Facultad 3

Implementación y Prueba del módulo Inspección a Locales de Detención y Centros Penitenciarios.

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

Autor(es): *Manuel E. Delgado Fernández*
Reiner Mangly López

Tutor(es): *Ing. Marieta Peña Abreu*
Ing. Dina Yasilik Torres

La Habana, Junio del 2011
"Año 53 de La Revolución"

DECLARACIÓN DE AUTORÍA

Declaro que somos los únicos autores de este trabajo y autorizo al Centro de Gobierno Electrónico de la Universidad de las Ciencias Informáticas para que hagan el uso que estimen pertinente con este trabajo.

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

_____ _____

Firma del Autor Firma del Autor

_____

Firma de la Tutora

DEDICATORIA

A mi mamá y al tato por apoyarme incondicionalmente en todo.

A mi hermana por ser mi ejemplo a seguir.

A mi hermanito por ser el objetivo de nuestro ejemplo de estudio y esfuerzo.

A mis abuelos por su cariño de toda la vida.

A mi esposa por estar a mi lado durante todo este tiempo.

A toda la familia por estar pendiente y preocuparse por mí.

A mis compañeros de toda la vida y a los de la universidad por los buenos momentos y el apoyo.

Reiner

A mi mamá por ser mi inspiración.

A mis abuelos, mis tíos mis primos por todo su apoyo de toda la vida.

A mi novia por estar presente en estos años.

A mis compañeros de toda la vida y a los de la universidad por los buenos momentos y el apoyo.

Manuel

RESUMEN

La Fiscalía General de la República de Cuba (FGR), es la institución rectora del sistema judicial cubano y el órgano del Estado responsable del control y conservación de la legalidad dentro del sistema social, cuyo funcionamiento está basado en el cumplimiento estricto de la Constitución y las leyes, creadas y aprobadas por los diferentes órganos del Estado, las distintas entidades y el pueblo en general en pleno ejercicio de la democracia.

El presente trabajo de diploma tiene como objetivo principal, implementar y probar el módulo de Inspección a Locales de Detención (LD) y Centros Penitenciarios (CP), que responde al proceso fiscal donde se planifican, preparan, ejecutan y concluyen las visitas de inspección a los establecimientos penitenciarios existentes para chequear su correcto funcionamiento. Este módulo forma parte de la informatización del proceso de Control de Legalidades en Establecimientos Penitenciarios (CLEP) dentro del Sistema de Gestión Fiscal (SGF), proyecto que tiene como objetivo la informatización de procesos que se desarrollan en la FGR con el fin de favorecer la toma de decisiones y minimizar errores, así como optimizar el tiempo y la eficiencia del trabajo.

Para lograr el objetivo de la investigación se propone un estudio de los diferentes artefactos generados en los flujos de análisis y diseño y su utilización como entradas para implementar y probar un módulo que automatice de manera confiable y eficiente el proceso de Inspección a Locales de Detención y Centros Penitenciarios.

Palabras Claves:

Centros Penitenciarios, Locales de Detención, proceso, implementación.

ÍNDICE

INTRODUCCIÓN

Actualmente la sociedad enfrenta cambios y transformaciones que han provocado impacto en todos los campos del saber humano, la incorporación de las Tecnologías de la Información y la Comunicaciones (TIC), ha promovido la necesidad de reconceptualizar los métodos para procesar, almacenar y transmitir información.

De ahí que sea necesario para las sociedades actuales el acceso a las nuevas tecnologías existentes en el campo de la Información y la Comunicaciones y el desarrollo de sistemas que informaticen sus funciones elementales. Cuba no está exenta de estos cambios y adopta, apoyado en la Universidad de las Ciencias Informáticas (UCI) y otras entidades, la difícil tarea de la informatización de la sociedad cubana.

Una de las instituciones que está inmersa en este proceso es La Fiscalía General de la República (FRG) que, según el artículo 127, capítulo XIII, de la Constitución de la República de Cuba, es el órgano del Estado al que corresponde, como objetivos fundamentales, controlar y preservar la legalidad en la sociedad cubana; esta entidad no cuenta con un sistema para gestionar sus procesos fundamentales de una forma óptima y para ello se creó el proyecto Sistema de Gestión Fiscal (SGF) en la UCI, que se encargará de automatizar las distintas funciones que se realizan en los órganos de la fiscalía.

El proyecto SGF dividió su desarrollo en varios subsistemas que están relacionados con los procesos que se ejercen en cada uno de los órganos de la fiscalía. Uno de ellos es el Control de la Legalidad en los Establecimientos Penitenciarios (CLEP), cuyo propósito fundamental es la gestión de la información referente al cumplimiento de la legalidad en la ejecución de las sanciones privativas de libertad, las medidas de seguridad de internamiento y las medidas cautelares de prisión provisional, así como la detención o aseguramiento de los ciudadanos en cualquier centro de reclusión, internamiento o local de detención. El subsistema CLEP está dividido en 3 módulos: Quejas, Reclamaciones y Denuncias; Dictámenes e Inspección a Locales de Detención (LD) y Centros Penitenciarios (CP).

El módulo de Inspección a LD y CP se encarga, de la planificación, preparación, ejecución y conclusión de las visitas de inspección a los establecimientos penitenciarios, centros de reclusión de asegurados, centros correccionales, unidades

en que se cumpla la prisión provisional de acusados y cualquier otro centro de reclusión o internamiento.

Para todos los procesos derivados de la inspección, los fiscales autorizados deben registrar y consultar un considerable número de normativas o disposiciones legales. Actualmente toda la información generada en las fiscalías se gestiona de forma manual y queda almacenada en formato duro, nombrado rollo, como resultado de la comprobación de los objetivos que se hayan propuesto a verificar durante la ejecución de las inspecciones. Esto trae consigo que la consulta de la información sea engorrosa debido al gran cúmulo de documentos y que en ocasiones los fiscales puedan incurrir en violaciones graves por no dar respuesta a un determinado asunto en el tiempo establecido. A partir de lo antes descrito surge la necesidad del desarrollo de una aplicación que informatice los procesos de Inspección a LD y CP de la FGR.

Lo antes descrito se recoge en una tesis de pre-grado precedente que se enmarca, dentro del proceso de desarrollo de software, en las fases conceptual y estructural. Se obtuvieron, como resultado de esta investigación, los artefactos del análisis y el diseño: modelo de negocio, especificación de requisitos y modelo de casos de uso del sistema.

Teniendo en cuenta lo expuesto anteriormente se tiene como **problema de investigación**: Las insuficiencias de los mecanismos manuales están afectando la adecuada gestión de los procesos de Inspección a Locales de Detención y Centros Penitenciarios en la Fiscalía General de la República

En consecuencia **el objeto de estudio** se enmarca en el Proceso de Desarrollo de Software para la gestión fiscal.

Para ello se traza como **objetivo general**: Realizar la implementación y pruebas del módulo Inspección a Locales de Detención y Centros Penitenciarios para la FGR. Se define como **campo de acción**: Implementación y Pruebas del módulo Inspección a Locales de Detención y Centros Penitenciarios. Además se plantea **la idea a defender** de que “con el desarrollo de un módulo para la gestión de los procesos en las inspecciones a Locales de detención y Centros Penitenciarios corrigiendo las deficiencias actuales entonces se obtendrán mejoras en la gestión de los procesos de la FGR”.

Para darle cumplimiento al objetivo general se definen los siguientes **objetivos específicos**:

1. Realizar el estudio de los artefactos del análisis y el diseño.
2. Investigar los softwares de gestión fiscal existentes.
3. Seleccionar herramientas, lenguajes de programación y metodologías.
4. Implementar las funcionalidades del Módulo de Inspección a CP y LD.
5. Realizar pruebas a la aplicación.

Para satisfacer estos objetivos específicos quedan definidas las siguientes **tareas de la investigación**:

1. Estudio del trabajo de diploma precedente.
2. Realización del Estado del arte.
3. Implementación de las funcionalidades del módulo Inspección a Locales de Detención y Centros Penitenciarios.
4. Realización de las pruebas de software para comprobar que el sistema funciona correctamente.

Para realizar dichas tareas de investigación se ponen en práctica los siguientes Métodos Científicos:

Métodos empíricos:

Entrevista: Se realizarán una serie de entrevistas de forma informal al fiscal especialista involucrado por parte de la Fiscalía General de la República en el desarrollo del subsistema CLEP, que se tomarán como modelo para la obtención de toda la información necesaria.

Métodos teóricos:

Analítico-Sintético: Se han estudiado los mecanismos que se utilizan en las diferentes fiscalías del país referente a las inspecciones a LD y CP y se hará uso de la documentación derivada de los flujos de análisis y diseño.

Histórico-Lógico: Para el análisis de sistemas informáticos de gestión jurídica a nivel nacional e internacional, así como investigaciones realizadas anteriormente de forma tal que se obtenga información vital para la obtención a una solución.

Hipotético-Deductivo: La investigación se traza un problema concreto, a partir de este se han planteado los objetivos y posibles soluciones que en el transcurso de la investigación son resueltos mediante el uso de métodos bien fundamentados.

La presente investigación está conformada por 3 capítulos cuya descripción es expuesta brevemente a continuación:

Capítulo 1: **"Fundamentos teóricos de la investigación".** Se describen los conceptos asociados al dominio del problema. Se muestra el estado del arte de los distintos sistemas de gestión relacionados con actividades fiscales o jurídicas tanto a nivel nacional e internacional, a los que se realiza un análisis crítico de sus ventajas y desventajas. Se caracterizan las tecnologías, metodologías y herramientas y se realiza la justificación de las seleccionadas para la solución del problema.

Capítulo 2: **"Implementación del módulo de Inspección a Locales de Detención y Centros Penitenciarios".** En este capítulo se realiza la descripción y el análisis de la solución obtenida con el apoyo de los artefactos del análisis y diseño propuesto por los analistas, como son los estándares de código a utilizar, el modelo de la Base de Datos, los diagramas de clases del diseño y la descripción de las funcionalidades a automatizar.

Capítulo 3: **"Validación de la solución".** En este capítulo se realiza la validación de los resultados obtenidos a través de pruebas de software que garanticen la calidad de la solución propuesta.

CAPÍTULO 1: "FUNDAMENTOS TEÓRICOS DE LA INVESTIGACIÓN"

INTRODUCCIÓN

Informatizar procesos en la actualidad se ha convertido en la opción principal de las empresas y entidades que pretenden obtener mejoras en su funcionamiento, evolucionar en el mercado o mantenerse a la par de los avances de las TIC. No siempre resulta en un procedimiento de automatización simple, pues no todas las funcionalidades de una entidad pueden ser automatizadas y los beneficios de la informatización a veces no son suficientes como para compensar el desarrollo de una aplicación. También existen numerosas vías desde las que se puede llegar a la construcción exitosa de un producto informático que cumpla con lo esperado por el cliente. Por esto se necesita realizar un estudio profundo no solo de lo que el cliente desea informatizar, sino también de cuál es la vía óptima y las herramientas necesarias para construir un software robusto y con calidad. En este capítulo se ofrece una fundamentación científica de todas las herramientas, metodologías y técnicas que se utilizan para conseguir el objetivo de esta investigación.

1.1 Sistemas Informáticos de Gestión Fiscal existentes en el mundo.

Infolex de acuerdo a las valoraciones de (Ruíz y Morejon, 2010) es un Software de Gestión Integral para Despachos Jurídicos. Fue desarrollado en España por Jurisoft, empresa dedicada a la Informática Jurídica. Permite realizar la gestión de expedientes, por lo que ofrece múltiples sistemas de localización de expedientes que facilita al usuario búsquedas ágiles y sencillas. Está dotado de una gran flexibilidad, debido a que se adapta a las necesidades de cada profesional y da cobertura a todas sus especialidades: Despachos Judiciales o Extrajudiciales, Letrados de Entidades Bancarias, Compañías de Seguros o Despachos Mercantilistas y otras. Cuenta además de un amplio Registro de poderes de clientes para llevar su control de forma eficaz y segura.

Infolex comprende la emisión de un gran número de listados que permiten extraer la información que le interese al cliente en cada caso: Listados e Informes sobre Clientes, Expedientes y seguimiento Procesal y Extraprocesal. Incluye además un exhaustivo control de la producción de todos y de cada uno de los profesionales del Despacho.

IurisExplorer es un software desarrollado en Argentina que posibilita organizar y consultar rápidamente la información manejada por el usuario. Permite gestionar el

trabajo de todo el estudio jurídico como administrar expedientes, redactar escritos, intercambiar documentación, llevar un orden de las causas y ordenar sus cosas particulares. Su diseño facilita satisfacer y administrar las más complejas exigencias del trabajo de un abogado, tanto en su desempeño individual como con tareas que involucren altos volúmenes, se adapta a las necesidades de los profesionales. IurisExplorer no sólo puede realizar modelos de Escritos, sino también modelos de expedientes con la estructura necesaria y el contenido predeterminado indicado. (Ruíz y Morejon, 2010)

Lex-Doctor es un conjunto de softwares jurídicos desarrollado en Argentina por Sistemas Jurídicos SRL, empresa que desarrolla y comercializa sistemas para ser aplicados al ámbito jurídico. En su web oficial (Sistemas Jurídicos SLR, 1999) afirman que el sistema permite realizar la gestión de expedientes, gestión documental, reportes y estadísticas. Su tecnología de administración de datos, posibilita manejar grandes volúmenes de información, y organizar grupos de trabajo operando en redes de gran cantidad de terminales. Brinda poderosas herramientas de búsqueda, clasificación, y análisis de datos, que permiten la creación de reportes que involucran cualquier dato cargado en cada expediente, y reportes que involucren solamente expedientes que poseen determinadas características.

Gedex en relación con (BRiNDYS Software, 1996) es un software jurídico con amplia experiencia en el sector. 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. Permite realizar el seguimiento completo de los expedientes jurídicos de un despacho, bufete o departamento. Este producto es aplicable a disímiles modos de gestión, gracias a su adaptabilidad, integrándose en su despacho sin cambiar el tratamiento de la información al que se encuentre habituado. Permite localizar y vincular rápidamente expedientes y contactos. Gedex ofrece un avanzado sistema de contraseñas, con el que puede limitar el acceso a la información, ocultar expedientes y contactos a ciertos pasantes o empleados.

1.2 Antecedentes de software de gestión fiscal existentes en Cuba.

Softlex es un sistema que brinda servicios de tratamiento documental, almacenamiento y consulta de legislaciones agrupadas en listas organizadas por: emisión de la gaceta, institución que genera la norma, número de la normativa, tipo de normativa y página de la gaceta. Brinda además la posibilidad de contar con un índice referativo de la actividad jurídica en un período de tiempo determinado. Da tratamiento a normativas y tiene un módulo de seguimiento de asuntos, hasta cierto punto

configurable por el usuario. Posee un número muy limitado de funcionalidades y su construcción es basada en las legislaciones y no en los procesos fiscales que se llevan a cabo en la Fiscalía. (Ruíz y Morejon, 2010)

Después de analizar los softwares que antes se mencionan, de cara a la investigación se puede decir que aun cuando el origen del Sistema Jurídico Cubano proviene del mismo sistema que los países mencionados, tal es el caso de España y Argentina, sus esquemas legislativos son muy diferentes del esquema socialista establecido en Cuba. Por esta razón es imposible adaptar estos sistemas para un funcionamiento óptimo dentro del sistema judicial.

Otro elemento importante son los costos elevados de estos sistemas, además del mantenimiento de sus licencias de uso, hacen que Cuba no sea capaz de acceder a estos sistemas por su situación económica actual.

Existe otro asunto fundamental sobre estos sistemas y es su instalación sobre Windows, tal es el caso del sistema IurisExplorer y Gedex que son nativos de este sistema operativo y por esta razón no se ajustan a las perspectivas de migración a Software Libre que se desarrolla en el país.

Teniendo en cuenta lo antes planteado se considera que Cuba debe auto proporcionarse un sistema con las características necesarias que se ajusten al sistema judicial socialista cubano y cumpla con las expectativas del proceso de informatización que se lleva a cabo en el país. Construyendo este sistema con herramientas profesionales de código abierto que permitan una solución económica.

1.3 Metodología de desarrollo de software.

Para transitar por las fases del proceso de desarrollo del software es necesario aplicar una de las metodologías existentes, las cuales proponen las pautas para arribar a un software con calidad.

Rational Unified Process (RUP).

Las especificaciones de (Universidad Politécnica de Valencia, 2004) y (Jacobson y Rumbaugh, 2000) revelan que RUP es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un software. Sin embargo; es más que un simple proceso, es un marco de trabajo genérico que puede especializarse para una gran variedad de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos. RUP

es una guía de cómo usar UML de la forma más efectiva. Existen tres características claves presentes en RUP, ellas son:

- **Dirigido por casos de uso**: Teniendo en cuenta que la razón de ser de un sistema es brindar servicios a los usuarios, define caso de uso como el conjunto de acciones que debe realizar un sistema para dar un resultado de valor a un determinado usuario y los utiliza tanto para especificar los requisitos funcionales del sistema, como para guiar todos los demás pasos de su desarrollo, dígase diseño, implementación y prueba.
- **Centrado en la arquitectura**: La arquitectura es una vista del diseño completo con las características más importantes, dejando a un lado los detalles. Esta no solo incluye las necesidades de los usuarios e inversores, sino también otros aspectos técnicos como el hardware, sistema operativo, sistema de gestión de base de datos, protocolos de red; con los que debe coexistir el sistema. En otras palabras, la arquitectura representa la forma del sistema, la cual va madurando en su interacción con los casos de uso hasta llegar a un equilibrio entre funcionalidad y características técnicas.
- **Iterativo e incremental**: La alta complejidad de los sistemas actuales hace que sea factible dividir el proceso de desarrollo en varios mini-proyectos. Cada uno de estos mini-proyectos se les denomina iteración y pueden o no representar un incremento en el grado de terminación del producto completo. En cada iteración los desarrolladores seleccionan un grupo de casos de uso, los cuales se diseñan, implementan y prueban. La iteración controlada reduce el riesgo de no sacar al mercado el producto en el calendario previsto, permite mantener la motivación del equipo, pues puede ver avances claros a corto plazo y que el desarrollo pueda adaptarse a los cambios en los requisitos.

