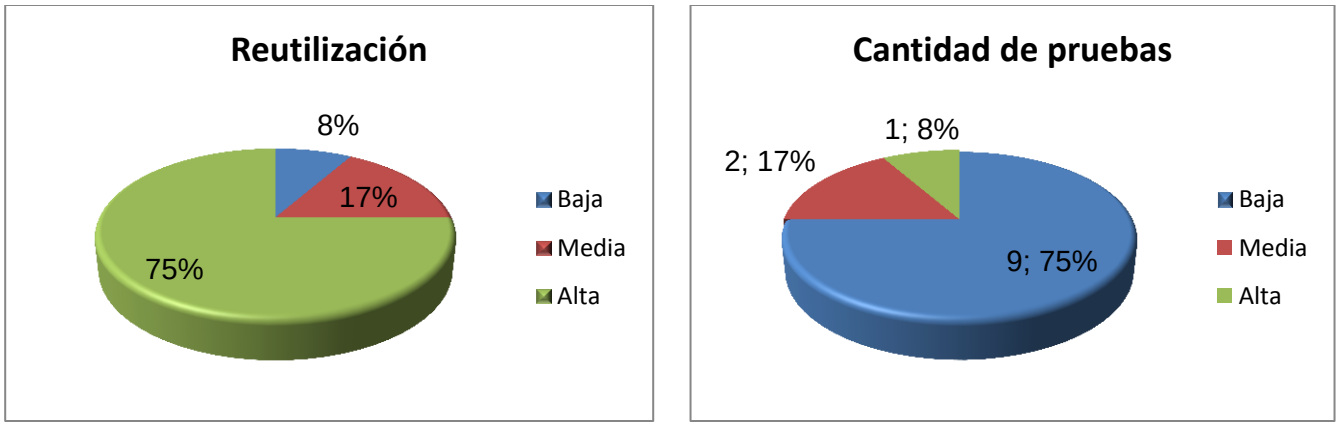


Figura 24: Representación de la incidencia de los resultados de la evaluación del instrumento RC en los atributos: cantidad de pruebas y reutilización.

Al hacer un análisis de los resultados obtenidos a través de la métrica de software RC aplicada se resuelve que el 42% de las clases incluidas no poseen acoplamiento alguno lo cual ayuda significativamente en el sentido de que al ser afectada una clase este cambio no repercutirá en las demás. El porciento de reutilización de clases (75%) es alto lo cual fomenta y potencia el uso de las mismas en otras clases, objetos, etc., en caso de ser necesario, además la complejidad de mantenimiento es un 75% baja facilitando que no resulte nada difícil el mantenimiento de las mismas (por ejemplo la optimización de métodos) y por último el atributo relacionado con la cantidad de pruebas es 75% bajo lo cual señala que a las clases se les puede realizar un bajo número de pruebas de poca complejidad y de bajos costes.

2. Pruebas

Los métodos de Prueba de Software tienen como objetivo diseñar pruebas que descubran diferentes tipos de errores en el menor tiempo y esfuerzo posible. Existen dos métodos de prueba:

Pruebas de Caja Negra: El sistema de pruebas de Caja Negra no considera la codificación dentro de sus parámetros a evaluar, es decir, no están basadas en el conocimiento del diseño interno del programa. Estas pruebas se enfocan en los requerimientos establecidos y en la funcionalidad del sistema. Según (Pressman, 2002) las pruebas de Caja Negra intentan encontrar los siguientes errores:

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores de interfaz.
- ✓ Errores de estructura de datos o accesos a base de datos externas.
- ✓ Errores de rendimiento.
- ✓ Errores de inicialización y de terminación.

Algunos de los métodos empleados en las pruebas de Caja Negra son los siguientes:

- ✓ Métodos de prueba basados en grafos.
- ✓ Partición equivalente.
- ✓ Análisis de valores límite.
- ✓ Prueba de la tabla ortogonal.
- ✓ Adivinando el error.

Pruebas de Caja Blanca: Al contrario de las de Caja Negra, estas se basan en el conocimiento de la lógica interna del código del sistema. Las pruebas completan los distintos caminos que se pueden generar gracias a las estructuras condicionales, a los distintos estados del mismo, entre otros factores.

Para la realización de estas pruebas existen diferentes métodos los cuales son:

- ✓ Prueba del camino básico.
- ✓ Prueba de condición.
- ✓ Prueba del flujo de datos.
- ✓ Prueba de bucles.

Como se puede observar en la **Figura 25** las pruebas de Caja Blanca necesitan conocer los detalles procedimentales del código, mientras que las de Caja Negra únicamente necesitan saber el objetivo o funcionalidad que el código ha de proporcionar.

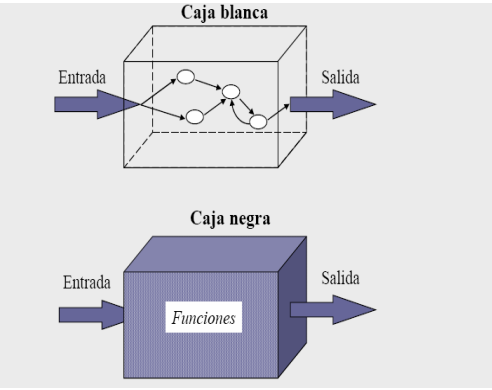


Figura 25: Representación de las pruebas de Caja Blanca y Caja Negra

2.1. Pruebas de Caja Blanca o Estructurales

Este tipo de técnicas es conocido también como técnicas de Caja Transparente o de Cristal. Este método se centra en cómo diseñar los casos de prueba atendiendo al comportamiento interno y la estructura del programa. Se examina así la lógica interna del programa sin considerar los aspectos de rendimiento. El

objetivo de la técnica es diseñar casos de prueba para que se ejecuten, al menos una vez, todas las sentencias del programa, y todas las condiciones tanto en su vertiente verdadera como falsa. Como se ha indicado ya, puede ser impracticable realizar una prueba exhaustiva de todos los caminos de un programa. Por ello se han definido distintos criterios de cobertura lógica, que permiten decidir qué sentencias o caminos se deben examinar con los casos de prueba. Estos criterios son:

- ✓ Cobertura de Sentencias: Se escriben casos de prueba suficientes para que cada sentencia en el programa se ejecute, al menos, una vez.
- ✓ Cobertura de Decisión: Se escriben casos de prueba suficientes para que cada decisión en el programa se ejecute una vez con resultado verdadero y otra con el falso.
- ✓ Cobertura de Condiciones: Se escriben casos de prueba suficientes para que cada condición en una decisión tenga una vez resultado verdadero y otra falso.
- ✓ Cobertura Decisión/Condición: Se escriben casos de prueba suficientes para que cada condición en una decisión tome todas las posibles salidas, al menos una vez, y cada decisión tome todas las posibles salidas, al menos una vez.
- ✓ Cobertura de Condición Múltiple: Se escriben casos de prueba suficientes para que todas las combinaciones posibles de resultados de cada condición se invoquen al menos una vez.
- ✓ Cobertura de Caminos: Se escriben casos de prueba suficientes para que se ejecuten todos los caminos de un programa. Entendiendo camino como una secuencia de sentencias encadenadas desde la entrada del programa hasta su salida.

Este último criterio, Cobertura de Caminos, es el que se va a utilizar.

2.1.1. Prueba del Camino Básico

La Prueba del Camino Básico es una técnica que permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto de caminos de ejecución. Estos casos de prueba son los que garantizan que se ejecute al menos una vez cada sentencia del programa. Para ilustrar esta técnica se hace uso de los denominados “grafos de flujo”, aunque no es necesario, constituyen una herramienta muy útil sobre todo cuando la lógica de la estructura de control de un módulo es muy compleja permitiendo trazar los caminos del programa de una forma más fácil. Los elementos de un grafo de flujo son los siguientes:

Nodos: Representan una o más sentencias procedimentales. Se denominan “nodos predicados” aquellos que contienen una condición (if - else) y se caracterizan porque dos o más aristas surgen de él.

Aristas: Representan el flujo de control. Deben terminar en un nodo aunque el nodo no constituya ninguna sentencia procedimental.

Regiones: Son las áreas delimitadas por aristas y nodos. El área exterior del grafo se cuenta como una región más.

Los pasos a realizar para aplicar esta técnica son:

1. Representar el programa en un grafo de flujo.
2. Calcular la complejidad ciclomática.
3. Determinar el conjunto básico de caminos independientes.
4. Derivar los casos de prueba que fuerzan la ejecución de cada camino.

Para realizar la prueba del Camino Básico al CU Configurar Juez Lego se eligió la clase ConfJuezLegoController y dentro de ésta específicamente al método AsignarFechaJuezLegoAction() que se muestra en la **Figura 26**, el mismo tiene la tarea de asignarle un rango de fechas determinado a un juez lego seleccionado previamente.

```
function AsignarFechaJuezLegoAction() {
    $fechaInicio = strtotime($this->_request->getPost('fi'));
    $fechaFin = strtotime($this->_request->getPost('ff'));
    $idpersona = $this->_request->getPost('idpersona');
    if (!$fechaInicio || !$fechaFin) {
        echo ("{success: false,'mensaje': 'Debe llenar Fecha de Inicio y  Fecha de Fin.'}");
        return;
    }
    if ($fechaInicio > $fechaFin) {
        echo ("{success: false,'mensaje': 'La fecha de inicio no puede ser mayor que la fecha de fin.'}");
        return;
    }
    $jl = Doctrine::getTable('Dlego')->find($idpersona);
    $jl->fechafin = $this->_request->getPost('ff');
    $jl->fechainicio = $this->_request->getPost('fi');
    try {
        $jl->save();
        echo( "{success: true,mensaje: 'La fecha fue asignada satisfactoriamente.'}");
        return;
    } catch (Doctrine_Exception $exc) {
        if (DEBUG_ERP)
            throw new Exception('La fecha no pudo ser asignada satisfactoriamente');
        return false;
    }
}
```

Figura 26: Método AsignarFechaJuezLego() de la clase ConfJuezLego

La **Figura 27** muestra el grafo de flujo para el método AsignarFechaJuezLego() compuesto por 6 nodos, 2 de ellos nodos predicados (1 y 3), 3 regiones (R1,R2 y R3) y 7 aristas.

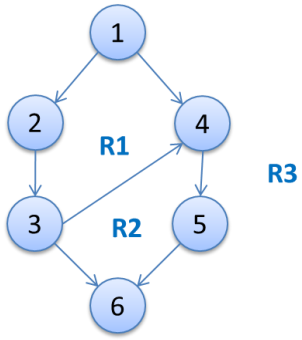


Figura 27: Grafo de flujo correspondiente al método AsignarFechaJuezLego()

A continuación se debe hallar la complejidad ciclomática del grafo de flujo, para ser calculada existen varias fórmulas entre ellas está: V (G1)=Aristas – Nodos +2, V (G2)= Nodos Predicado + 1, también se calcula contando el número de regiones que lo conforman.

V (G1)= 7 -6 + 2 = 3, V (G2)= 2 + 1 = 3, V (G3) = Número de regiones = 3

El cálculo efectuado mediante las fórmulas antes presentadas muestran una complejidad ciclomática de valor 3, por lo que existen 3 posibles caminos por donde el flujo puede circular, este valor representa el número mínimo de casos de pruebas para el procedimiento tratado. El siguiente paso es definir cuál es el conjunto básico de caminos lo que significa las posibles situaciones por las que puede pasar el programa. Un camino es nuevo cuando incluye al menos una arista que no ha sido recorrida en los caminos anteriores.

Caminos independientes **#1**: 1-2-3-6, **#2**: 1-2-3-4-5-6, **#3**: 1-2-4-5-6

Finalmente se deben forzar los casos de pruebas para que sean recorridos todos y cada uno de los caminos del programas al menos una vez. Para definir los casos de prueba se debe tener en cuenta lo siguiente:

- ✓ Descripción: Se describe el caso de prueba y de forma general se tratan los aspectos fundamentales de los datos de entrada.
- ✓ Condición de ejecución: Se especifica cada parámetro para que cumpla una condición deseada y así ver el funcionamiento del procedimiento.
- ✓ Entrada: Se muestran los parámetros que serán la entrada al procedimiento.
- ✓ Resultados esperados: Se expone el resultado esperado que debe devolver el procedimiento después de efectuado el caso de prueba.

N. C.[15]	*Descripción*	*Condición de ejecución*	*Entrada*	*Resultados esperados*	*Resultado obtenido*
1	*Se debe seleccionar uno de los jueces legos que aparecen en el grid y posteriormente se le asigna un rango de fecha determinado.*	*La fecha de inicio se envía sin datos y la fecha de fin con sus datos correspondientes.*	*fecha-inicio=null fecha-fin=6/07/2011*	*Se lanza un mensaje de error informando al usuario que la fecha-inicio y/o fecha-fin no pueden entrarse sin datos.*	*El resultado obtenido fue correcto.*
2	*Se debe seleccionar uno de los jueces legos que aparecen en el grid y posteriormente se le asigna un rango de fecha determinado.*	*La fecha de fin se envía sin datos y la fecha de inicio con sus datos correspondientes.*	*fecha-inicio=10/08/2010 fecha-fin=null*	*Se lanza un mensaje de error informando al usuario que los campos fecha-inicio y/o fecha-fin no pueden estar vacíos.*	*El resultado obtenido fue correcto.*
3	*Se debe seleccionar uno de los jueces legos que aparecen en el grid y posteriormente se le asigna un rango de fecha determinado.*	*La fecha de fin se envía con datos y la fecha de inicio con sus datos correspondientes pero la fecha de inicio es mayor que la fecha de fin.*	*fecha-inicio=10/10/2011 fecha-fin=10/02/2010*	*Se muestra un mensaje de error al usuario informándole que la fecha de inicio debe ser menor que la fecha de fin.*	*El resultado obtenido fue correcto.*

[15]Número de camino

4	*Se debe seleccionar uno de los jueces legos que aparecen en el grid y posteriormente se le asigna un rango de fecha determinado.*	*Las fechas de inicio y fecha de fin se envían con sus datos correctos.*	*fecha-inicio=5/10/2010 fecha-fin=8/10/2010*	*Se ejecuta el método satisfactoriamente lanzando un mensaje de que se asignó correctamente la fecha.*	*El resultado obtenido fue correcto.*

Tabla 6. Casos de prueba para pruebas estructurales del método AsignarFechaJuezLego()

2.2. Pruebas de Caja Negra o Funcionales

Como técnica de prueba se utilizan las pruebas de Caja Negra que se llevan a cabo sobre la interfaz del software. Su objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene. Son diseñadas para validar los requisitos funcionales sin fijarse en el funcionamiento interno del programa. Las mismas se centran en el ámbito de información del programa, de forma que se proporcione una cobertura completa de prueba.

Criterios mínimos que guiarán el resultado de las pruebas:

- ✓ Valores fáciles: El programa se depurará con datos fáciles de comprobar.
- ✓ Valores típicos realistas: Se ensayará un programa con datos seleccionados para que representen cómo se aplicará. Estos datos han de ser suficientemente sencillos, de modo que los resultados sean verificables de forma manual, valores futuros candidatos a ser usados en la aplicación.
- ✓ Valores extremos: Muchos programas cometen errores en los límites de sus rangos de aplicaciones.
- ✓ Valores ilegales: Se entrarán datos cuyos valores no corresponden con los requerimientos por el sistema, su reacción inmediata habrá de ser por lo menos un mensaje de error adecuado para el usuario.

El criterio utilizado es el de particiones equivalentes, principalmente se centra en la definición de casos de prueba que descubre las clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. El diseño de casos de prueba para la partición equivalente se basa en la evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Regularmente, una condición de

entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica. Estas pruebas detectaron un total de 109 no conformidades, valor que representa aproximadamente el 58.23% del total de no conformidades detectadas (187) y fueron corregidas al término de la 4ta iteración de pruebas.

2.2.1. Casos de prueba

Un caso de prueba especifica una forma de probar el sistema, incluyendo la entrada o resultado con la que se ha de probar y las condiciones bajo las que ha de probarse. (Jacobson, 2000).A continuación se podrá apreciar un caso de prueba de uno de los casos de usos más significativos, CU Configurar Juez Lego.