RUP divide el proceso de desarrollo en ciclos, teniendo un producto al final de cada ciclo. Cada ciclo se divide en cuatro fases, y cada fase concluye con un hito bien definido.

1. **Inicio**: Descripción del negocio y alcance del proyecto.
2. **Elaboración**: Se define la arquitectura del sistema.
3. **Construcción**: Funcionalidad operativa del software.
4. **Transición**: Liberación del software.

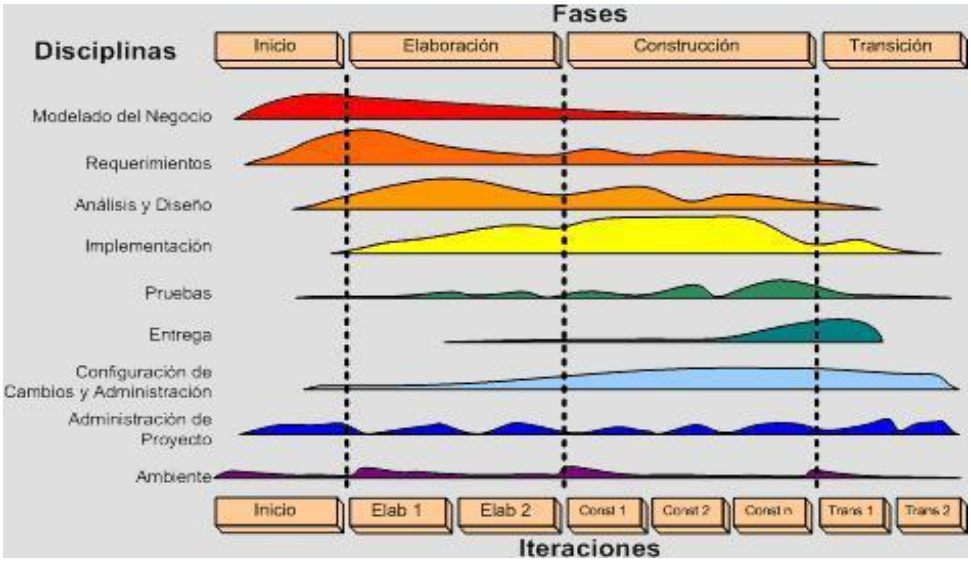

Ilustración 1: Fases y Flujos de trabajo de RUP.

Las disciplinas de RUP son: Modelado del negocio, Requerimientos, Análisis y Diseño, Implementación, Pruebas, Despliegue, Gestión del proyecto, Configuración y Control de Cambios y Entorno.

El elemento más importante que hace de RUP la correcta elección lo constituye sin duda la capacidad detallada de documentar todo el proceso de desarrollo del software, pues en el proyecto se maneja información confidencial y sensible perteneciente a la Fiscalía General, además del perfecto conocimiento sobre el funcionamiento de la misma lo que hace necesario llevar una documentación detallada del desarrollo del proyecto.

1.4 Técnicas de Programación.

Una técnica o paradigma de programación es un modelo de desarrollo de aplicaciones que se ha perfeccionado con el paso del tiempo y permite producir programas basados en un conjunto de reglas definidas.

Programación estructurada

La programación estructurada es una forma de escribir programas de ordenador (programación de computadora) de manera clara. Para ello utiliza únicamente tres estructuras: secuencia, selección e iteración; siendo innecesario el uso de la instrucción o instrucciones de transferencia incondicional (GOTO, EXIT FUNCTION, EXIT SUB o múltiples RETURN). Los programas son fáciles de entender, ya que pueden ser leídos de forma secuencial, sin necesidad de hacer seguimiento a saltos de línea (GOTO) dentro de los bloques de código para entender la lógica. El seguimiento de los fallos o errores del programa ("debugging") se facilita debido a la

estructura más visible, por lo que los errores se pueden detectar y corregir más fácilmente. El principal inconveniente de este método de programación es que se obtiene un único bloque de programa, que cuando se hace demasiado grande puede resultar problemático su manejo, como es el caso del software que se desea implementar en cuanto a complejidad y tamaño del mismo. (Programación Estructurada, 2004)

Programación Orientada a Objetos (POO)

La programación orientada a objetos es un paradigma de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas informáticos. Está basado en varias técnicas, incluyendo herencia, abstracción, polimorfismo y encapsulamiento. Su uso se popularizó a principios de la década de los años 1990. Un objeto es una abstracción de algún hecho o ente del mundo real que tiene atributos que representan sus características o propiedades y métodos que representan su comportamiento o acciones que realizan. Todas las propiedades y métodos comunes a los objetos se encapsulan o se agrupan en clases. Una clase es una plantilla o un prototipo para crear objetos, por eso se dice que los objetos son instancias de clases. Esto permite hacer los programas y módulo más fáciles de escribir, mantener y reutilizar. En la actualidad, existe variedad de lenguajes de programación que soportan la orientación a objetos. (Programacion Orientada a Objetos, 2004)

Luego de haber realizado un estudio de algunas de las técnicas de programación existentes en la actualidad, se selecciona para la realización de la presente investigación la POO. Esta técnica fomenta la reutilización del código, facilitando el trabajo en equipos. Además permite la creación de sistemas complejos y agiliza considerablemente el desarrollo de software. En la universidad su uso es prácticamente generalizado y ha permitido el desarrollo de aplicaciones en el pasado evidenciando los frutos de su utilización.

1.5 Lenguajes de programación

Lenguaje de programación JAVA

En correspondencia con lo planteado por (Muñoz y Rodríguez, 2010) Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. La implementación original y de referencia del compilador, la máquina virtual

y las bibliotecas de clases de Java fueron desarrolladas por Sun Microsystems en 1995.

Se destacan en la programación del lado del servidor los servlets y las Java Server Pages (JSP). Los servlets son programas que se ejecutan en el servidor y que extienden sus funcionalidades. Para su uso, además de la máquina virtual de Java se debe instalar un programa denominado contenedor de servlets, como por ejemplo el Tomcat. Por otra parte, las JSP son una forma alternativa y sencilla de construir servlets. Generan contenido web dinámico en forma de documentos HTML, aunque también generan otros formatos como XML. Permiten la utilización del código Java mediante scripts y etiquetas que aparecen dentro del documento HTML antes de ser enviados al cliente. Estas tecnologías facilitan en gran medida la implementación de sitios web dinámicos, por lo que han tenido un auge en popularidad.

Ventajas:

- Un lenguaje multiplataforma potente, se puede descargar de forma gratuita. Dispone de una gran cantidad de paquetes o librerías que añaden una potencia increíble al lenguaje.
- La integración con la red es asombrosa ya que sus posibilidades son muy altas. (WebTaller, 2007)

Desventajas:

- Su principal desventaja es el compilador pues este dice si un programa al compilarlo tiene errores pero no dice con precisión el lugar donde está el error. La Máquina virtual Java consume muchos recursos, así que se debe tener un ordenador con muy buenas propiedades sino se nota mucho la recarga de memoria. (WebTaller, 2007)

Lenguaje de programación PHP 5

PHP (Hipertexto Pre-processor) puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, soporta la mayoría de los servidores Web de hoy en día y ofrece soporte para unos 20 gestores de bases de datos. Su característica de ser software libre trae como consecuencia que implique menos costes y servidores más baratos. Es además muy rápido, contiene una biblioteca nativa de funciones sumamente amplia e incluida, no requiere definición de tipos de variables ni manejo detallado del bajo nivel, presenta mejoras de rendimiento, es un lenguaje multiplataforma que funciona en todas las plataformas que soporten Apache. No es un lenguaje de marcas y su principal meta es permitir rápidamente a los desarrolladores la generación dinámica de páginas. Soporta el uso de otros servicios que usen protocolos como IMAP, SNMP, NNTP, POP3, HTTP y derivados. También se pueden

abrir sockets de red directos (raw sockets) e interactuar con otros protocolos. Al ser un lenguaje libre dispone de una gran cantidad de características que lo convierten en la herramienta ideal para la creación de páginas Web dinámicas: Integración con varias bibliotecas externas, permite generar documentos en PDF (documentos de Acrobat Reader) y analizar código XML. (PHP, 2001)

Lenguaje HTML

Es el lenguaje de marcado predominante para la construcción de páginas Web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). HTML puede describir hasta un cierto punto la apariencia de un documento, y puede incluir un *script* (por ejemplo JavaScript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML. Es usado para referirse al contenido del tipo de MIME text/html o todavía más ampliamente como un término genérico para el HTML, ya sea en forma descendida del XML (como XHTML 1.0 y posteriores) o en forma descendida directamente de SGML. (Torre, 2000)

Lenguaje JavaScript

JavaScript es un lenguaje de programación utilizado para crear pequeños programas encargados de realizar acciones dentro del ámbito de una página Web. Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. El código Javascript es embebido directamente en el código HTML, haciendo fácil la creación de páginas Web con contenido dinámico. Está diseñado para controlar la apariencia y manipular los eventos dentro de la ventana del navegador Web y es soportado por la gran mayoría de los navegadores lo que lo coloca como el lenguaje de programación web del lado del cliente más utilizado.

Con JavaScript se pueden crear efectos especiales en las páginas y definir interactividades con el usuario. El navegador del cliente es el encargado de interpretar las instrucciones JavaScript y ejecutarlas para realizar estos efectos e interactividades, de modo que el mayor recurso, y tal vez el único, con que cuenta este lenguaje es el propio navegador. JavaScript es el siguiente paso, después del HTML, que puede dar un programador de la web que decida mejorar sus páginas y la potencia de sus proyectos. Es un lenguaje de programación bastante sencillo y pensado para trabajar con rapidez, a veces con ligereza. Entre las acciones típicas que se pueden realizar en JavaScript se tienen dos vertientes. Por un lado los efectos especiales sobre páginas web, para crear contenidos dinámicos y elementos de la página que tengan

movimiento, cambien de color o cualquier otro dinamismo. Por el otro, Javascript permite ejecutar instrucciones como respuesta a las acciones del usuario, con las que se pueden crear páginas interactivas con programas como calculadoras, agendas, o tablas de cálculo. Es un lenguaje con muchas posibilidades, permite la programación de pequeños scripts, pero también de programas más grandes, orientados a objetos, con funciones, estructuras de datos complejas, etc. (Javascript, 2005)

Un elemento importante en PHP que facilita el trabajo del desarrollador lo constituye la no definición de tipos de variables. Se selecciona por su capacidad de ser multiplataforma, su alto rendimiento, bajo coste de desarrollo y la existencia de grandes comunidades de desarrollo de PHP donde se brinda a los desarrolladores soluciones y experiencias para perfeccionar su trabajo. Se utiliza además en el desarrollo los lenguajes HTML para la programación de código del lado del cliente, JavaScript para las validaciones y mejoras del diseño de interfaz y CSS para el estilo de diseño que se les dará a las páginas.

1.6 Herramientas

1.6.1 Frameworks de desarrollo en PHP

Framework de Desarrollo Symfony 1.3.

Symfony 1.3, es un completo framework diseñado para optimizar el desarrollo de las aplicaciones web gracias a sus características. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. (Potencier y Zaninotto., 2008)

Características

- Fácil de instalar y configurar en sistemas Windows, Mac y Linux.
- Funciona con todas las bases de datos comunes (MySQL, PostgreSQL, SQLite, Oracle, MS SQL Server).
- Basado en la premisa de *"convenir en vez de configurar"*, en la que el desarrollador solo debe configurar aquello que no es convencional.
- Preparado para aplicaciones empresariales, ya que se puede adaptar con facilidad a las políticas y arquitecturas propias de cada empresa u organización.

- Flexible hasta cualquier límite y extensible mediante un completo mecanismo de plugins.
- Publicado bajo licencia MIT de software libre y apoyado por una empresa comprometida con su desarrollo.
- Traducido a más de 40 idiomas y fácilmente traducible a cualquier otro idioma.
- Protección ante ataques XSS y CSRF obligatoria.

Symfony está basado en el patrón de arquitectura MVC (Modelo-Vista-Controlador), que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y el controlador representa la Lógica de negocio. (Potencier y Zaninotto., 2008)

Framework de Desarrollo Zend

Zend Framework es un framework de código abierto para desarrollar aplicaciones web y servicios web con PHP 5. La estructura de sus componentes es algo único; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado. Entre los componentes qué destacaría se encuentran: Zend_Config para temas de configuración de aplicaciones web, Zend_Db para tratar con bases de datos, Zend_Search o Zend_Feed entre otros.

Zend Framework ofrece un gran rendimiento y una robusta implementación del patrón Modelo Vista Controlador, una abstracción de base de datos fácil de usar, validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos. (Zend Studio, 2006)

Un elemento importante es su dependencia a la plataforma “Zend Studio” la cual es un poco limitada en cuanto a posibilidades de desarrollo y exige un alto rendimiento de la máquina donde se desarrolle.

Framework de Desarrollo phpMVC

Implementa el patrón de diseño Modelo-Vista-Controlador (MVC) y alienta el diseño de aplicaciones basadas en el paradigma del modelo. Este modelo de diseño permite que la página web u otros contenidos puedan ser en su mayoría, separados del código y de la aplicación interna (Controller / Model), lo que hace que sea más fácil para los diseñadores y programadores centrarse en sus respectivas esferas de competencia. El contenido resultante se devuelve al navegador del cliente o a través de otro protocolo

como SMTP. Es un framework que no está muy desarrollado en la actualidad y no cuenta con muchos seguidores de modo que su auge inicial ha sido opacado por otros frameworks. ()

A pesar de las numerosas ventajas que poseen los frameworks Zend y phpMVC se selecciona Symfony porque es independiente de plataforma, corre en el IDE de desarrollo NetBeans 6.9 cuya selección se fundamenta más adelante, es compatible con la mayoría de los gestores de bases de datos y posee una amplia bibliografía y foros en español a los que se les puede hacer referencias en cualquier situación o duda. Esto permite la agilidad de la implementación y la reducción del tiempo de capacitación de los desarrolladores.

1.6.2 IDES de desarrollo

NetBeans 6.9.

NetBeans es un proyecto exitoso de código abierto con una gran base de usuarios, una comunidad en constante crecimiento, y con cientos de socios en todo el mundo. "Sun MicroSystems" fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos.

NetBeans IDE es un entorno de desarrollo - una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans y es un producto libre, gratuito y sin restricciones de uso. (Sun Microsystem, 2006)

De acuerdo con las valoraciones de (El código K, 20009) y (Sun Microsystem, 2006), las principales características del IDE de desarrollo que lo convierten en una opción importante para la investigación son:

- **Creación de Proyectos PHP.**

NetBeans provee de una estructura para los proyectos que se puede crear junto a este IDE, propone un esqueleto para organizar el código fuente, el editor conjuntamente integra los lenguajes como HTML, JavaScript y CSS. Además NetBeans posee un sistema para examinar todo los directorios de cada proyecto, haciendo reconocimiento y carga de clases, métodos y objetos, para acelerar la programación.

- **Integración con Symfony.**

Una de las características que lleva a utilizar NetBeans es justamente su integración con estos populares Framework de PHP, realizar aplicaciones con estos Framework es muy ágil. Gracias a NetBeans ya es posible dejar de lado la consola de comandos de Symfony y centrarse en desarrollar en el IDE, además se encuentra cargadas todas las clases, ayuda en línea, etc.

- **Editor de Código Fuente**.

A lo largo de todo este tiempo, se notó la mejora en su editor, sobre todo en el editor de PHP, es mucho más ágil y a la vez robusto, contiene más ayuda en línea, reconocimiento de sintaxis y todo lo que provee la última versión de PHP, la 5.3.

- **Integración con PHP Unit Testing**

Es posible crear test con PHPUnit, para diferentes funciones, luego realizar la comprobación y ver todos los resultados. En las propiedades PHPUnit puede definir una configuración personalizada de archivos XML, un archivo de arranque para las opciones de línea de comandos, o una serie de pruebas a medida, o puede que el IDE genera el código esqueleto para usted.

- **Depuración de PHP**.

NetBeans integra muy bien la utilización Xdebug, gracias a esto se puede inspeccionar y examinar cada variable local, establecer puntos de interrupción y evaluar el código en la lógica.

El IDE de NetBeans para PHP también ofrece la línea de comandos de depuración: La salida del programa PHP aparece en una pantalla de línea de comandos en el IDE de sí mismo y se puede inspeccionar el código HTML generado sin tener que cambiar a un navegador.

- **Integración con Sistemas de Control de Versiones**.

Esta es una de las condiciones necesarias para los proyectos y es la posibilidad de contar con la integración de sistemas de control de versiones, tales como SVN, CVS, Mercurial y Git.

Desde el editor es posible realizar la administración de estos sistemas versionados, a través de acciones como commit, branch, importar, exportar, revert, clonar entre otras.

ECLIPSE

Es considerado tan potente como popular. Incorpora un sinfín de utilidades para simplificar la labor de los programadores. Aparte de ser un entorno de desarrollo súper completo, una de las particularidades más interesantes para la comunidad es que es de código libre y gratuito. Está programado en Java, por lo que se necesita el entorno de ejecución de Java instalado para que funcione. Existen diversos plugins o añadidos para proveer de nuevas utilidades al programa, enfocadas a diversos usos que los programadores pueden necesitar. Uno de los añadidos de Eclipse más interesante para los desarrolladores de páginas web, es el módulo para programación en PHP, que está formado de varios componentes. Como beneficio de todas las ventajas para programar en PHP se puede descargar Eclipse, instalarlo y luego instalar dentro de Eclipse los componentes necesarios. (IDE php, 2007)

Zend Studio

Es un IDE para PHP, soportando PHP 4 y PHP 5. Está escrito en Java, y disponible para diversas plataformas, como Windows, MacOS-X y GNU/Linux, aunque es un software privado. No requiere la instalación previa de PHP ni del entorno de ejecución de Java. Presenta resaltado de sintaxis, autocompletamiento de código, detección de errores de sintaxis en tiempo real, ayuda de código (integra phpDoc) y lista de parámetros de funciones y métodos de clase. (Zend Studio, 2006)

La versatilidad del IDE NetBeans y su amplia posibilidad de aplicación garantizan que el desarrollo se realice de manera ágil y segura, que pueda utilizarse elementos actuales que faciliten la implementación y la integración con el resto de las herramientas que conforman la infraestructura de desarrollo del proyecto como el lenguaje PHP 5 y el framework Symfony. Es multiplataforma, software de libre distribución y posee un gran número de módulos externos que extienden sus funcionalidades.