Nombre de la sección	***Escenarios de la sección***	***Descripción de la funcionalidad***	***Flujo Central***
SC1: Asignar fecha.	*EC1.1: Asignar fecha satisfactoriamente.*	*Se asigna una fecha de inicio y fin a un juez lego, esta es la fecha en que trabajarán en un tribunal.*	*1- El sistema muestra una interfaz solicitando los siguientes datos:* ✓ *Fecha de Inicio* ✓ *Fecha Fin* *2- Se introducen los datos correspondientes y se selecciona la opción "Aceptar".* *3- El sistema valida que los datos introducidos son correctos y que no hay campos obligatorios vacíos. Terminando así el caso de uso.*
	EC1.2: Asignar fecha dejando campos en blanco.	*No se asigna una fecha de inicio o fin a un juez lego.*	*1- El sistema muestra una interfaz solicitando los siguientes datos:* ✓ *Fecha de Inicio* ✓ *Fecha Fin* *2- Solo se introduce la fecha de inicio y selecciona la opción "Aceptar".* *3- El sistema valida que los datos introducidos son correctos y al haber*

			campos obligatorios vacíos muestra un mensaje alertando que el campo vacío es obligatorio. Terminando así el caso de uso.
	EC1.3: Asignar fecha insertando valores incorrectos (fecha inicio mayor que fecha de fin).	*No se asigna una fecha de inicio o fin a un juez lego.*	*1-El sistema muestra una interfaz solicitando los siguientes datos:* ✓ *Fecha de Inicio* ✓ *Fecha Fin* *2-Se introduce un valor de fecha de inicio mayor que el de fecha de fin y selecciona la opción "Aceptar".* *3-El sistema valida que los datos introducidos son correctos y al tratar de insertar un rango de fecha incorrecto muestra un mensaje alertando que el fecha de inicio debe ser menor que la fecha de fin. Terminando así el caso de uso.*
	EC1.4: Operación cancelada.	*No se asigna una fecha de inicio o fin a un juez lego. Se cancela la operación y vuelve a la interfaz previa.*	*1- Se solicita cancelar la operación.* *2- Cancela la Operación y vuelve a la interfaz previa. Terminando así el caso de uso.*
SC2: Búsqueda general.	*EC 2.1: Búsqueda general satisfactoriamente.*	*Se busca un juez lego por los datos solicitados.*	*1. El sistema muestra una interfaz con el siguiente campo: Búsqueda general. Para realizar la búsqueda de un juez lego por cualquiera de sus datos, menos el rango de fecha que tiene asociado.* *2. Introduce el dato deseado y selecciona*

			la opción "Buscar". *3. El sistema busca el o los jueces lego que coincida con el campo introducido, terminando así el caso de uso.*
	EC 2.1: Búsqueda General sin pasar criterio de búsqueda.	*No trae resultados la búsqueda.*	*1. El sistema muestra una interfaz con el siguiente campo: Búsqueda general. Para realizar la búsqueda de un juez lego por cualquiera de sus datos, menos el rango de fecha que tiene asociado.* *2. No se introduce ningún dato y se selecciona la opción "Buscar".* *El sistema no muestra resultados de búsqueda y recarga la interfaz, terminando así el caso de uso.*
SC 3: Búsqueda avanzada.	*EC 3.1: Búsqueda avanzada satisfactoriamente.*	*Se busca un juez lego por cualquier dato.*	*1-El sistema muestra la interfaz para buscar un juez lego por los siguientes criterios: Primer nombre, Segundo nombre, Primer apellido, Segundo apellido, Usuario, Carné de identidad, Fecha de Inicio, Fecha Fin.* *2-Se introducen los datos según los criterios deseados.* *3-Solicita aceptar la operación.* *4-El sistema valida que los datos introducidos son correctos.* *5-El sistema busca el juez lego, terminando así el caso de uso.*
	EC 3.2: Búsqueda avanzada sin pasar el criterio de	*No se busca un juez lego, se muestra un mensaje de error*	*1-El sistema muestra la interfaz para buscar un juez lego por los siguientes criterios: Primer nombre, Segundo*

	búsqueda.	*indicando que se debe llenar al menos un criterio de búsqueda.*	*nombre, Primer apellido, Segundo apellido, Usuario, Carné de identidad, Fecha de Inicio, Fecha Fin.* *2-No se introducen criterios de búsqueda.* *3-Solicita aceptar la operación.* *4-El sistema valida que los datos introducidos son correctos.* *5-El sistema busca el juez lego, terminando así el caso de uso.*
	EC3.3: Operación cancelada.	*No se busca un juez lego. Se cancela la operación y vuelve a la interfaz previa.*	*1- Se solicita cancelar la operación.* *2- Cancela la Operación y vuelve a la interfaz previa. Terminando así el caso de uso.*

Tabla 7 Casos de prueba CU Configurar Juez Lego

2.3. Pruebas de Aceptación

El objetivo de las pruebas de Aceptación es constatar el grado de satisfacción del cliente con respecto a la aplicación desarrollada, ya que es precisamente el cliente quien define las pruebas de Aceptación preparadas por el equipo de desarrollo. Al realizar las pruebas de aceptación, el producto está listo para implantarse en el entorno del cliente. El usuario debe ser el que realice las pruebas, ayudado por personas del equipo de pruebas, siendo deseable, que sea el mismo usuario quien aporte los casos de prueba. Estas pruebas se caracterizan por:

- ✓ Participación activa del usuario, que debe ejecutar los casos de prueba ayudado por miembros del equipo de pruebas.
- ✓ Están enfocadas a probar los requisitos de usuario, o mejor dicho a demostrar que no se cumplen los requisitos, los criterios de aceptación o el contrato. Si no se consigue demostrar esto el cliente deberá aceptar el producto
- ✓ Corresponden a la fase final del proceso de desarrollo de software.