1.6.3 Servidores Web

Internet Information Server (IIS)

Es una serie de servicios para los ordenadores que funcionan con Windows. Originalmente era parte del Option Pack para Windows NT. Luego fue integrado en otros sistemas operativos de Microsoft destinados a ofrecer servicios, como Windows 2000 o Windows Server 2003. Windows XP Profesional incluye una versión limitada de IIS. Los servicios que ofrece son: FTP, SMTP, NNTP y HTTP/HTTPS. Este servicio convierte a un ordenador en un servidor de Internet o Intranet por lo que en las computadoras que tienen este servicio instalado, se pueden publicar páginas web

tanto local como remotamente. Los Servicios de Internet Information Services (IIS) proporcionan las herramientas y funciones necesarias para administrar de forma sencilla un servidor Web seguro. Se basa en varios módulos que le dan capacidad para procesar distintos tipos de páginas. Pueden ser incluidos los de otros fabricantes, como PHP o Perl. (Postigo, 2008)

Servidor Web Apache 2.

Hoy en día es el servidor web más utilizado del mundo, encontrándose muy por encima de sus competidores, tanto gratuitos como comerciales. Es un software de código abierto que funciona sobre cualquier plataforma. Por supuesto, se distribuye prácticamente con todas las implementaciones de Linux. Tiene capacidad para servir páginas tanto de contenido estático, como de contenido dinámico a través de otras herramientas soportadas que facilitan la actualización de los contenidos mediante bases de datos, ficheros u otras fuentes de información.

Es un software de libre distribución, que publica su código fuente, lo que permite que cualquiera pueda modificarlo y colaborar así a su desarrollo. Apache está diseñado para ser un servidor web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos. Esto induce que a menudo sean necesarias diferentes características o funcionalidades. Apache se ha adaptado siempre a una gran variedad de medios a través de su diseño modular. Este diseño permite a los administradores de sitios web elegir qué características van a ser incluidas en el servidor seleccionando que módulos se van a cargar, ya sea al compilar o al ejecutar el servidor. (Soluciones Informáticas, 2004)

Tiene interfaz con todos los sistemas de autenticación. Facilita la integración como "plugins" de los lenguajes de programación de Páginas Web dinámicas más comunes. Tiene integración en estándar del protocolo de seguridad SSL y provee interfaz a todas las bases de datos.

Por las características anteriormente descritas se puede concluir que el servidor Apache es el que reúne las mejores condiciones para ser utilizado en la investigación, debido a ser libre, fácilmente integrable con PHP y multiplataforma, entre otras ventajas.

1.6.4 Servidor de Base de Datos

Sistema Gestor de Base de Datos PostgreSQL.

PostgreSQL en concordancia con los planteamientos de (Equipo de desarrollo de PostgreSQL, 1999) y (Arpug, 2001) es un gestor de bases de datos relacional

orientada a objetos, libre y gratuito. Está liberado bajo la licencia BSD, lo que significa que se puede disponer de su código fuente, modificarlo a voluntad y redistribuirlo libremente. Contiene bloques de código que se ejecutan en el servidor y pueden ser escritos en varios lenguajes, con la potencia que cada uno de ellos brinda; desde las operaciones básicas de programación, tales como bifurcaciones y bucles, hasta las complejidades de la programación orientada a objetos o la programación funcional.

Presenta las siguientes propiedades:

Atomicidad: Asegura la realización de una operación, por lo que ante un fallo del sistema la operación fallida no compromete la integridad del sistema.
Consistencia: Posibilita la ejecución de aquellas operaciones que no van a romper las reglas y directrices de integridad de la base de datos.
Aislamiento: Mediante un sistema denominado MVCC (Acceso concurrente multiversión) asegura que una operación no pueda afectar a otras, de esta manera dos transacciones sobre la misma información no genera error.

Sistema Gestor de Base de Datos MySQL**.**
Es uno de los más importantes en cuanto al diseño y la programación de bases de datos de tipo relacional. Cuenta con millones de aplicaciones y aparece en el mundo informático como uno de los más utilizados por usuarios del medio. Se usa como servidor a través del cual pueden conectarse múltiples usuarios y utilizarlo al mismo tiempo. Una de las características más interesantes de MySQL es que permite recurrir a bases de datos multiusuario a través de la web y en diferentes lenguajes de programación que se adaptan a las necesidades existentes. Por otro lado, es conocido por desarrollar alta velocidad en la búsqueda de datos e información, a diferencia de otros sistemas. Las plataformas que utiliza son de diversos tipos y entre ellas se pueden mencionar; LAMP, MAMP, SAMP, BAMP y WAMP (aplicables a Mac, Windows, Linux, BSD, Open Solaris, Perl y Phyton entre otras). El código fuente se puede descargar y está accesible a cualquiera. Usa la licencia GPL para aplicaciones no comerciales. Es una base de datos muy rápida, segura y fácil de usar. Gracias a la colaboración de muchos usuarios, la base de datos se ha ido mejorando optimizándose en velocidad. Por eso es una de las bases de datos más usadas en Internet. (Victoria, 2009)

La UCI pertenece a la Comunidad Iberoamericana de PostgreSQL una organización que promueve el desarrollo con esa herramienta, por lo que su uso es condicionado por la propia tendencia existente en la universidad. Es importante mencionar que el servidor de bases de datos PostgreSQL será gestionado por una aplicación llamada

PgAdminIII que es una de las más completas y populares con licencia Open Source, escrita en C++ se puede usar en sistemas operativos como: GNU/Linux y Windows. Posee una interfaz gráfica que soporta todas las características de PostgreSQL y facilita enormemente la administración e incluso pueden realizarse cambios en el modelo de datos desde la herramienta de manera muy sencilla.

1.6.5 Sistema de control de versiones

Subversion (SVN) 1.4.

El sistema de control de versiones es un software que administra el acceso a un conjunto de ficheros y mantiene un historial de cambios realizados. El control de versiones es útil para guardar cualquier documento que se cambie con frecuencia, como el código fuente de un programa. Normalmente consiste en una copia maestra en un repositorio central, y un programa cliente con el que cada usuario sincroniza su copia local lo que permite compartir los cambios sobre un mismo conjunto de ficheros. Además, el repositorio guarda registro de los cambios realizados por cada usuario, y permite volver a un estado anterior en caso de necesidad. Es necesario el uso de este sistema de control de versiones dada las ventajas que brinda el mismo:

1. Actualización de ficheros modificados.
2. Copias de seguridad centralizadas.
3. Historial de cambios.
4. Brinda acceso remoto.
5. Provee seguridad al sistema. (Subversion, 2005)

El Subversion 1.4 posee un excelente dominio de las versiones y un mayor control del trabajo que se está realizando sobre cada uno de los elementos de configuración brindando soporte a todas las operaciones que se realizan sobre los mismos haciendo el entorno de desarrollo del proyecto inmune a errores fatales. Su característica de software libre hace que su selección sea una opción de prioridad para la investigación.

1.7 Pruebas de calidad

Prueba de Caja Negra

La prueba de Caja Negra se centra principalmente en los requisitos funcionales del software. Estas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos. La prueba de Caja Negra no es una alternativa a las técnicas de prueba de la Caja Blanca, sino un enfoque complementario que intenta descubrir diferentes

tipos de errores a los encontrados en los métodos de la Caja Blanca, como por ejemplo:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

Para preparar los casos de pruebas, se necesitan un número de datos que ayuden a la ejecución de estos casos y permitan que el sistema se ejecute en todas sus variantes. Se pueden introducir datos válidos o inválidos para el programa según lo que se desea: hallar un error o probar una funcionalidad. Los datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa corra bien. (D'Onofrio, 2006)

Pruebas de Usuario

Las pruebas realizadas por el usuario constituyen pruebas de caja negra que se centran en el cumplimiento de los requisitos definidos para el sistema una vez concluido. Por lo tanto, estas pruebas constituyen la validación de la aplicación por parte del usuario final y según Pressman plantean lo siguiente: *"La validación del software se consigue mediante una serie de pruebas de caja negra que demuestran la conformidad con los requisitos. Un plan de prueba traza la clase de pruebas que se han de llevar a cabo, y un procedimiento de prueba define los casos de prueba específicos en un intento por descubrir errores de acuerdo con los requisitos. Tanto el plan como el procedimiento estarán diseñados para asegurar que se satisfacen todos los requisitos funcionales, que se alcanzan todos los requisitos de rendimiento, que la documentación es correcta e inteligible y que se alcanzan otros requisitos (por ejemplo, portabilidad, compatibilidad, recuperación de errores, facilidad de mantenimiento)".* (Pressman, 2002)

Estas pruebas no solo permitirán garantizar que el software realizado cumple con las necesidades del cliente, también comprobará que fue construido siguiendo normas de desarrollo de software óptimas y que posee un código fuente seguro y fiable. Además las mismas validarán que el presente trabajo de diploma cumpla con su objetivo principal.

Prueba de Caja Blanca

La prueba de la caja blanca del software se encarga de comprobar los caminos lógicos del software proponiendo casos de prueba para que se ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado o mencionado. (D'Onofrio, 2006)

Algunos elementos utilizados alrededor de éste método son los siguientes:

- Grafo de flujo o grafo del programa: representa el flujo de control lógico de un programa y se utiliza para trazar más fácilmente los caminos de éste. Cada nodo representa una o más sentencias procedimentales y cada arista representa el flujo de control. (Caja Blanca, 2008)

- Complejidad ciclomática: es una métrica de software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto de las pruebas, el cálculo de la complejidad ciclomática representa el número de caminos independientes del conjunto básico de un programa. Esta medida, ofrece al probador de software, un límite superior para el número de pruebas que debe realizar, garantizando así que se ejecutan por lo menos una vez cada sentencia. (Caja Blanca, 2008)

- Camino independiente: cualquier camino del programa que introduce, por lo menos, un nuevo conjunto de sentencias de proceso o una nueva condición. (Caja Blanca, 2008)

Para la ejecución de las pruebas de caja blanca se seguirán los siguientes pasos descritos según la prueba estructurada de (Watson y McCabbe, 1996):

Paso 1: Obtener el grafo de flujo, a partir del diseño o del código del módulo.

Paso 2: Obtener la complejidad ciclomática del grafo de flujo.

Paso 3: Definir el conjunto básico de caminos independientes.

Paso 4: Preparar un CP por camino hallado en el paso anterior.

Para la ejecución de las pruebas se utilizará la herramienta de prueba PHPUnit es un framework para pruebas de unidad en específico a *PHP*, que permite realizar prueba rápidamente, que son fáciles de hacer, leer y analizar e independientes entre sí.

Presenta métodos *Assert* los cuales realizan una prueba y retornan un *"AssertionFailedError"* si la prueba falló.

Otros frameworks de pruebas estudiados resultaron ser de poca utilidad como el Zend test que es dependiente de la suite Zend Studio. También Symfony 1.3 incluye un framework para pruebas unitarias llamado Lime pero los desarrolladores de Symfony recomiendan utilizar PHPUnit e incluso en su última versión sustituyen Lime por PHPUnit.

Pruebas de Carga y Stress.

Las pruebas de Carga y Stress son realizadas con el objetivo de comprobar el funcionamiento del sistema bajo condiciones planificadas. Estas pruebas demuestran si la aplicación cumple con los diferentes criterios de rendimiento permitiendo conocer con precisión las estadísticas referidas a los diferentes procesos que se ejecuten durante el uso del sistema.

Para realizar las pruebas se utiliza la herramienta JMeter, una aplicación multiplataforma que permite simular el uso de diferentes aplicaciones creando registros de rendimiento de las mismas para facilitar su análisis.

1.8 Métricas

Métricas orientadas a objetos.

Dada la evidencia que existen sistemas basados en programación orientada a objetos (POO) que no son robustos, cuya mantenibilidad es cuestionable e incluso su grado de reusabilidad es mínimo, se procede a aplicar algunas de las métricas más relevantes creadas y aplicadas en el campo del desarrollo de software como producto de la evolución del paradigma orientado a objetos. El objetivo principal de la ejecución de estas métricas es garantizar que el sistema posea alta calidad y robustez.

Métrica de Proporción de Métodos Heredados (MIF)

MIF se define como proporción de la suma de todos los métodos heredados en todas las clases entre el número total de métodos (localmente definidos más los heredados) en todas las clases. (Abreu y Melo, 1996)

$$MIF = \frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_a(C_i)}$$

Donde:

$M_a(C_i) = M_d(C_i) + M_i(C_i)$

Y:

$M_d(C_i)$ *es el número de metodos declarados en una clase.*

$M_a(C_i)$ *es el numero de metodos que pueden se rinvocados en relacion a* C_i.

$M_i(C_i)$*es el numero de metodos heredados y no definidos en* C_i.

TC es el número total de clases en el sistema.

Métrica de Proporción de Atributos Heredados (AIF)

AIF se define como la proporción del número de atributos heredados entre el número total de atributos. (Abreu y Melo, 1996)

$$AIF = \frac{\sum_{i=1}^{TC} A_i(C_i)}{\sum_{i=1}^{TC} A_a(C_i)}$$

Donde:

$A_a(C_i) = A_d(c_i) + A_i(C_i)$

$A_d(C_i)$*es el número de atributos declarados en una clase.*

$A_a(C_i)$ *es el número de atributos que pueden ser invocados asociados a* C_i.

$A_i(C_i)$*es el numero total de atributos heredados y no definidos en* C_i.

TC es el número total de clases en el sistema.

Métrica de Tamaño Operacional de Clases.

Las métricas del producto son una medida cuantitativa que permite tener una visión profunda de la eficacia del proceso del software. Las métricas son también utilizadas para señalar áreas con problemas de manera que se puedan desarrollar estrategias para mejorar la calidad del desarrollo del software.

La métrica de tamaño operacional de clases (TOC) está condicionada por el número de métodos declarados para las clases del sistema y su resultado es reflejo de los indicadores de responsabilidad, complejidad de implementación y reutilización.

Los criterios de evaluación relacionados con el número de clases se encuentran relacionados con los atributos de calidad, los que se ven afectados de acuerdo a los valores medios calculados en la métrica.

Eficiencia de Algoritmos.

Los elementos que intervienen en la eficiencia de un algoritmo tienen una naturaleza variada, y están determinados fundamentalmente por el propio algoritmo, los datos de entrada y el medio de cómputo sobre el cual se ejecuta. El proceso de análisis se refiere específicamente a medir el uso eficiente de los recursos en función del tiempo que tarda en ejecutarse (**complejidad temporal**). (Mairelys y Osley, 2010)

Cotas de complejidad: Las cotas permiten asegurar, en el caso de la Cota máxima que el problema se puede resolver en ese tiempo como máximo; en el caso de la Cota

mínima que el problema no se puede resolver con ningún algoritmo que tenga un tiempo de ejecución menor que el de la función que lo acota.

Órdenes de Complejidad: Se dice que O (f(n)) define un "orden de complejidad", de manera que se tiene: O (1) orden constante, O (log n) orden logarítmico, O (n) orden lineal, O (n^2) orden cuadrático y O (c^n) orden exponencial.

Reglas para determinar la **complejidad temporal** de un algoritmo:

1-El tiempo requerido para: acceder a un valor, realizar operaciones aritméticas y almacenar el resultado en memoria es constante, con un valor de O (1).

2-El tiempo requerido para la ejecución de un algoritmo de secuencia lineal es la sumatoria de los tiempos de cada instrucción.

3-El tiempo requerido para la ejecución de una instrucción condicional if C then I1 else I2 es: T = T(C) + Máximo (T (I1), T (I2)). Se aplica la regla de la suma, de modo que se calcula el tiempo de ejecución tomando el máximo de los tiempos de ejecución de cada una de las partes, (sentencias individuales) en que puede dividirse.

4-La complejidad de un ciclo está dada por el total de veces que se evalúa la condición por la complejidad de evaluar la condición más el producto del total de veces que se ejecuta el ciclo por la complejidad del cuerpo del ciclo.

5-Bloque de sentencias: Se aplica la regla de la suma.

6-Llamadas a funciones: La sentencia de código pasa a tener la misma complejidad que la función invocada. (Mairelys y Osley, 2010)

El cálculo de la complejidad según el parámetro Tiempo permite obtener una medida de la eficiencia del mismo la cual no debe ser mayor a O (n^2), pues estos algoritmos se consideran intratables o desprovistos de solución.

Métrica de Usabilidad.

La realización de esta métrica tiene como objetivo demostrar que la aplicación provee al usuario una vía sencilla de gestión del contenido de trabajo. Comprueba mediante la ejecución de las funcionalidades que la interfaz es comprensible y que el proceso de manejo del sistema no es complejo.

Para la medición se seleccionan un conjunto de tareas y usuarios midiendo el tiempo de realización de las mismas y si necesitaron asistencia o no. (Pressman, 2002)

CONCLUSIONES PARCIALES.

Con el estudio de diferentes sistemas de gestión jurídicos se demostró la necesidad del desarrollo del proyecto SGF y por ende del módulo Inspección a Locales de Detención y Centros Penitenciarios.

En este capítulo se ha introducido toda la información necesaria para la comprensión de los pasos seguidos por la directiva del proyecto Sistema de Gestión Fiscal, el arquitecto y los clientes, en cuanto a la selección de las tecnologías y herramientas a utilizar. Esta selección garantiza un desarrollo óptimo del proyecto, evita ambigüedades y proporciona la selección de herramientas funcionales, actualizadas, eficientes y multiplataforma que se integren a las necesidades actuales de la migración a software libre y de esta forma disminuir gasto de recursos. Además son herramientas que poseen una gran cantidad de información para aprendizaje y comunidades de desarrollo en Internet que permiten el ahorro de tiempo por concepto de adiestramiento.

Se selecciona a RUP como metodología de desarrollo, PHP5 como lenguaje de programación, Symfony 1.3 como framework de desarrollo, NetBeans 6.9 como IDE de desarrollo, Apache 2 como servidor web, PostgreSQL como sistema de bases de datos y Subversion1.4 como sistema de control de versiones.

También se define la estrategia a seguir para la comprobación de la calidad y funcionalidad de la solución a crear.

De esta forma queda definida la infraestructura necesaria para transitar por las fases de implementación y prueba con el fin de dar solución a la problemática y cumplir el objetivo trazado.

CAPÍTULO 2: "IMPLEMENTACIÓN DEL MÓDULO DE INSPECCIÓN A LOCALES DE DETENCIÓN Y CENTROS PENITENCIARIOS"

INTRODUCCIÓN

En este capítulo se describen los elementos más importantes en la implementación de los componentes de la solución. Se expone los requisitos funcionales detectados por los analistas el cual supone el punto de partida a la hora de conseguir un adecuado desarrollo de la aplicación. Se muestra cómo queda concebida la arquitectura a través de las posibilidades que proporcionan los marcos de trabajo utilizados y cómo influye la misma en la programación de las funcionalidades implementadas.

2.1 Requisitos.

Según el Standard Glossary of Software Engineering Terminology del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) un requisito se puede definir como una:

- ✓ Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
- ✓ Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.
- ✓ Una representación documentada de una condición o capacidad como en los casos anteriores.(Álvarez J. G., 2009)

La especificación de requisitos es un reflejo detallado de las necesidades de los clientes o usuarios del sistema y por lo tanto es donde se realiza la concepción más profunda del software que se va construir.

La especificación de requisitos abarca todas las funcionalidades, características y propiedades del Módulo Inspección a Locales de Detención y Centros Penitenciarios del Subsistema CLEP.

Requisitos Funcionales.

Los requisitos funcionales son elementos de gran importancia que deben ser cumplidos por el sistema, se refieren principalmente a los procesos de negocio que una vez terminada la aplicación esta les brindará a los usuarios de manera automatizada. La información completa se encuentra en el documento de especificación de requisitos del módulo Inspección a Locales de Detención y Centros

Penitenciarios en su versión 1.0.0.4 que se anexa a este trabajo de diploma, a continuación se muestra un ejemplo.

Crear Registro de Control de Visitas a Prisión o Campamentos.

El sistema permitirá entrar los datos a la aplicación para la creación de un nuevo Registro de Inspección a Centros Penitenciarios.

Datos	Tipo	Obligatorio
✓ Municipio.	(string)	(Obligatorio)
✓ Provincia	(string)	(Obligatorio)
✓ Departamento.	(string)	(Obligatorio)
✓ Dirección.	(string)	(Obligatorio)
✓ Número de orden	(int)	(Obligatorio)
✓ Centro visitado	(string)	(Obligatorio)
✓ Fecha de visita (Fecha inicio y Fecha culminación).	(date)	(Obligatorio)
✓ Visitas Conjuntas (Fiscal Militar, otros especialistas, Nivel Superior).	(string)	(Obligatorio)
✓ Violaciones detectadas.	(int)	(Obligatorio)
✓ Clasificación (Tratamiento, Integridad física, Servicios Médicos, Laboral, Registral, Otros).	(int)	(Obligatorio)
✓ Violaciones Resueltas.	(int)	(Obligatorio)
✓ Violaciones Pendientes.	(int)	(Obligatorio)

✓ Funcionarios puestos a disposición de la fiscalía Militar	(int)	(Obligatorio)
✓ Cantidad de Reclusos entrevistados (Individual o Grupal).	(int)	(Obligatorio)
✓ Expedientes revisados.	(int)	(Obligatorio)
✓ Reacción Fiscal (Acta, Resolución, Otra modalidad se especifica su tipo)	(string)	(Obligatorio)

Tabla 1: Requisitos Funcionales para el Caso de Uso Crear Registros.

Requisitos no Funcionales.

Los requisitos no funcionales responden principalmente a características que el software debe tener. No están relacionadas directamente con las funcionalidades derivadas del negocio a automatizar sin embargo son fundamentales para lograr el éxito de la aplicación tanto en sus propias características como en la infraestructura donde va a ser aplicada. En el documento de especificación de requisitos del módulo Inspección a Locales de Detención y Centros Penitenciarios en su versión 1.0.0.4 que se anexa a este trabajo de diploma se encuentra también información sobre los requisitos no funcionales, de estos se trae un ejemplo:

1 Usabilidad

- El software brindará una ayuda para cualquier tipo de usuario, lo que le permitirá un avance considerable en la explotación de la aplicación en todas sus funcionalidades.
- Existirán servidores locales con capacidad necesaria para el procesamiento de las solicitudes del conjunto de aplicaciones de las diferentes oficinas.
- Las aplicaciones siempre solicitarán los datos a través del servidor local.
- Desde cada servidor local se establecerá la conexión con servidores centrales para mantener la actualización de los datos en ambos sentidos.

Ilustración 2: Requisitos no funcionales de usabilidad.

2.2 Diagrama de casos de uso del sistema.

En el documento de modelo de casos de uso del sistema en su versión 1.0.0.6 anexado a este trabajo de diploma se contempla la descripción del módulo Inspección a Locales de Detención y Centros Penitenciarios en términos de casos de uso, la interacción con los actores del sistema. Este documento es de vital ayuda para la posterior implementación de las funcionalidades por ser el documento donde se reflejan de una manera más específica las características que van a ser implementadas. Los elementos más ilustrativos de este documento son los prototipos de interfaz y el diagrama de casos de usos del sistema, este último muestra la interacción entre las funcionalidades a implementar y los actores del sistema.

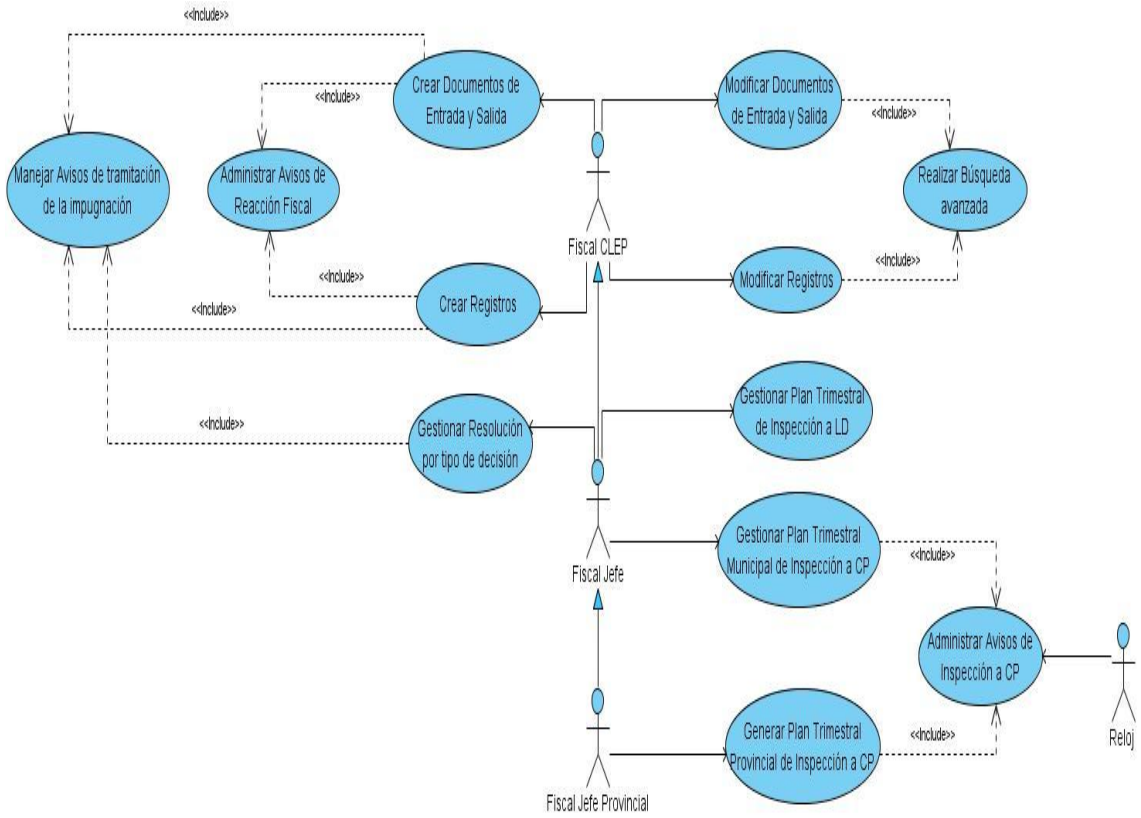

Ilustración 3: Diagrama de casos de uso del sistema.

2.3 Diagrama de clases del diseño.

Un Modelo de Diseño es una abstracción del Modelo de Implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño. Es usado como entrada esencial en las actividades relacionadas a la implementación. Representa a los casos de uso en el dominio de la solución. Este Modelo puede contener: los diagramas, las clases, paquetes, subsistemas, relaciones,

colaboraciones, atributos, las realizaciones de los casos de uso, entre otros que se puedan considerar para el sistema en desarrollo. El documento Modelo de Diseño en su versión 1.0 anexado al presente trabajo de diploma contiene la descripción de todos los diagramas de clases, diagramas de secuencia y descripción de las clases del diseño del módulo de Inspección a LD y CP. Uno de los diagramas es mostrado a continuación:

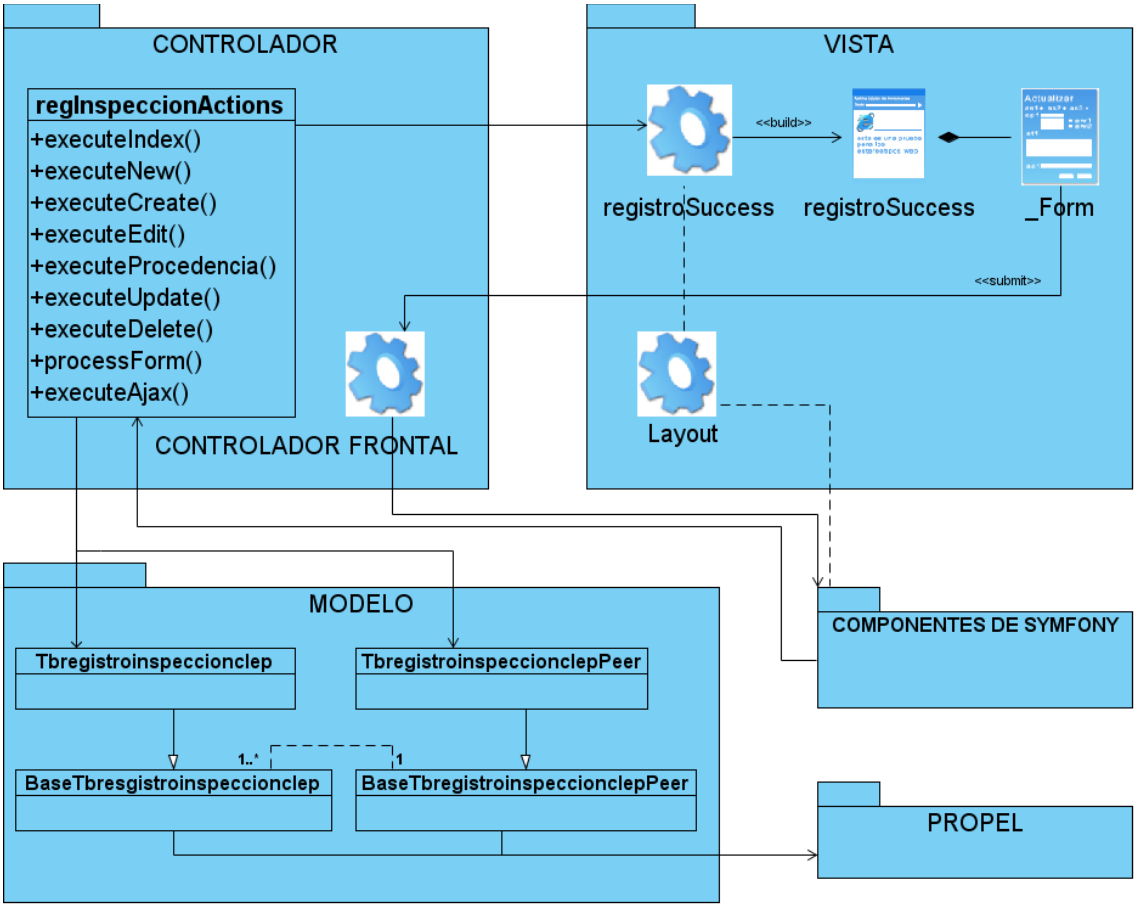

Ilustración 4: Diagrama de Clases del Caso de Uso Crear Registros.

2.4 Patrones aplicados.

Un patrón de diseño es una solución a un problema de diseño no trivial que es efectiva (ya se resolvió el problema satisfactoriamente en ocasiones anteriores) y reusable (se puede aplicar a diferentes problemas de diseño en distintas circunstancias).

Los patrones son soluciones de sentido común que deberían formar parte del conocimiento de un diseñador experto. Además facilitan la comunicación entre diseñadores, pues establecen un marco de referencia (terminología, justificación). (Lagos, 2002)

Patrones GRASP aplicados.

Experto: Este es uno de los patrones que se utilizan al trabajar con el framework de desarrollo Symfony 1.3, un ejemplo de esto es la inclusión de Propel para mapear la base de datos. Este genera las clases para la gestión de las entidades con las responsabilidades debidamente asignadas según el patrón Experto, pues cada una de estas clases cuenta con un conjunto de funcionalidades relacionadas directamente con la entidad que representan.

Creador: La clase (centroClepActions) es la que contiene las acciones del módulo Inspección a LD y CP referente a los distintos establecimientos penitenciarios y es la encargada de ejecutar las mismas. En dichas acciones se crean los objetos de las clases que representan las entidades, por lo cual la clase centroClepactions es "creador" de dichas entidades, por ejemplo, en la acción executecentroClep se evidencia el uso de este patrón mediante la creación de la instancia de la clase entidad que contienen los datos de los distintos establecimientos penitenciarios (cetroclep).

Alta cohesión: Una de las características principales de Symfony es la organización del trabajo en el mismo en cuanto a la estructura del proyecto, lo cual permite crear y trabajar con clases con una alta cohesión. Por ejemplo, las clases con sufijo Actions en su nombre contienen varias funcionalidades estrechamente relacionadas entre ellas, teniendo un sentido común y un propósito único, siendo las encargadas de controlar las acciones de las plantillas y por lo tanto pertenecen a la capa del Controlador dentro de la arquitectura Modelo-Vista-Controlador. Esto hace posible que el software sea flexible a cambios sustanciales con efecto mínimo.

Bajo acoplamiento: Este patrón está evidenciado en el framework ya que dentro de la capa modelo las clases de abstracción de datos son las más reutilizables y no tienen asociaciones con las clases de la capa vista ni con el controlador.

Controlador: Symfony es un framework basado en el patrón arquitectónico Modelo-Vista-Controlador, que define una capa específica para los controladores, que son el núcleo del mismo. Este patrón se puede observar en las clases sfFrontController, sfWebFrontController, sfContex propias de Symfony. También tiene una estructura bien organizada de sus controladores que parte desde el la página "indexSuccess.php" y se cumplimenta en la clase Actions. Cada clase de esta capa tiene su responsabilidad y es única, por ejemplo, hay controladores que se encargan de la seguridad del sistema trabajando con ficheros .yml.

Patrones GOF aplicados.

Patrón de Creación Singleton: Este patrón se aplica en la clase sfRouting, ya que es muy utilizada porque es la encargada de enrutar todas las peticiones que se hagan al registro. El Singleton sfRouting también define otros métodos que pueden llegar a ser muy útiles para el manejo manual de las rutas como son: el hasRoutes () y el getRoutesByName ().

El Singleton de contexto (que se obtiene mediante sfContext:: getInstance ()) almacena una referencia a todos los objetos que forman el núcleo de Symfony y puede ser accedido desde cualquier punto de la aplicación.(Potencier, 2007)

Patrón de Comportamiento Command: Este patrón se pone de manifiesto en el método dispatch () de la clase sfWebFrontController, que es la encargada de determinar cuál módulo y acción usar en dependencia de la petición del usuario.

Patrón Estructural Decorador: En este método de la clase abstracta sfView padre de todas las vistas, tiene cada una un decorador para permitir añadir funcionalidades a las vistas dinámicamente. Este patrón se observa en el archivo denominado layout.php que contiene el Layout (plantilla global) de todas las páginas del registro. El mismo almacena el código HTML que es común a todas las páginas del registro, para no tener que repetirlo en cada página, por lo que el Layout decora la plantilla.

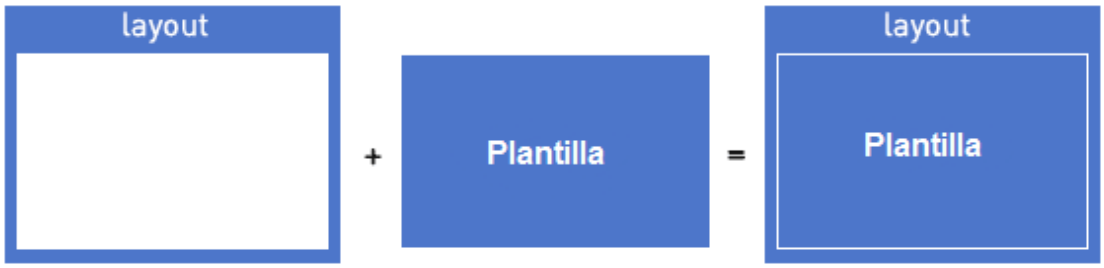

Ilustración 5: Funcionamiento del Layout.

Patrón de Arquitectura Modelo-Vista-Controlador

Para el desarrollo de aplicaciones informáticas, hay creados diferentes estilos arquitectónicos que solucionan un problema, en dependencia de las características del mismo, como ejemplo de esto se pueden señalar: el modelo-vista-controlador, arquitectura en capas entre otros. Por tanto es de vital importancia una selección óptima del mismo.