Las pruebas de Aceptación se realizaron por el personal que manejará el sistema, en este caso por los jueces y la secretaria de gobierno de los tribunales supremo y provincial de La Habana. A lo largo de la

implementación se fueron realizando dichas pruebas con los clientes, quiénes expresaban su conformidad con la funcionalidad mostrada y detectaban errores, también realizaban solicitudes de cambios. Para la generación de casos de prueba de aceptación se utilizaron técnicas de Caja Negra, específicamente el criterio de particiones de equivalencia. En total luego de las 3 primeras iteraciones de pruebas fueron detectadas 53 no conformidades en las pruebas de Aceptación, lo que representa aproximadamente el 28.34 % del total de no conformidades (187) detectadas.

2.4 Pruebas de Regresión

Una vez corregidas las no conformidades detectadas en las pruebas de Caja Negra y de Aceptación se realizaron pruebas de Regresión. La regresión consiste en la repetición selectiva de pruebas para detectar fallos introducidos durante la modificación de un sistema o componente de un sistema. Se efectuaron para comprobar que los cambios no originaron efectos adversos no intencionados o que se siguen cumpliendo los requisitos especificados. Este tipo de pruebas ha de realizarse, tanto durante el desarrollo como cuando se produzcan cambios. Arrojaron un total de 20 no conformidades, cifra que representa aproximadamente el 10.7% del total de no conformidades detectadas (187) y fueron corregidas al término de la 4ta iteración de pruebas.

2.5 Pruebas Unitarias

Las pruebas Unitarias son un procedimiento para validar que una porción del código del sistema funciona apropiadamente de manera aislada. Cuando se está programando orientado a objetos la menor de estas unidades es siempre una clase. Rod Johnson[16] define las pruebas Unitarias como el nivel más fino de granularidad en las pruebas, que verifican una simple unidad de funcionalidad y deben probar que cada método de una clase satisface su contrato documentado. (JOHNSON, ROD 2003)

Durante la realización del proceso de pruebas se decidió aplicar pruebas Unitarias a los métodos que presentan una complejidad considerable dentro del subsistema. A continuación se presenta un ejemplo de la aplicación de dichas pruebas al método AsignarFechaJuezLego($fechaInicio,$fechaFin, $idPersona) de la clase ConfJuezLegoController. Para la ejecución de estas pruebas se utilizó el siguiente caso de prueba:

[16] Autor de los libros "Expert One-on-One Design J2EE y el Desarrollo" y "J2EE sin EJB", una autoridad mundial en Java y J2EE, y un empresario. Fundador y CEO de SpringSource. Rod tiene una licenciatura con honores en Ciencias de la Computación, Matemáticas y musicología, así como un doctorado de la Universidad de Sydney.

#Jd[17]	$fechaInicio	$fechaFin	$idPersona	Resultado esperado	Resultado obtenido
1	null	null	null	Mensaje: "Debe llenar Fecha de Inicio y Fecha de Fin"	Mensaje: "Debe llenar Fecha de Inicio y Fecha de Fin"
2	'2012-06-01'	'2012-05-29'	1	Mensaje: "La fecha de inicio no puede ser mayor que la fecha de fin."	Mensaje: "La fecha de inicio no puede ser mayor que la fecha de fin."
3	'2012-05-29'	'2012-06-01'	null	Mensaje: "El id de persona no puede ser nulo."	Mensaje: "El id de persona no puede ser nulo."
4	'2012-05-29'	'2012-06-01'	1	Mensaje: "Fecha asignada satisfactoriamente"	Mensaje: "Fecha asignada satisfactoriamente"

Tabla 8 Caso de prueba unitaria aplicado al método AsignarFechaJuezLego de la clase ConfJuezLegoController.

Las figuras 28 y 29 ilustran la aplicación de las pruebas unitarias al caso de prueba anteriormente descrito con los juegos de datos 2 y 3.

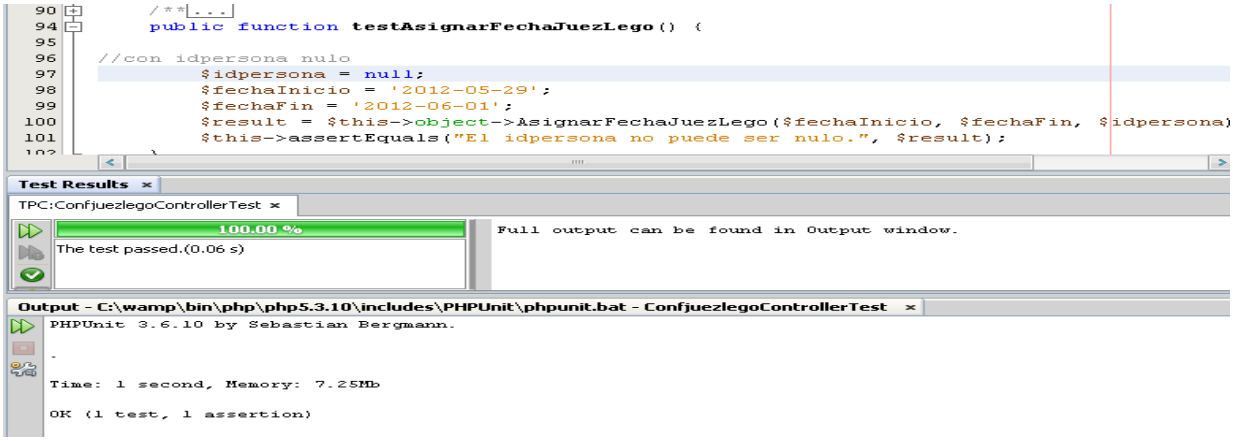


Figura 28 Resultado de la aplicación del juego de datos #3 del caso de prueba anterior.

[17] Juego de datos

Figura 29 Resultado de la aplicación del juego de datos #2 del caso de prueba anterior.

Como resultado de la aplicación de los casos de pruebas diseñados se detectaron un total de cinco errores, en su totalidad errores de validación al aplicarle condiciones límites a los métodos. La Figura 30 ilustra el resultado de la aplicación de las pruebas donde se puede observar la cantidad de errores por métodos y como fueron corregidos tras una segunda iteración de pruebas.

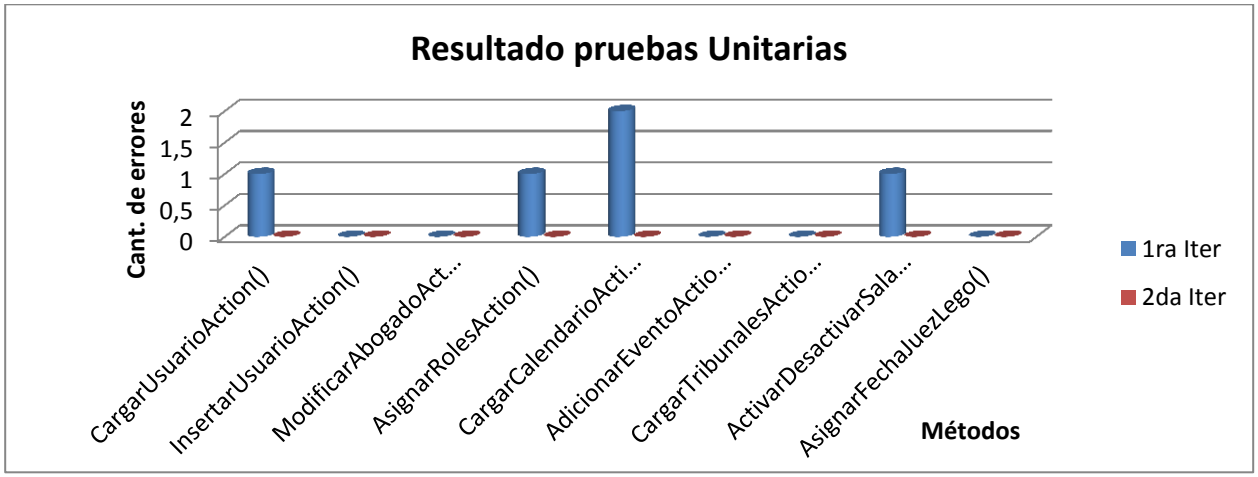


Figura 30 Resultado de la aplicación de las pruebas Unitarias

2.6 Resultados de las pruebas

Luego de realizadas cuatro iteraciones de pruebas, fueron detectadas un total de 187 no conformidades, de ellas 161 en casos de uso significativos y 26 en los nomencladores, la Figura 31 ilustra este resultado. La Figura 32 muestra las no conformidades detectadas por CU en cada iteración, todas fueron resueltas

en su totalidad al término de la cuarta iteración, se puede concluir que las pruebas fueron exitosas, el sistema está apto para su utilización y cumple con todas las funcionalidades para las que fue concebido.

Figura 31: Resultados generales de las pruebas.