En la implementación del módulo Inspección a Locales de detención y Centros penitenciarios, se utilizó el Patrón de Arquitectura Modelo-Vista-Controlador; que se

manifiesta con el uso del Framework Symfony descrito en el capítulo anterior, para la implementación de todo el sistema propuesto a desarrollar.

Con esta selección se persigue el objetivo de utilizar la separación de las responsabilidades de cada una de las capas que lo conforman y así lograr facilidades de desarrollo. Posteriormente se comprobará durante la descripción de las clases como están quedan distribuidas dentro de la arquitectura jugando cada una un papel diferente pero relacionadas entre sí.

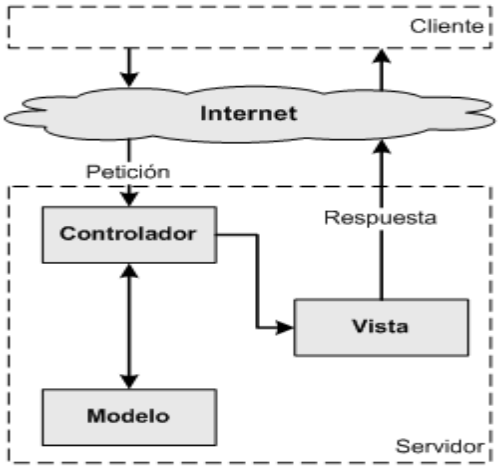

Ilustración 6: Patrón Modelo Vista Controlador.

Arquitectura Modelo Vista Controlador.

El **modelo** representa la información con la que trabaja la aplicación, es decir, la lógica del negocio.

La **vista** transforma el modelo en una página web que permite al usuario interactuar con ella.

El **controlador** se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

2.5 Diagramas de Componentes.

Un diagrama de componentes representa la separación de un sistema de software en componentes físicos (por ejemplo archivos, cabeceras, módulos, paquetes, etc.) y muestra las dependencias entre estos componentes. Muestra la organización y las dependencias entre un conjunto de componentes.

A continuación se presenta un diagrama de componentes que muestra cómo se produce la interacción del módulo Inspección con el framework Symfony.

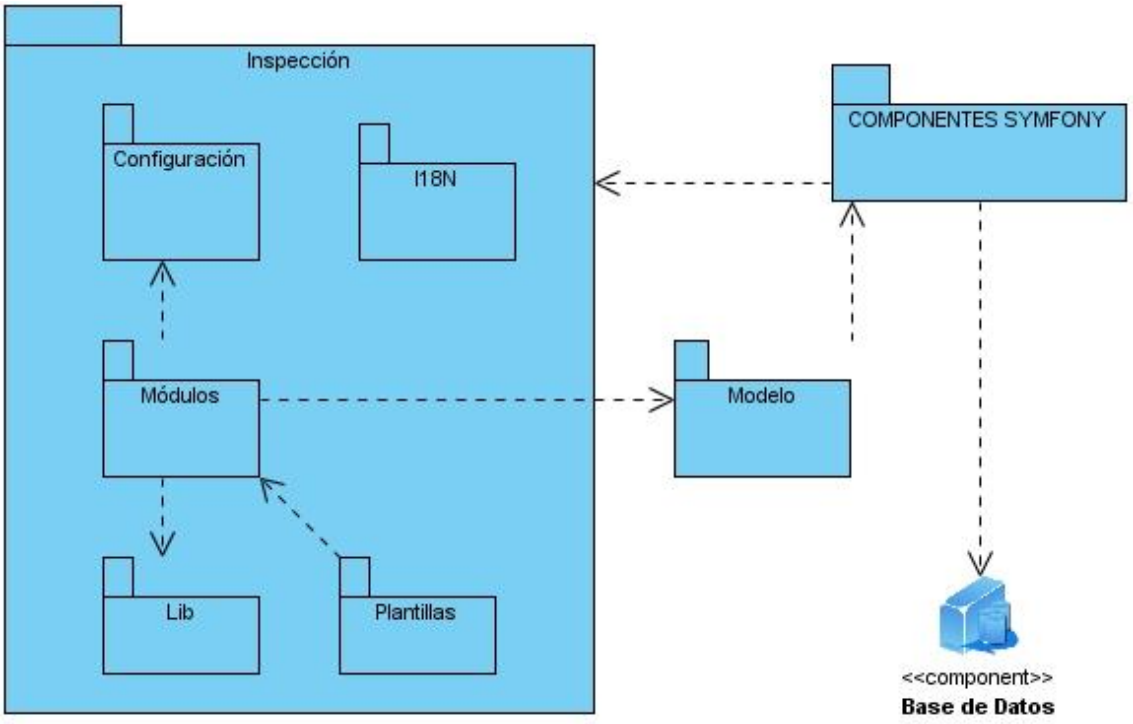

Ilustración 7: Diagrama de Componentes del Módulo Inspección.

2.6 Modelo de despliegue.

En Cuba existen: Una Fiscalía General de la República (FGR), 14 Fiscalías Provinciales, más en el municipio especial Isla de la Juventud y 149 Fiscalías Municipales. Como se refleja en el Documento de despliegue del proyecto SGF en su versión 1.0 se pretende igualar desde el punto de vista tecnológico a las fiscalías del país de forma tal que cumplan al menos con los requerimientos mínimos para desplegar la aplicación y explotarla de manera exitosa.

En las fiscalías municipales con el fin de ahorrar recursos y debido a que no se sobrecarga la aplicación por ser la instancia más específica se instalan las herramientas en un solo nodo formando un servidor de aplicación y de base de datos conjunto capaz de funcionar de forma óptima.

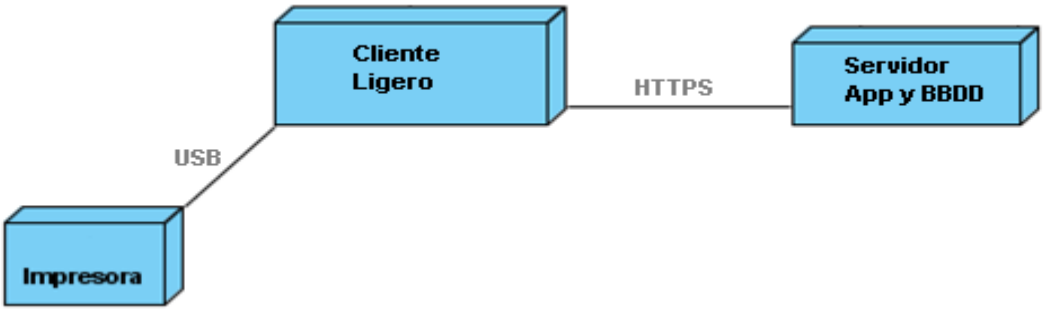

Ilustración 8: Modelo de despliegue en las fiscalías municipales.

En todos los casos la comunicación entre los nodos se realiza tratando la seguridad como elemento primario. Entre el servidor de bases de datos y el servidor de aplicación web se utiliza el protocolo Secure Socket Layer (SSL) y entre el servidor de aplicación web se utiliza el Protocolo de Transferencia de Hipertexto Seguro (HTTPS).

2.7 Buenas Prácticas de Desarrollo en PHP

Los estándares de códigos utilizados están recogidos en el PHP Development Best Practices de los autores (Naberezny, y otros, 2006) en el que se recogen directrices sobre como realizar una correcta programación ademas de vias óptimas para realizar el desarrollo colaborativo y despliegue de aplicación. Dentro de las buenas prácticas se encuentran dos categorías principales:

Las prácticas de Programación y las de Herramientas y Procesos.

En las prácticas de Programación se destacan los métodos de documentación del código como son el correcto uso de los comentarios y la estructura del código fuente; también se mencionan las recomendaciones para la revisión del código dejando bien claro la necesidad de la ejecución de pruebas unitarias y que la utilización de los metodos print() y var_dump() no deben utilizarse para las pruebas. Por último el elemento escencial que resalta dentro de las buenas prácticas es sin duda el uso de los estándares de codificación.

Algunos de estos estándares creados por la compañía desarrolladora de PHP Zend se explican a continuación.

1. La indentación debe ser a cuatro espacios sin caracteres de tabulación. Esto es debido a que ciertos IDE's de desarrollo introducen caracteres de tabulación cuando indentan un texto automáticamente. Se recomienda el uso de herramientas o editores generales como EMACS u otros.

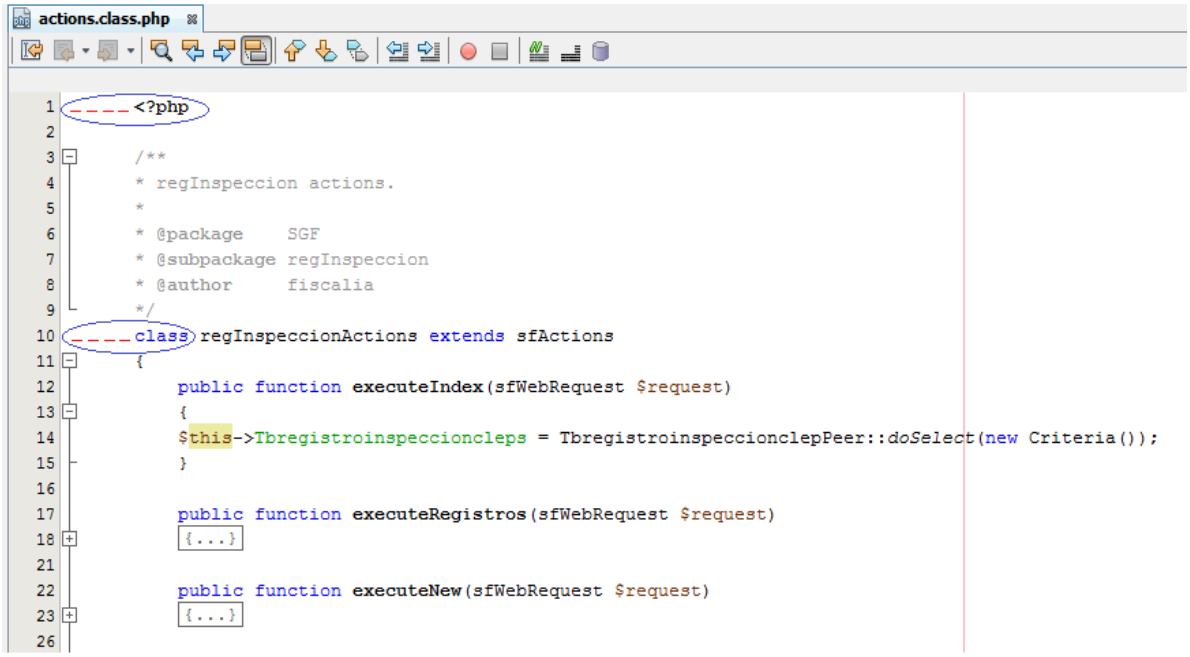

Ilustración 9: Indentación del código fuente.

2. Las estructuras de control deben tener un espacio entre el keyword de la estructura y el signo de apertura de paréntesis para distinguir entre las llamadas de las funciones y el signo de llaves debe estar sobre la línea de la estructura.

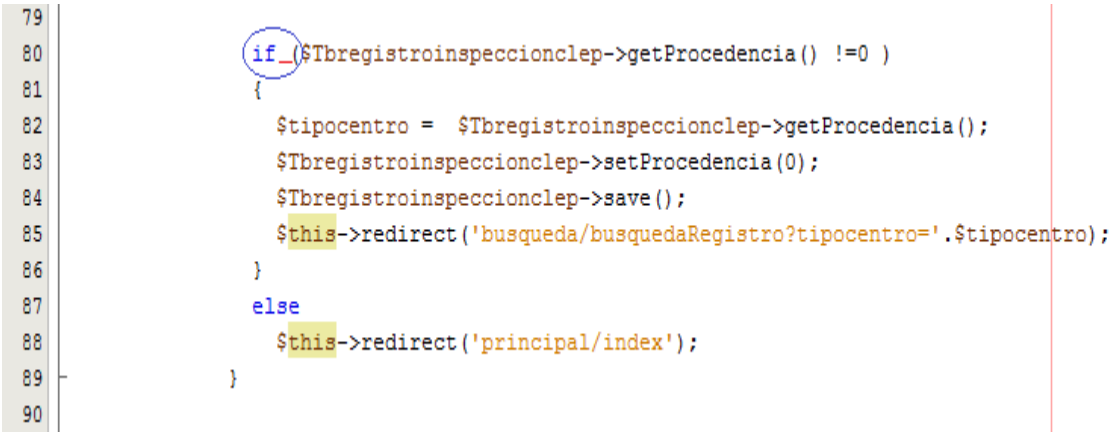

Ilustración 10: Como escribir las estructuras de control.

3. Las funciones deben ser llamadas sin espacios entre el nombre de la función, el signo de paréntesis y el primer parámetro; espacios entre cada coma por parámetro y sin espacios entre el ultimo paréntesis, el signo de paréntesis cerrado y el signo de punto y coma (;).

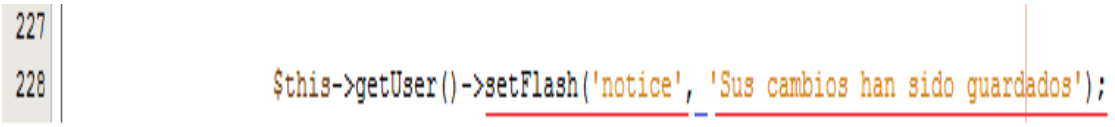

Ilustración 11: Llamada a función.

4. El estilo de los comentarios debe ser como el estilo de comentarios para C (/* */ ó //), no debe de utilizarse el estilo de comentarios de Perl (#).

```
2
3       /**
4        * regInspeccion actions.
5        *
6        * @package    SGF
7        * @subpackage regInspeccion
8        * @author     fiscalía
9        */
```

Ilustración 12: Comentarios.

5. Cuando se incluya un archivo de dependencia incondicionalmente utilice require_once y cuando sea condicionalmente, utilice include_once.

```
2
3   require 'lib/model/om/BaseTbactainspeccionclep.php';
```

Ilustración 13: Inclusión de archivos dependientes.

6. Siempre utilice las etiquetas <?php?> para abrir un bloque de código. No utilice el método de etiquetas cortas, porque esto depende de las directivas de configuración en el archivo PHP.INI y hace que el script no sea tan portable.

```
17      <?php foreach ($Tbactainspeccioncleps as $Tbactainspeccionclep): ?>
```

Ilustración 14: Uso de las etiquetas de PHP.

7. Para la nomenclatura de las clases, variables y métodos se utiliza Lower Camel Case.

```
10      class regInspeccionActions extends sfActions
11      {
```

Ilustración 15: Nombre de la Clase.

8. Los archivos con código PHP, deben de ser guardados en formato ASCII utilizando la codificación ISO-8859-1. El formato ASCII con codificación ISO-8859-1, es el formato en que se guardan los archivos de texto plano (.txt). La razón de este estándar es que determinados editores HTML (en especial Dreamweaver), agregan códigos de carácter extraño de salto de línea (como si se tratara de un

archivo binario) y esto puede ocasionar que el intérprete de PHP, encuentre problemas a la hora de leer el script.

La categoría de Herramientas y Procesos se refiere principalmente a prácticas de colaboración, control de recursos y despliegue. Los elementos principales están referidos a estrategias comunicación del grupo de desarrollo, al uso de herramientas de control de versiones y a la utilización de entornos de producción, pruebas y despliegue.

2.8 Descripción de clases.

Clases Controladoras.

La clase controladora es responsable de ejecutar una determinada lógica en función de las acciones que se producen en una aplicación. Se encargan de la captura de eventos, de la creación de entidades y permiten la comunicación con las clases del negocio. (Potencier, 2007)

Crear acta de inspección	
Nombre: actaInspeccionActions	
Nombre	**Descripción**
executeVistaActa	Crea una instancia de la clase Tbactainspeccionclep y la envía hacia la vista.
executeNew	Crea una instancia de la clase TbactainspeccionclepForm y la envía hacia la vista.
executeCreate	Crea una instancia de la clase TbactainspeccionclepForm y se hace una llamada al método processForm enviando como parámetros la instancia creada.
executeEdit	Crea una instancia de la clase TbactainspeccionclepForm a partir del idactainspeccionclep y la envía hacia la vista.
executeUpdate	Crea una instancia de la clase TbactainspeccionclepForm a partir del idactainspeccionclep y se hace una llamada al método processForm enviando

	como parámetros la instancia creada.
processForm	Guarda en la BD un acta de inspección.
executeExportarPDF	Crea un documento PDF.
executeMypdf	Crea un documento PDF

Tabla 2: Clase Controladora actaInspeccionActions.

Clases del Modelo.

Las clases entidad pertenecientes al modelo, manejan la información que poseen una larga vida y que a menudo son conceptos. También se denominan clase dominio, ya que suelen tratar con abstracciones de entidades del mundo real. Se encargan de modelar la información del sistema y el comportamiento asociado a una información.(Potencier, 2007)

Tipo de clase: Entidad
Nombre de la clase
Nviolacionclep
Tbrespuestamodalidadreaccionfiscalclep
Tbresoluciontipoclep
Tbresolucionclep
Tbregresolucionesclep
Tbregistroinspeccionclep
Tbplanprovincialclep
Tbplanobjetivosclep
Tbplanmedidasclep
Tbplaninspeccionclep
Tbotramodalidadclep
Nobjetivoclep
Tbnotificacionclep

Tbestudiolegalidadclep
Tbcomunicacionclep
Tbcentroclep
Tbautolibertadclep
Tbactainspeccionclep

Tabla 3: Clases del Modelo.

Clases de la Vista.

Permiten la interacción del usuario con el sistema mediante un formulario donde se le muestran los datos que deben ser introducidos por el mismo para realizar las diferentes acciones que serán manejadas en el controlador. (Potencier, 2007)

Crear acta de inspección	
Nombre: templates	
Nombre	**Descripción**
_form	Crea la estructura del formulario.
editSuccess	Carga el formulario con datos para realizar modificaciones.
newSuccess	Carga el formulario vacío para entrar datos.
vistaActaSuccess	Carga una vista previa del documento usando el formato real definido por la fiscalía.
indexSuccess	Obtiene los datos de un acta existente utilizando un identificador.
_origenDoc	Carga los datos referentes al origen de la información.
_mensajesDoc	Carga los mensajes de error para ser mostrados en caso necesario.

Tabla 4: Clases de la Vista del Acta resultado de la Inspección.

Clases Form.

La mayoría de sitios web requieren de la creación de formularios. Tienes que crear el código HTML, utilizar reglas de validación para los datos de todos los campos, procesar los datos enviados por los usuarios, realizar el manejo de errores y guardar la información en la base de datos. Symfony incluye un framework para la gestión de los formularios y está compuesto de tres partes:

▪ **Validación:** el subframework de validación incluye las clases necesarias para validar los datos (números enteros, cadenas de texto, direcciones de email, etc.).

▪ **Widgets:** el subframework de widgets incluye las clases que muestra el código HTML de los campos del formulario (<input>, <textarea>, <select>,...).

▪ **Formularios:** las clases de formulario representan a los formularios construidos con widgets y validadores y proporcionan métodos para facilitar la gestión del formulario. Cada campo del formulario dispone de su propio validador y su propio widget. (Potencier, 2007)

Un formulario de Symfony es una clase formada por campos de formulario. Cada campo dispone de un nombre, un widget y un validador.

```
23      $this->widgetSchema['horainicio'] = new sfWidgetFormTime(array('can_be_empty'=>true));
24      $this->validatorSchema['horainicio' ]  = new sfValidatorTime(array('required' => true),
25      array('required' => 'El campo "Hora de Inicio" es requerido.',
26      'invalid' => 'El campo "Hora de Inicio" no contiene una hora válida.'));
27
28      $this->widgetSchema['horafin'] = new sfWidgetFormTime(array('can_be_empty'=>true));
29      $this->validatorSchema['horafin' ]  = new sfValidatorTime(array('required' => true),
30      array('required' => 'El campo "Hora de Fin" es requerido.',
31      'invalid' => 'El campo "Hora de Fin" no contiene una hora válida.'));
32
```

Ilustración 16: Composición de un Formulario en Symfony.

Los widgets son definidos en el método configure() de la clase formulario, llamado automáticamente por el constructor de la clase. El método setWidget() puede declararse atendiendo a un gran número de widgets existentes como el sfWidgetFormTime() al cual se le pasa un arreglo asociativo con la configuración. En la aplicación se utilizan algunos widgets entre los que se encuentran: sfWidgetFormChoice(), sfWidgetFormDate(), sfWidgetFormInput(), sfWidgetFormInputHidden(), sfWidgetFormJQueryDate() y sfWidgetFormPropelChoice().

Los formularios pueden ser generados automáticamente a través del ORM PROPEL, ejecutando la tarea: **"propel: build-forms"** el ORM como resultado del mapeo de la base de datos obtiene los tipos de datos de cada columna de la tabla y crea una clase formulario correspondiente, esto permite enfocarse en los detalles específicos ya que se tiene una gran cantidad de trabajo realizado. A continuación se describe el funcionamiento del formulario creado para el Registro de Inspección ("TbregistroinspeccionclepForm"):

Cada vez que se accede a la página /regInspeccion/new/, se crea una nueva instancia de un formulario y se pasa a la plantilla en la acción new. Cuando el usuario envía el formulario (acción create), se asocia (mediante el método bind()) con los valores enviados por el usuario y se ejecuta la validación de los datos. Cuando el formulario está asociado, ya se puede comprobar su validez con el método isValid(). Si el formulario es válido (el método isValid () devuelve true), el registro se guarda en la base de datos ($form->save()) y se redirige al usuario a la página que previsualiza el registro y se muestra un mensaje de confirmación. Si el formulario no es válido, se vuelve a mostrar la plantilla newSuccess.php con los mismos datos que envió el usuario y con todos los mensajes de error asociados.

Clases para la construcción de archivos PDF.

Durante el proceso de Inspección se hace necesaria la impresión de documentos. Para cada uno de los documentos generados existe un formato según las normas establecidas en la Fiscalía y deben ser construidos a partir de la información que se encuentra introducida en el sistema. Para ello se utilizan las clases (HTML_PDF y html2ps) las que se encargan de construir y exportar el archivo PDF a partir de código fuente en HTML.

A continuación se muestra como queda conformada una vista previa del documento Acta Resultado de Inspección:

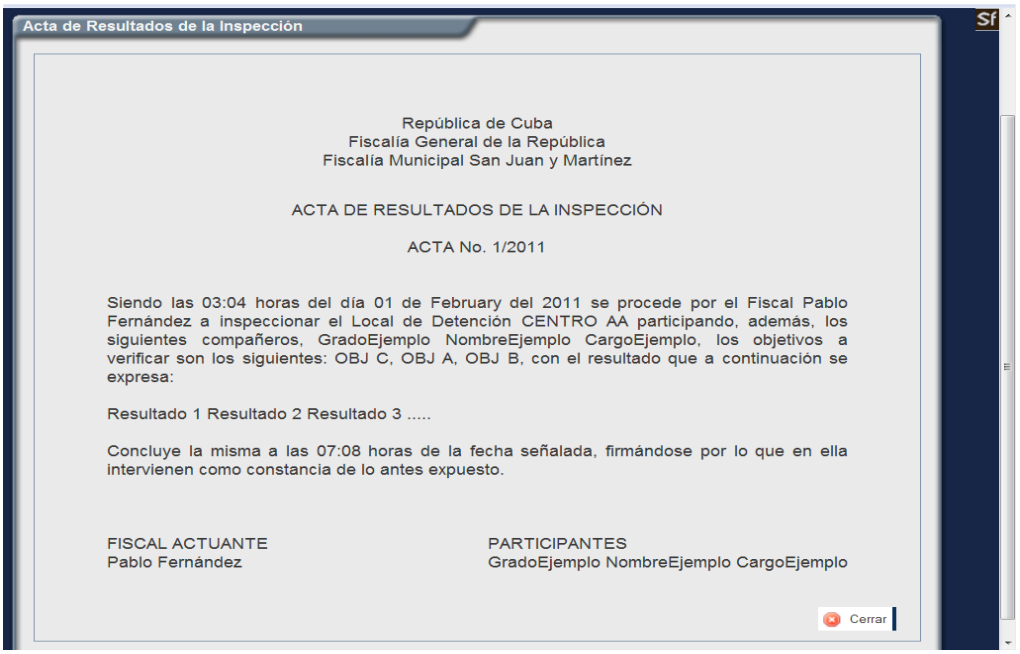

Ilustración 17: Vista Previa del Acta de resultados de la inspección.

Clases para la comunicación asíncrona con el Servidor.

Para evitar que un mismo contenido sea enviado una y otra vez al servidor de manera innecesaria cada vez que se actualicen datos de una página se utiliza AJAX (Asynchronous JavaScript And XML). Esta técnica permite actualizar el contenido de partes de la página sin tener que recargarla completamente. Envía solicitudes independientes al servidor que son transparentes al usuario.

Para hacer uso de esta técnica se empleó la biblioteca JQuery. Esta es una biblioteca o framework de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, permitiendo manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas web.

JQuery, al igual que otras librerías, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código. Es decir, con las funciones propias de esta librería se logran grandes resultados en menos tiempo y espacio. (The JQuery Project, 2010)

A continuación se muestran algunos de los métodos utilizados en la solución propuesta para garantizar la comunicación con el servidor empleando AJAX.

De esta manera se mantiene una comunicación ligera con el servidor para actualizar datos, favoreciendo el flujo de la información por el sistema y la interacción del usuario. El cliente necesita obtener siempre listas actualizadas de los distintos

establecimientos penitenciarios durante la creación de los distintos documentos y registros, para facilitar una dinámica en la selección se obtienen los establecimientos de manera independiente sin necesidad de actualizar la página.

```
154     public function executeAjax($request)
155     {
156         $this->getResponse()->setContentType('application/json');
157         $prueba = TbcentroclepPeer::ObtenerCentroClep($request->getParameter('idorigen'),
158         $request->getParameter('origen'),$request->getParameter('tipocentro'));
159         return $this->renderText(json_encode($prueba));
160     }
161 }
```

Ilustración 18: Ejecutando Ajax.

2.9 Tratamiento de errores.

El tratamiento de errores es uno de los procesos más importantes en la creación de cualquier sistema informático debido que es la cara del sistema frente a las situaciones adversas que se puedan presentar. Con un buen tratamiento de errores se puede garantizar el correcto funcionamiento del sistema así como la satisfacción de los usuarios del mismo. Una vez que el usuario introduce los datos, estos serán validados en el lado del servidor mediante archivos YAML (.yml) y en caso de haber algún error, se mostrarán mensajes que le indican donde ha ocurrido dicho error, luego, cuando los datos sean correctos, irán a la base de datos del sistema sin peligro alguno de que queden datos incompletos.

Las validaciones garantizan que se realice un correcto manejo de los errores en la aplicación. Debe realizarse en lado del cliente y en el lado del servidor para evitar la ocurrencia de acciones no esperadas.

Validaciones del lado del cliente.

El Tratamiento de errores del lado del cliente se realiza utilizando JavaScript con expresiones regulares y algunas de sus funciones principales son: garantizar los datos obligatorios, permitir solamente tipos de datos definidos, evitar la repetición de datos y garantizar la longitud de las cadenas.

```
424 function validarLetras(checkStr)
425 {
426         var checkOK = "ABCDEFGHIJKLMNÑOPQRSTUVWXYZÁÉÍÓÚ" + "abcdefghijklmnñopqrstuvwxyzáéíóú ";
427         var allValid = true;
428         for (i = 0; i < checkStr.length; i++)
429         {
430                 ch = checkStr.charAt(i);
431                 for (j = 0; j < checkOK.length; j++)
432                 if (ch == checkOK.charAt(j))
433                                 break;
434                 if (j == checkOK.length)
435                 {
436                         allValid = false;
437                         break;
438                 }
439         }
440         return allValid;
441 }
```

Ilustración 19: Función para validar letras.

Validaciones del lado del servidor.

Symfony utiliza la clase sfValidatorBase como padre de todas las clases validadoras existentes, en ella se definen los métodos y parámetros básicos para todas las validaciones. De ella heredan clases validadoras como: sfValidatorDate para las fechas, sfValidatorBoolean para los datos booleanos, sfValidatorPropelChoice para datos cargados desde la base de datos, sfValidatorChoice para validar un conjunto de opciones predefinidas, sfValidatorSchemaCompare para comparar un determinado rango de fechas o una fecha con la fecha actual, sfValidatorString para cadenas y sfValidatorInteger para números.

```
125
126         $this->validatorSchema['objetivos']= new sfValidatorPropelChoice(array
127             ('multiple' => true,'model' => 'Nobjetivoclep','required' => false));
128         $this->validatorSchema['violacion']= new sfValidatorPropelChoice(array
129             ('multiple' => true,'model' => 'Nviolacionclep','required' => false));
130
```

Ilustración 20: Validación usando sfValidatorPropelChoice.

- sfValidatorPropelChoice: Se le pasa como parámetros el nombre de la tabla de donde provienen los datos, si puede seleccionarse más de uno y si estos son requeridos o no.

También se pueden realizar funciones para validación utilizando la clase "sfValidatorError" que permite capturar un error de forma más dinámica y lanzarlo con la clase "sfValidatorErrorSchema" .

2.10 Seguridad.

El registro también restringe el acceso a ejecutar las diferentes acciones a los usuarios, estos tienen determinados privilegios y necesitan autenticarse antes de comenzar a usar el sistema informático. Antes de ser ejecutada, cada acción pasa por un filtro especial que verifica si el usuario actual tiene privilegios de acceder a la acción requerida. En Symfony, los privilegios están compuestos por dos partes:

- ✓ Las acciones seguras requieren que los usuarios estén autenticados.
- ✓ Las credenciales son privilegios de seguridad agrupados bajo un nombre y que permiten organizar la seguridad en grupos.
- ✓ Para restringir el acceso a una acción se crea y se edita un archivo de configuración YAML llamado, security.yml en el directorio config/ del módulo. En este archivo, se pueden especificar los requerimientos de seguridad que los usuarios deberán satisfacer para cada acción o para todas las acciones. (Potencier y Zaninotto., 2008)

Utilizando los archivos de configuración security.yml se realiza una gestión global de la seguridad en la aplicación a través de las credenciales.

2.11 Descripción de la aplicación.

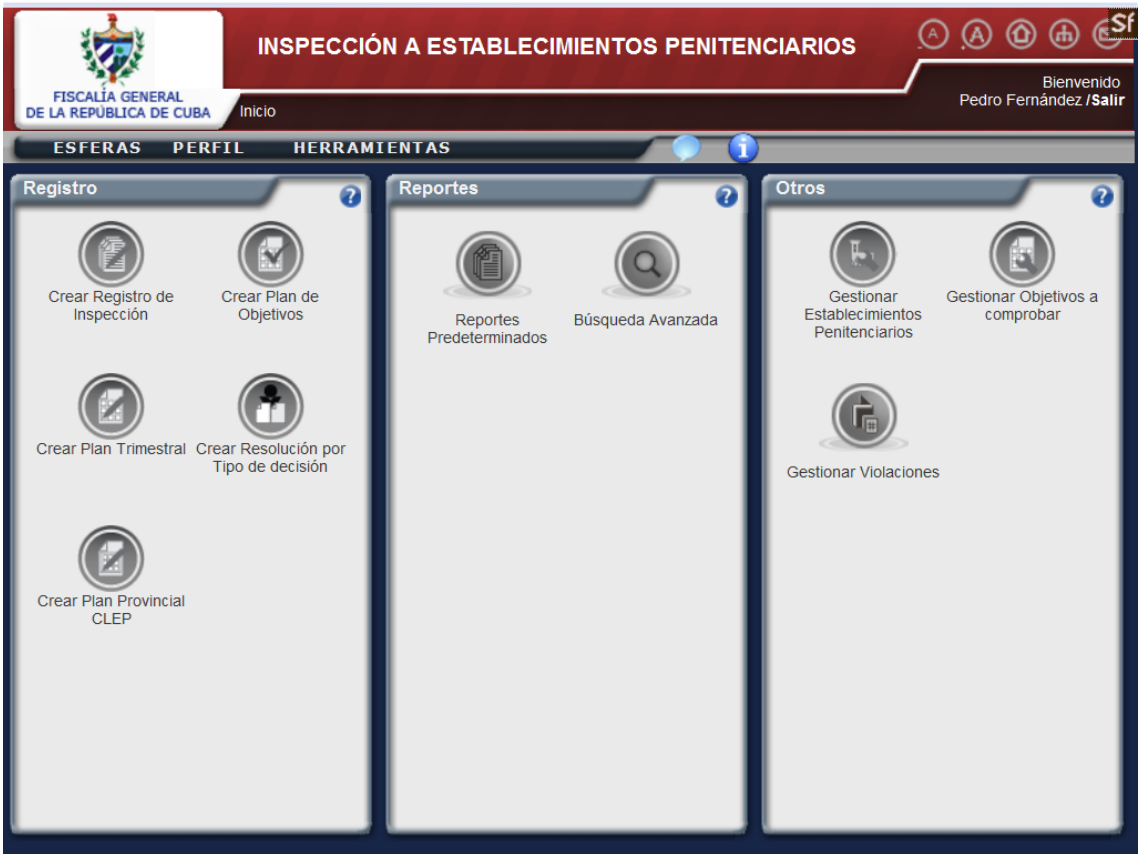

Ilustración 21: Interfaz principal del módulo Inspección a CP y LD.

La interfaz muestra todas las funciones habilitadas para un usuario con privilegios administrativos, al aplicar las credenciales de seguridad quedaran establecidos los niveles de accesos según los privilegios que tengan los usuarios. Desde el menú principal se puede acceder a todo el proceso fiscal de Inspección a CP y LD respetando siempre la secuencia de pasos lógicos que define la creación de los documentos y procesamiento de la información activándose las funcionalidades según las dependencias entre un proceso y otro. Se informatizan procesos de la esfera CLEP en todos los niveles, prueba de ello son la creación del Plan Provincial realizado solamente a nivel provincial y del cual dependen las inspecciones que se realizan a nivel municipal, esto favorece la comunicación y asignación de tareas entre la estructura de trabajo establecida en la FGR

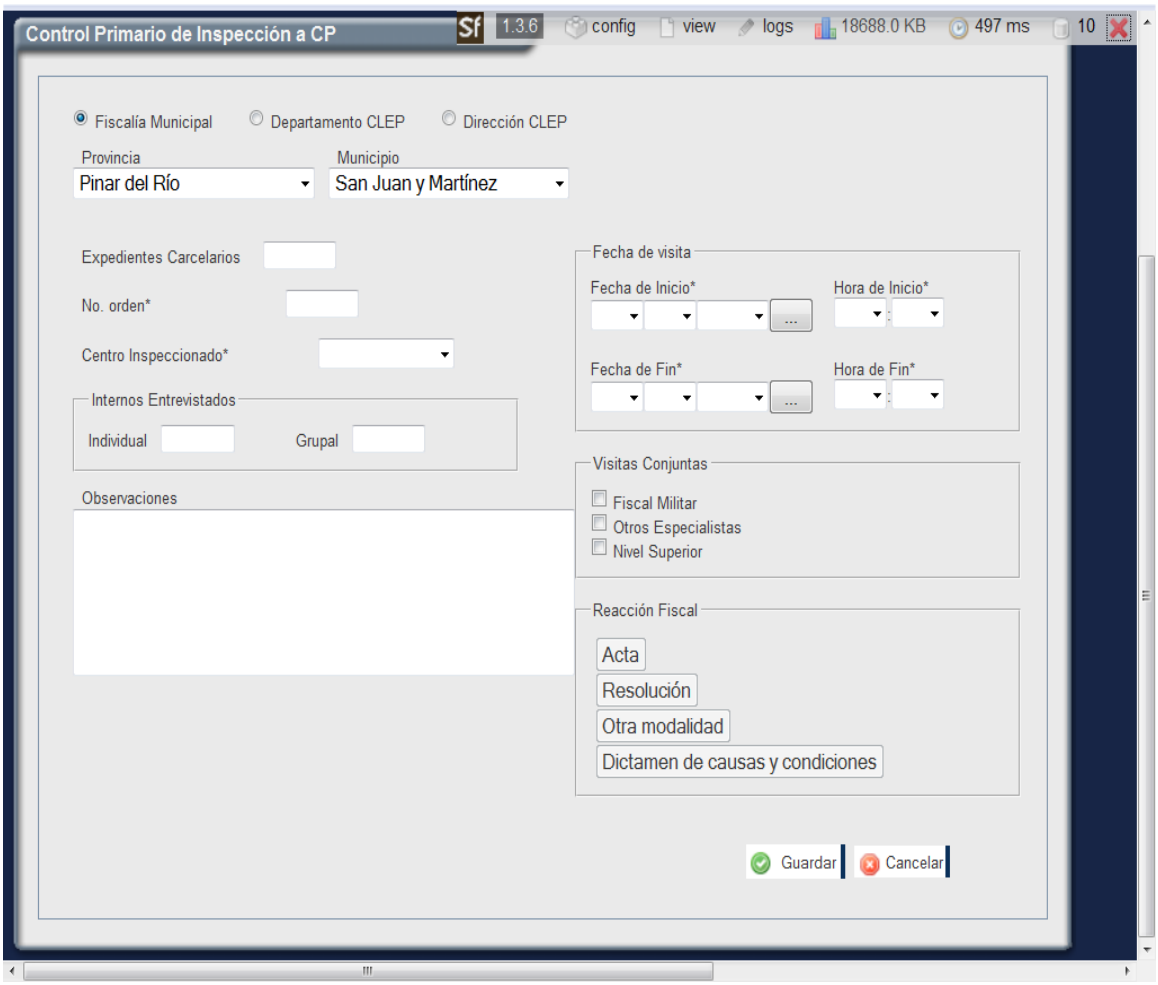

Ilustración 22: Interfaz del Registro Primario de Inspección a CP.