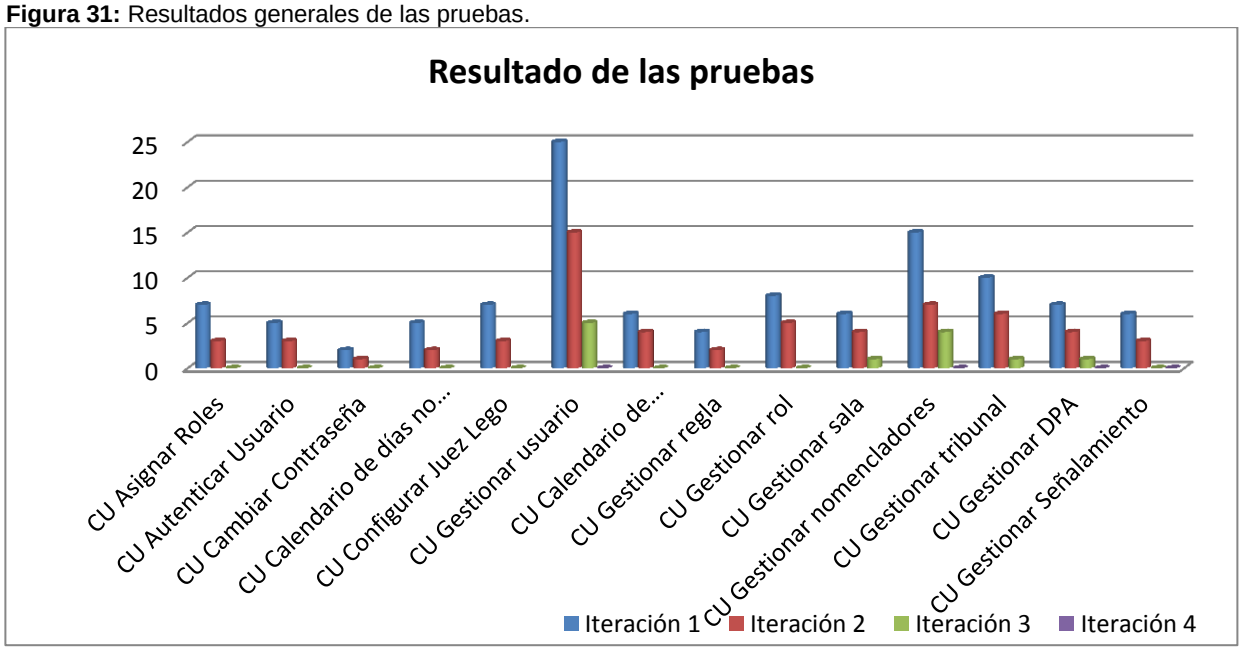


Figura 32: Gráfico que ilustra el resultado de las pruebas al sistema

Conclusiones parciales

Con la conclusión de este capítulo finaliza la implementación del Módulo Administración y Gobierno para el Sistema de Informatización de Tribunales junto con su fase de pruebas de aseguramiento de la calidad del software. Fueron implementadas las clases obtenidas en el diseño. Se obtuvieron los diagramas de componentes que describían el módulo a construir y el de despliegue que muestra su organización. Fueron realizadas las pruebas al sistema con resultados satisfactorios que demuestran que la aplicación cuenta con las características y funcionalidades para las que fue concebido.

Conclusiones generales

Luego de realizarse la investigación, se concluye que:

- ✓ Del estudio de los procedimientos realizados en los TPC para el vencimiento de términos, señalamiento de actos judiciales y búsqueda de información, se obtuvieron como resultado los problemas que afronta el tribunal en la actualidad.
- ✓ Del estudio de los procedimientos de creación de usuarios, roles, asignación de permisos, nomencladores, se obtuvo como resultado la necesidad de crear un módulo para la administración y configuración de dichos procesos sumándoles el vencimiento de términos y el señalamiento de actos judiciales al ser procesos generales y necesarios para el funcionamiento de los 6 subsistemas restantes del SIT, lo que valida la necesidad de la investigación y la actualidad del problema.
- ✓ No se encontró un sistema informático en el mundo que le diera solución al problema a resolver, aunque los sistemas internacionales estudiados aportaron ideas y conocimientos para el desarrollo de la solución propuesta.
- ✓ Del flujo de diseño se obtuvieron todos los artefactos del Modelo de diseño necesarios para la implementación del sistema, logrando así un mayor entendimiento de los procesos a implementar.
- ✓ Del flujo de implementación se obtuvo como resultado el Modelo de implementación, así como una versión entregable del producto, la cual luego de ser sometida a un conjunto de pruebas (Caja Negra, Caja Blanca, pruebas de Aceptación, pruebas de Regresión, pruebas Unitarias) y métricas de software (TOC y RC) por los desarrolladores, el grupo de calidad del proyecto, el cliente, y de corregidas las no conformidades encontradas, se encuentra listo para su liberación por CALISOFT y su posterior entrega al cliente.

El objetivo general de la investigación se cumplió, pues se logró implementar el subsistema Administración y Gobierno. El sistema desarrollado cumple con los requisitos para los cuáles fue implementado, posee una interfaz agradable al usuario y fácil de manipular. La comunicación con los demás subsistemas del SIT se establece mediante servicios web garantizando así la seguridad e integridad en los datos. Se creó una base de datos unificada para todo el sistema evitando así la redundancia de la información. Por todo lo antes planteado se concluye que la investigación fue completada con éxito.

Recomendaciones

Una vez cumplidos los objetivos de la investigación y teniendo en cuenta las experiencias obtenidas en la misma, se recomienda:

- ✓ Actualizar las librerías y marcos de trabajo de la aplicación SIT a las últimas versiones disponibles, con el fin de lograr mejoras sustanciales en el rendimiento del sistema.
- ✓ Incluir en versiones posteriores del SIT las solicitudes de cambio y mejoras detectadas en las revisiones al sistema.

Bibliografía consultada

1. González, R.A.H.L.y.S.C., El proceso de investigación científica.(2011): Editorial Universitaria del Ministerio de Educación Superior. 110.
2. Sampieri, R.H., Metodologia De La Investigacion. (1991).
3. Appleton, B. (02/14/2000) "Patterns and Software: Essential Concepts and Terminology."
4. Productiva, D. d. C. d. l. I. (2011). MODELO DE SISTEMA V2.0. SIT. Módulo Administración y Gobierno: 354.
5. Jurisoft, E. "Infolex - Gestion de Despachos, tomado dehttp://www.infolex.es/ie/index.aspx.
6. Software, B. "GEDEX Software para la Gestión de Expedientes Jurídicos para Despachos de Abogados y Profesionales, tomado dehttp://www.brindys.com/gedex/casmenu.html.
7. SRL, E. S. J. (2011). "Lex-Doctor, GESTIÓN JURÍDICA."tomado dehttp://www.lex-doctor.com/productos_estudiosjuridicos_info.php.
8. Morell, M. D. E. C. and M. J. R. G. Guadarramas (2008). Sistema para la Tramitación de Procesos Penales. Facultad Matemática-Física-Computación. Villa Clara,Cuba, Universidad Central „Marta Abreu“ de Las Villas.
9. Lira, K. G., Y. D. Rivero, et al. (2010). "ESTRATEGIA DE MODELADO DE PROCESOS DE NEGOCIO PARA EL DESARROLLO DE SISTEMAS DE INFORMÁTICA JURÍDICA.".
10. "Ingenieros de Software. Patrones de diseño.",
    tomado de http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.
11. "NetBeans 6.5 el único IDE que necesitas.",
    tomado de http://www.techbloog.com/talks/netbeans65es_cl.pdf.
12. "Documentación-Manual de PHP."
    tomado de http://www.hospedajeydominios.com/mambo/documentacion-manual_php.html.
13. "ExtJS en Español. Comunidad de desarrollo ", tomado de http://extjs.es/.
14. La biblia Servidor Apache 2.
15. "NetBeans IDE. Sitio Oficial.", tomado de http://netbeans.org/.
16. "¿Qué es Doctrine ORM? ." tomado dehttp://www.tecnoretales.com/programacion/que-es-doctrine-orm/.
17. Gamma, E., R. Helm, et al. Design Patterns.

18. Vizcaíno, A., F. Ó. García, et al. (2008). Trabajando con Visual Paradigm for UML. Univ. Cantabria – Fac. de Ciencias.
19. Renaud, P. E. (2007). Introduction to client/server systems: a practical guide for systems professionals. W. 1993. Universidad de Michigan, Wiley professional computing**:** 333 páginas.

Bibliografía referenciada

1. Díez de Castro Emilio Pablo, G. d. J. J., Martín Jimenez Francisca , Periáñez Cristobal Rafael (2004). Administración Administración y Dirección, McGraw-Hill Interamericana: 4.
2. Heinz, K. H. y. W. (2004). Administración Un Perspectiva Global, McGraw-Hill Interamericana: 6 y 14.
3. Pérez, J. A. H. (2008) Módulo de Administración en un Sistema Informático.
4. Ecu-Red. (2011). "Informática Jurídica - EcuRed." [En línea] [Citado el :2012-02-25 14:49:10], tomado dehttp://www.ecured.cu/index.php/Informática_jurídica.
5. Hernández, J. L. (2010). "Sistema seguro de intercambio de documentos Lexnet." Ministerio de Justicia.
6. García, A. M. D. and R. O. Cuello (2007). "Iniciativas recientes de la e-justicia en España." Revista de Internet, DERECHO Y POLÍTICA: 9.
7. Juiz, O. (2009). INFORME DEL SISTEMA SISECO.
8. Figueroa, R. G., C. J. Solís, et al. (2008). METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES. Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación.
9. ¿Qué es un Framework? [En línea] [Citado el: 27 de 02 de 2012.] tomado de http://buenmaster.com/?a=554.
10. Madrid., F. d. i.-U. P. d. and U. D. d. I. d. Software Patrones del “Gang of Four”.
11. Gutierrez, J. A. S. (2007) PATRONES GRASP (Patrones de Software para la asignación General de Responsabilidad).Parte II. [En línea] [Citado el: 28 de 02 de 2012.] tomado dehttp://jorgesaavedra.wordpress.com/category/patrones-grasp/
12. QUESADA, R. A. D. and A. N. D. P. POPULAR (1997). Ley de los Tribunales Populares.[En línea] [Citado el: 28 de 02 de 2012.]tomado dehttp://www.gacetaoficial.cu/html/itribunales.html.
13. Sebastián, J. (2010) Modelo Vista Controlador – Definición y Características.
14. Perdomo, J. G. (2008). Arquitectura Multicapas Arquitectura y Bases de Datos.[En línea] [Citado el: 28 de 02 de 2012.] tomado de http://sistemasydatos.blogspot.com/2008/02/arquitectura-multicapas.html
15. Ivar Jacobson, G. B. y. J. R. (2000). El Proceso Unificado de Desarrollo de Software.

16. Guillerón, G. (2009). Patrón Inversión de Control (IoC). Gln Blog ingeniería&consultoría.[En línea] [Citado el 9 de abril de 2012] tomado de http://glnconsultora.com/blog/?p=3
17. EcuRed (Marzo 2012) Flujo de trabajo análisis y diseño.[En línea] [Citado el 10 de abril de 2012] tomado de http://www.ecured.cu/index.php/Flujo_de_Trabajo_An%C3%A1lisis_y_Dise%C3%B1o#Principales_artefactos_del_dise.C3.B1o
18. PRESSMAN, R.S. Ingeniería del Software: un enfoque práctico. Edtion ed.: McGraw-Hill/Interamericana de España, 2002.
19. POSTGRESQL, E.E.D.D.D. Manual del usuario de PostgreSQL. In., 2008.

Glosario de términos

TPC: Tribunales Populares Cubanos.

DCD: Diagrama de clases del diseño.

CU: Caso de uso.

DS: Diagrama de secuencia.

DC: Diagrama de componente.

CP: Casos de prueba, conjunto de condiciones o variables bajo las cuáles se determinará si el requisito de una aplicación es parcial o completamente satisfactorio

NC: No conformidades, problemas detectados en un artefacto según insatisfacción con el resultado final de un Elemento de Configuración, lo pactado con anterioridad con el cliente, o no cumplimiento de un requisito.

GRD: Generador de reportes dinámicos.

DQL: Lenguaje de consulta de datos.

UCID: Unidad de Compatibilización Integración y Desarrollo de Software para la Defensa.