El proceso de creación de los registros es el eje principal del desarrollo de la Inspección, ya sea a un CP o a un LD el usuario deberá introducir los datos generales que caracterizan la inspección realizada y luego proceder a la creación de los documentos pertinentes de acuerdo al resultado de su visita. Durante la creación de estos documentos como son: El acta, la resolución, etc. El usuario debía repetir información referente a la inspección dando lugar a errores. Utilizando la aplicación, los datos repetidos se gestionan de manera automática facilitando y acelerando el trabajo, se reduce la posibilidad de cometer errores ya que al realizarse el proceso de manera automática se obliga al usuario a seguir la línea de tiempo sobre la que se basa su trabajo y se le impide la creación de documentos innecesarios. De esta forma el proceso en general experimenta una mejora y permite el desarrollo de nuevas estrategias para lograr eficiencia partiendo de las facilidades de gestión de estadísticas y creación de reportes que permite la aplicación.

CONCLUSIONES PARCIALES.

En el desarrollo de este capítulo se parte de los procesos objetos de automatización lo que proporciona una temprana idea de la complejidad de la solución y de los posibles aportes en cuanto a código fuente y almacenamiento de datos que se podría generar. Se describe la correcta aplicación de los distintos patrones y su influencia directa en el desarrollo de la solución. Todo lo anterior posibilitó la implementación de las interfaces de cada uno de los componentes con sus respectivas tablas en la base de datos del proyecto así como la completa infraestructura de almacenamiento de datos para las relaciones e información predefinida. También se tuvieron en cuenta elementos muy importantes como la seguridad y el tratamiento de errores para garantizar la estabilidad del sistema. De forma general se logró implementar el módulo Inspección a Locales de Detención y Centros Penitenciarios, perteneciente al subsistema Control de la Legalidad en Establecimientos Penitenciarios del proyecto Sistema de Gestión Fiscal.

CAPÍTULO 3: "VALIDACIÓN DE LA SOLUCIÓN"

INTRODUCCIÓN

En este capítulo se valida la solución propuesta en el capítulo anterior, de manera que se puede comprobar la eficiencia de las clases u operaciones utilizadas, por lo que se presenta la descripción de casos de prueba que verifican la validez de las funcionalidades del módulo Inspección a CP y LD, así como verificar el rendimiento interno de las mismas. También se aplican métricas orientadas a objeto para determinar en qué medida el código es reutilizable, mantenible y entendible.

3.1 Aplicación de métricas.

La aplicación de las métricas orientadas a objeto tiene como objetivo principal comprobar estadísticamente el punto de contradicción existente entre la reutilización de código y el mantenimiento y comprensión del mismo. Al ejecutar las métricas se espera la obtención de un valor de proporción medio que indique que tanto la reutilización como el mantenimiento y entendimiento del código se encuentran en un estado aceptable.

A continuación se aplican dos métricas orientadas a la herencia que serán aplicadas a muestras diferentes con el propósito de abarcar diferentes partes de la arquitectura de la aplicación.

Métrica Factor de Herencia de Métodos (MIF).

Según las investigaciones de (Abreu y Melo, 1996) proponen la aplicación de la métrica MIF como una medida de la herencia y como consecuencia, una medida del nivel de reúso. También se propone como ayuda para evaluar la cantidad de recursos necesarios a la hora de testear.

No(i)	Clase(Ci)	Md	Ma	Mi
1	actaInspeccionActions	11	11	0
2	registroInspeccionActions	10	10	0
3	planTrimestralActions	7	7	0
4	busquedaClepForm	1	74	73
5	TbActaInspeccionClepForm	1	7	6

6	TbRegistroInspeccionClepForm	2	8	6
7	TbPlanTrimestralClepForm	3	8	5
8	planMedidaActions	14	14	0
9	resolucionActions	13	13	0
Total		62	152	90

Tabla 5: Datos de la muestra para MIF

$$\mathbf{MIF} = \frac{\mathbf{90}}{\mathbf{152}} = \mathbf{0.59}$$

El valor arrojado al aplicar la métrica indica que los niveles de reutilización, de mantenimiento y de comprensión se encuentran balanceados aunque el factor de herencia se muestra por encima del 50% se asegura que el valor total del factor calculado para todo el sistema debe disminuir producto del poco uso de los numerosos métodos heredados lo que hace que el valor de la sumatoria de los métodos que pueden ser invocados (Ma) aumente disminuyendo el resultado del factor.

Métrica Factor de Herencia de Atributos AIF.

La aplicación de AIF se propone para comprobar el factor de herencia pero desde el punto de vista de los atributos del sistema.

No(i)	Clase(Ci)	Ad	Aa	Ai
1	TbAccionClepForm	2	3	1
2	TbCentroClepForm	2	3	1
3	TbDocumentoClepForm	2	3	1
4	TbPlanObjetivoClepForm	2	3	1
5	TbResolucionTipoClepForm	2	3	1
6	NobjetivoClepForm	2	3	1
7	otraModalidadActions	7	7	0
8	NvistasConjuntasActions	7	7	0

9	reportesActions	1	1	0
10	TbActaInspeccionClep	0	22	22
Total		27	55	28

Tabla 6: Datos de la muestra para AIF

$$\mathbf{AIF} = \frac{\mathbf{28}}{\mathbf{55}} = \mathbf{0.50}$$

El resultado obtenido corrobora el valor del factor de herencia para ambas métricas y la tendencia a disminuir favoreciendo al mantenimiento y comprensión del código fuente.

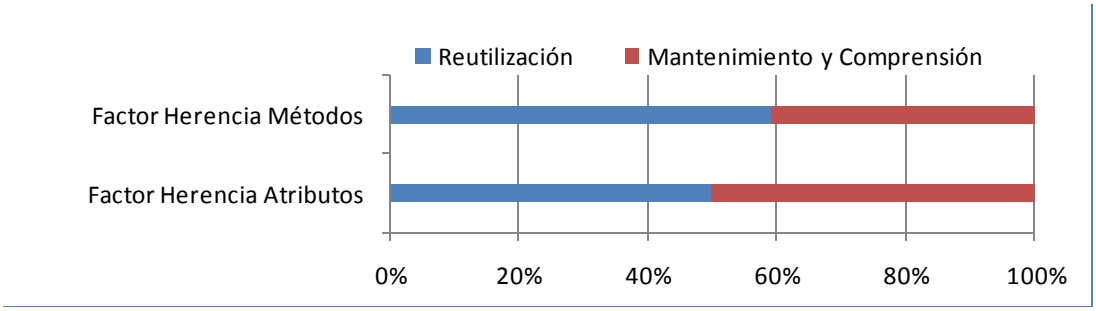

Ilustración 23: Factor herencia.

La aplicación de las métricas para el factor de la herencia arrojó un resultado satisfactorio pues a pesar de lograr las ventajas de la reutilización de código para otras etapas del proyecto SGF. También el código tiende a ser comprensible y por consiguiente mantenible lo que será útil para etapas futuras del proyecto.

Métrica de Tamaño Operacional de Clases.

Para la aplicación de esta métrica se tomará como muestra un total de 10 clases distribuidas en la capa controladora y en el modelo registrándose un total de 120 métodos con un promedio de 12 procedimientos por clase.

No	Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
1	actaInspeccionActions	11	Media	Media	Media
2	autoLibertadActions	10	Media	Media	Media
3	busquedaActions	14	Media	Media	Media
4	centroClepActions	8	Baja	Baja	Alta
5	objetivosActions	8	Baja	Baja	Alta
6	planMedidasActions	12	Media	Media	Media
7	planObjetivosActions	9	Media	Media	Media
8	planprovclepActions	10	Media	Media	Media
9	regInspeccionActions	10	Media	Media	Media

10	reportesActions	28	Alta	Alta	Baja

Tabla 7: Indicadores relacionados al número de procedimientos.

Los resultados obtenidos son reflejados en diagramas de acuerdo a los atributos de calidad para mejorar la capacidad de análisis y comprensión.

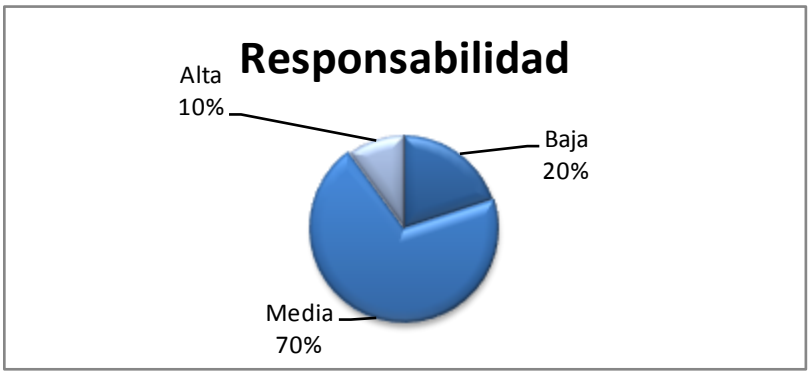

Ilustración 24: Indicador de Responsabilidad TOC.

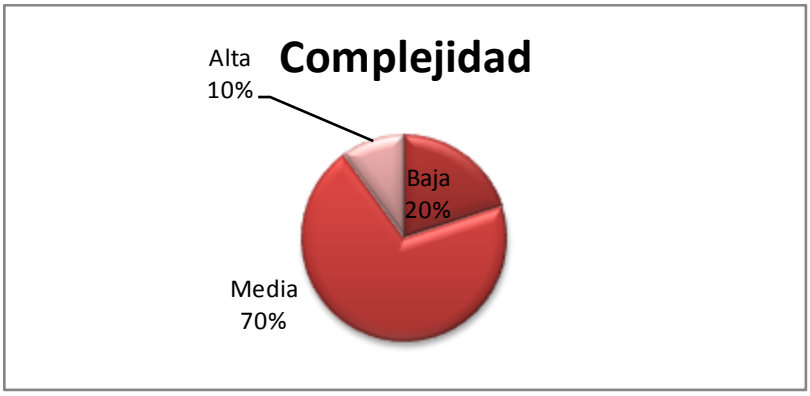

Ilustración 25: Indicador de Complejidad TOC.

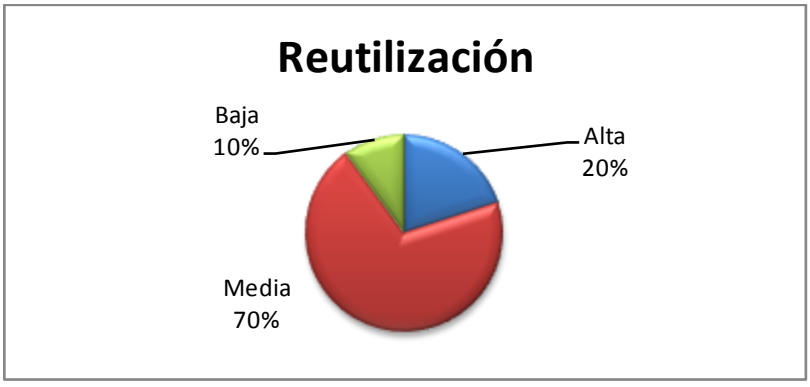

Ilustración 26: Indicador de Reutilización TOC.

El análisis de los indicadores reafirma lo planteado en la ejecución de las métricas orientadas a objetos ya que el valor de reutilización se mantiene en la media con un 70% de las clases de la muestra, lo que significa que en el sistema existe un balance en cuanto a la reutilización. Los indicadores de complejidad y responsabilidad de igual

forma se manifiestan sobre la media ya que al estar condicionado por el factor de reutilización de manera inversa al aumentar la reutilización, la complejidad y responsabilidad disminuye que en el caso del sistema es lo esperado. Los resultados reflejan la posibilidad de adaptar las clases para utilizarlas en futuras implementaciones del proyecto lo que representa una gran ventaja en cuanto al ahorro de recursos y tiempo.

Cálculo de la Complejidad Temporal.

Para la realización del cálculo se seleccionaron 10 algoritmos de diferentes clases y se calcularon sus respectivas cotas máximas para determinar la complejidad temporal.

CLASE	ALGORITMO	TEMPORAL
busquedaActions	executeBusquedaRegistro	O(1)
TbregistroinspeccionclepPeer	RegistroA	O(n)
TbregistroinspeccionclepPeer	RegistroR	O(n)
TbdatosficheroPeer	ObtenerDatosFichero	O(n)
TbplaninspeccionclepPeer	ObtenerTipoCentro	O(n)
objetivosActions	executeCreate	O(1)
TbdocumentoclepPeer	ObtenerFiscaliasProvinciales	O(n)
TbdocumentoclepPeer	ObtenerProvinciaMunicipio	O(n)
TbactainspeccionclepPeer	ObtenerNumero	O(n)
reportesActions	executeReclusoEntrevistado	O(1)

Tabla 8: Complejidad Temporal.

En la muestra analizada la utilización de los recursos se comporta de manera satisfactoria según el Tiempo, en todos los casos el orden de complejidad es lineal: O(n) o constante O(1), que comparada con los demás órdenes indica la eficiencia del mismo, pues necesita menos tiempo para ejecutarse.

Métrica de usabilidad.

Para la aplicación de la métrica de usabilidad se recogieron los datos necesarios durante el proceso de revisión en el laboratorio de calidad de la facultad durante la

primera iteración en la cual trabajaron sobre la aplicación un total de 6 analistas los que realizaron las siguientes tareas:

No	TAREA	DESCRIPCION
1	Crear un Registro de Inspección.	Ubicarse en la interfaz de creación del registro y luego introducir los datos necesarios y de forma correcta.
2	Buscar una Resolución.	Posicionarse en la interfaz correspondiente, seleccionar los parámetros y obtener resultados.
3	Generar un reporte.	Seleccionar cualquier reporte y obtener el resultado.
4	Crear una resolución por tipo de decisión.	Buscar la resolución, posicionarse en la interfaz correspondiente, agregar los datos de la resolución tipo y guardar sin la ocurrencia de errores.

Tabla 9: Tareas para medir la usabilidad.

Para la ejecución de las tareas se explicó de manera breve cómo funciona el flujo de información dentro de la aplicación, luego realizó la medición del tiempo para el completamiento de cada tarea.

El resultado del completamiento de las tareas indica que son funcionalidades fáciles de ejecutar y que el usuario se familiariza rápidamente con la aplicación. Los indicadores solo mostraron niveles de complejidad medio en poca proporción salvo la tarea de creación de resolución por tipo de decisión la cual es un poco extensa y requiere de un número de paso mayor que el resto.

Ilustración 27: Completamiento-Confianza de las tareas.

La gráfica 44 muestra que todas las tareas fueron completadas satisfactoriamente con un nivel de confianza superior al 90% lo que indica que los usuarios son capaces de manera sencilla de completar las tareas reflejando el entendimiento con las funcionalidades.

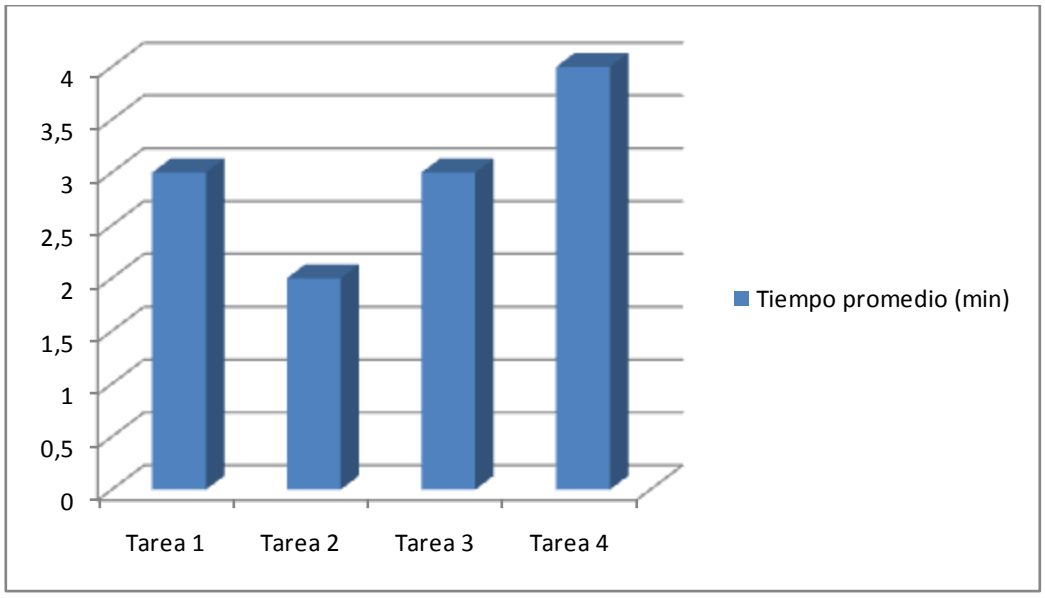

Ilustración 28: Tiempo promedio de completamiento de tareas.

La gráfica de tiempo de completamiento refleja que los tiempos que los usuarios demoraron en completar las tareas son pequeños lo que demuestra que la interfaz es comprensible y la aplicación es sencilla de utilizar.

3.2 Prueba de Caja Negra.

Las pruebas de caja negra realizadas durante el proceso de revisión del software en calidad del centro pretenden encontrar a través de los casos de prueba estos tipos de errores:

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

Descripción del Caso de Prueba "Crear Resolución por tipo de decisión".

El documento generado durante la fase de pruebas "Caso de Prueba Crear Resolución por tipo de decisión" que se adjunta a la investigación constituye una guía para llevar el control satisfactorio del proceso de creación del documento de esta

forma se puede someter a deferentes situaciones para observar el comportamiento del sistema. Una vez detectadas las anomalías se realizan iteraciones para observar el proceso de rectificación de los errores encontrados por lo que los documentos de casos de pruebas son de utilidad para no cometer errores durante el desarrollo de uno de los procesos más importantes del desarrollo del software.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Fiscalía Municipal	Campo de selección	No	
2	Departamento CLEP	Campo de selección	No	
3	Dirección CLEP	Campo de selección	No	
4	Provincia	Lista desplegable	No	
5	Municipio	Lista desplegable	No	
6	Fiscalías Provinciales	Lista desplegable	No	
7	Nombre	Campo de texto	No	No permite la entrada de números y de caracteres extraños. Solo admite letras
8	Grado	Campo de texto	No	No permite la entrada de números y de caracteres extraños. Solo admite letras
9	Cargo	Campo de texto	No	No permite la entrada de números y de caracteres extraños. Solo admite letras
10	Motivos	Campo de texto	Si	Se puede introducir letras y números
11	Fecha de impugnación	Lista desplegable	No	
12	Consideraciones	Campo de texto	Si	Se puede introducir letras y números

13	Fundamentación	Campo de texto	Si	Se puede introducir letras y números
14	A quien se remite la copia de la resolución	Campo de texto	No	No permite la entrada de números y de caracteres extraños. Solo admite letras
15	Decisión con Autoridad	Lista desplegable	No	

Tabla 10: Descripción de las variables probadas en el caso de prueba.

Utilizando la descripción del caso de prueba se realiza el flujo varias veces recreando posibles escenarios. Para todos los casos la aplicación debe ser capaz de detectar el error e informar debidamente al usuario para su posterior rectificación, de lo contrario se toma como una no conformidad y se tendrá que rectificar previo a la otra iteración de las pruebas. El proceso se repite para todos los casos de uso de la aplicación garantizando la cobertura total de las funcionalidades. Los resultados para todos los escenarios posibles con la respectiva respuesta del sistema se anexan a la investigación a través del documento de prueba para el caso de uso crear resolución por tipo de decisión.

3.3 Prueba de Caja Blanca.

Para la ejecución de la prueba de caja blanca se selecciona un algoritmo para obtener el grafo de flujo. La imagen del algoritmo se adjunta al trabajo de diploma por poseer un tamaño bastante grande como se puede observar en la ilustración 43.

Para construir el Grafo de Flujo se debe introducir la notación para la representación del flujo de control, en el cual cada nodo del grafo corresponde a una o más sentencias de código. Un grafo de flujo está formado por 3 componentes fundamentales: nodos, aristas y áreas.

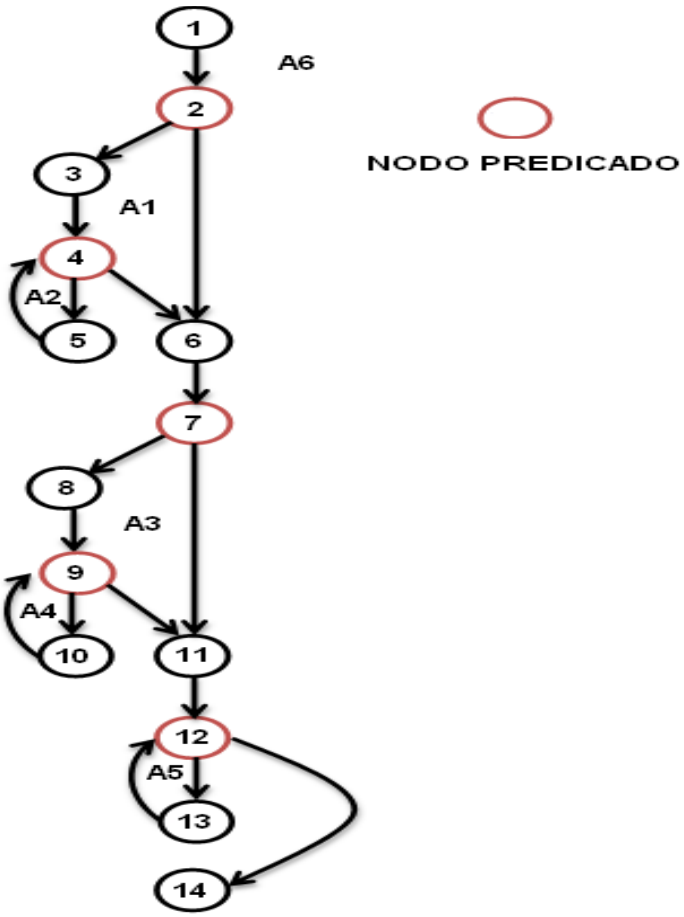

Ilustración 29: Grafo de flujo.

El siguiente paso es el cálculo de la complejidad ciclomática, para lo cual existen 3 formas:

V (G) = A

V (G) = 6

Siendo A la cantidad de áreas. Se incluye el área exterior del grafo como otra región.

V (G) = (A - N) + 2

V (G) = (A - N) + 2

V (G) = (18 - 14) + 2

V (G) = 4 + 2

V (G) = 6

Siendo A la cantidad de aristas y N la cantidad de nodos.

V (G) = P + 1

V (G) = P + 1

V (G) = 5 + 1

V (G) = 6

Siendo P la cantidad de nodos predicados.

La complejidad ciclomática del código es de 6, lo que significa que existen esa cantidad posible de caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado. Seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo.

No	Camino
1	1,2,6,7,11,12,14
2	1,2,6,7,11,12,**13**,12,14
3	1,2,**3,4**,6,7,11,12,13,12,14
4	1,2,3,4,**5**,4,6,7,11,12,13,12,14
5	1,2,6,7,**8,9**,11,12,13,12,14
6	1,2,6,7,8,9,**10**,11,12,13,12,14

Tabla 11: Caminos básicos.

Después de haber extraído los caminos básicos del flujo, se procede a ejecutar los casos de pruebas para este procedimiento, se debe realizar al menos un caso de prueba por cada camino básico. Para realizarlos es necesario cumplir con las siguientes exigencias:

- **Descripción:** Se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.
- **Condición de ejecución:** Se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.
- **Entrada:** Se muestran los parámetros que entran al procedimiento.
- **Resultados Esperados:** Se expone resultado que se espera que devuelva el procedimiento.

No	Descripción	Condición	Entrada	Resultado

1	Se procede a buscar un registro de Inspección.	El usuario introduce criterio inexistente de búsqueda.	idcentroclep=40 tipocentro=1	Satisfactorio
2	Se procede a buscar un registro de Inspección.	El usuario introduce criterio existente de búsqueda.	idcentroclep=33 tipocentro=1	Satisfactorio
3	Se procede a buscar un registro de Inspección.	El usuario introduce criterio existente de búsqueda.	idcentroclep=33 tipocentro=1 violaciones[]=1	Satisfactorio
4	Se procede a buscar un registro de Inspección.	El usuario introduce criterio existente de búsqueda.	idcentroclep=33 tipocentro=1 violaciones[]=1 violaciones[]=14	Satisfactorio
5	Se procede a buscar un registro de Inspección.	El usuario introduce criterio existente de búsqueda.	idcentroclep=33 tipocentro=1 objetivos[]=30	Satisfactorio
6	Se procede a buscar un registro de Inspección.	El usuario introduce criterio existente de búsqueda.	idcentroclep=33 tipocentro=1 objetivos[]=29 objetivos[]=30	Satisfactorio

Tabla 12: Casos de prueba.

Los casos de prueba se realizaron usando PHPUnit. Para ello se construyó un método para cada uno de los casos de prueba.

```
public function testCamino1()
{
  // #1
  $prueba = TbregistroinspeccionclepPeer::RegistroA(40, null, null,null, null, 1, null, false,
        null, null, null, null, null, null, null, null,null );
  //$t->is($prueba, null,'#1');
  $this->assertEquals($prueba, null);
}

public function testCamino2()
{
  //#2
  $prueba = TbregistroinspeccionclepPeer::RegistroA(33, null, null,null, null, 1, null, false,
        null, null, null, null, null, null, null, null,null );
  //$t->is($prueba[0]['nombre'], 'CENTRO AAA','#2');
  $this->assertEquals($prueba[0]['nombre'], 'CENTRO AAA');
}

public function testCamino3()
{
  //#3
  $violaciones[] = 1;
  $prueba = TbregistroinspeccionclepPeer::RegistroA(33, null, null,null, null, 1, null, false,
        null, null, null, null, null, null, null, null,$violaciones );
  //$t->is($prueba[0]['nombre'], 'CENTRO AAA','#3');
  $this->assertEquals($prueba[0]['nombre'], 'CENTRO AAA');
}

public function testCamino4()
{
  //#4
  $violaciones[] = 1;
  $violaciones[] = 14;
  $prueba = TbregistroinspeccionclepPeer::RegistroA(33, null, null,null, null, 1, null, false,
        null, null, null, null, null, null, null, null,$violaciones);
  //$t->is($prueba[0]['nombre'], 'CENTRO AAA','#4');
  $this->assertEquals($prueba[0]['nombre'], 'CENTRO AAA');
}

public function testCamino5()
{
  //#5
  $objetivos[] = 30;
  $prueba = TbregistroinspeccionclepPeer::RegistroA(33, null, null,null, null, 1, null, false,
        null, null, null, null, null, null, null, $objetivos,null );
  //$t->is($prueba[0]['nombre'], 'CENTRO AAA','#5');
  $this->assertEquals($prueba[0]['nombre'], 'CENTRO AAA');
}

public function testCamino6()
{
  //#6
  $objetivos[] = 29;
  $objetivos[] = 30;
  $prueba = TbregistroinspeccionclepPeer::RegistroA(33, null, null,null, null, 1, null, false,
        null, null, null, null, null, null, null, $objetivos,null );
  //$t->is($prueba[0]['nombre'], 'CENTRO AAA','#6');
  $this->assertEquals($prueba[0]['nombre'], 'CENTRO AAA');
}
```

Ilustración 30: Funciones para los casos de prueba.

La prueba de la caja blanca demostró que el estado real del software coincide con el esperado, comprobándose a través de los casos de prueba desarrollados con la herramienta PHPUnit donde la ejecución de las condiciones y los bucles tuvieron resultados satisfactorios.

3.4 Pruebas de Usuario.

Las pruebas de usuario fueron realizadas durante la ejecución del primer laboratorio de aceptación. Preparado por la dirección del proyecto en conjunto con la Fiscalía General. Durante una semana se realizó la revisión de todas las funcionalidades comparando estas con la realidad desde la perspectiva de los futuros usuarios de la aplicación los que se sintieron complacidos por las posibilidades que brinda la aplicación y las diferentes herramientas que posee para mejorar el trabajo de los fiscales. Durante el desarrollo del laboratorio los señalamientos fueron solucionados de forma casi instantánea y los fiscales responsables reconocieron la capacidad de automatización que propone la aplicación con respecto al trabajo de Inspección a LD y CP.

3.5 Pruebas de Carga y Stress.

Para ejecutar las pruebas se seleccionó un servidor con las siguientes características:

- **Hardware**: Dual Core, 3 Gb RAM, SO Ubuntu 10.10 en una partición de 30Gb, HDD 320 Gb.
- **Software**: Gestor de Base de Datos: Postgres SOL 8.4. Servidor de Aplicaciones: Apache 2. Máquina Virtual: JDK 6.
- **LAN** a 100 Mbps de velocidad.

Se definió un máximo de 100 Hilos (cantidad de usuarios que simularon la prueba.), un valor de 1 como Período de Subida en segundos (especifica el período intermedio en segundos en los cuales va a ir iniciándose cada uno de los usuarios) y también el valor 1 como Contador del Bucle, por el cual se pudo realizar la simulación una sola vez. Estos valores aportaron como resultados que se realizaron un total de 100 muestras por prueba, ejecutándose 100 veces, a razón de 100 usuarios por vez. Los resultados arrojados al culminar las pruebas fueron los siguientes:

Criterio	Resultado	Significado
Muestras	1000	Cantidad de páginas que simulan la cantidad de usuarios que están interactuando con el sistema desde la misma URL.
Media	7827	Media de páginas que se cargaron de manera satisfactoria.

Mediana	3523 ms	Tiempo promedio que han tardado en cargarse las páginas.
Mínimo	192 ms	Tiempo mínimo que ha demorado en cargarse una página.
Máximo	38862 ms	Tiempo Máximo que ha tardado en cargarse una página.
Línea 90%	25922 ms	Tiempo en que el 90 por ciento de las páginas se cargaron de manera satisfactoria.
% Error	0%	Por ciento de error de las páginas que no se llegaron a cargar de manera satisfactoria.
Kb/seg	31.87 Kb/seg	Velocidad de carga de las páginas.
Tiempo de Respuesta	11.3 seg	Total del tiempo que demoró en cargarse la cantidad de muestras de esa prueba.

Tabla 13: Resultados de la Prueba de Stress.

Estos indicadores permitieron llegar a la conclusión de que las pruebas fueron satisfactorias bajo el entorno de prueba creado. El tiempo total de carga de las páginas es de 11.3 segundos para un promedio de 1.1 segundos el cual es mucho menor que el requisito no funcional de 3 segundos de tiempo de respuesta establecido. Esto demuestra que el rendimiento del sistema una vez desplegado el mismo será aún mejor debido a las características superiores del hardware de los servidores donde se instalará la aplicación en las fiscalías.

CONCLUSIONES PARCIALES.

En el desarrollo de este capítulo se describió como la aplicación web estuvo sometida a una amplia gama de pruebas realizadas con el objetivo de garantizar la calidad y funcionalidad del sistema. Se realizaron un conjunto de métricas relacionadas con el código fuente que permitió valorar la reutilización, mantenimiento, complejidad y comprensión de la implementación realizada. Por otra parte se realizaron cálculos de complejidad temporal y espacial a varios algoritmos los que demostraron poseer un tiempo de ejecución y tamaño en memoria pequeños siendo estos resultados muy positivos. También se aplicó una métrica de usabilidad donde a través de la medición del tiempo de ejecución de varias tareas del sistema se valora la complejidad del mismo y si agiliza el trabajo de los futuros usuarios. Se realizaron pruebas unitarias al sistema, las pruebas de caja negra facilitaron la corrección de errores en validación de datos y revisión ortográfica debido a que el resto de los indicadores no presentaron problemas. Otro elemento que arrojo resultados satisfactorios fueron las pruebas de usuario, realizadas por los fiscales en la que se demostró que la aplicación proporcionaba ventajas para el trabajo de Inspección a CP y LD .En cuanto a la realización de las pruebas de caja blanca, los algoritmos revisados mostraron estar correctamente construidos y el acceso a datos demostró un funcionamiento óptimo. Por último la ejecución de las pruebas de carga y stress arrojaron que el sistema es muy ligero en cuanto a rendimiento registrándose tiempos de carga muy rápidos y cero ocurrencia de errores.

CONCLUSIONES GENERALES

A partir del estudio y refinamiento de los artefactos del análisis y el diseño, de la investigación sobre los diferentes softwares de gestión fiscal existentes, de la selección y aprendizaje de las herramientas, lenguajes de programación y metodología de desarrollo, se logró realizar de forma satisfactoria la implementación del módulo de Inspección a LD y CP logrando la automatización del proceso. Además la realización de diferentes pruebas demostró que el sistema cumple con las necesidades de los clientes y con las pautas del desarrollo de software. La construcción de este módulo no solo facilita el trabajo de los fiscales sino que permite gestionar de forma dinámica y personalizada las estadísticas necesarias para la realización de informes y contribuye a la toma de decisiones por parte de los directivos de la fiscalía. Sin duda el cumplimiento del objetivo de esta investigación representa un aporte significativo para la FGR y para el proceso de informatización que se realiza en Cuba.

RECOMENDACIONES

1. Realizar un análisis más profundo que permita incorporar nuevas funcionalidades al software.
2. Realizar una revisión con el objetivo de incorporar la gestión de nuevos reportes.
3. Incorporar estrategias de Inteligencia Artificial que faciliten la toma de decisiones sobre la realización de un Acta o una Resolución así como para otras funcionalidades.
4. Luego de poner en funcionamiento el sistema retroalimentarse de las experiencias de los usuarios y mejorar aún más la Arquitectura de Información realizada.

BIBLIOGRAFÍA

Abreu y Melo. 1996. *Evaluating the impact of Object-Oriented Design on Software Quality.* Berlin : s.n., 1996.

Álvarez, José García Fanjul y Claudio de la Riva. 2009. [Online] 2009. [Cited: marzo 8, 2010.] http://www.di.uniovi.es/~claudio/isoft/recursos/Requisitos.pdf.

Arpug. 2001. Arpug. [Online] 2001. [Cited: diciembre 21, 2010.] http://www.arpug.com.ar/trac/wiki/tutorial.html.

Barrios, Johanna Rojas y Emilio. 2007. Grupo ARQUISOFT. [Online] 2007. [Cited: abril 26, 2010.] http://www.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node28.html.

BRiNDYS Software. 1996. GEDEX. [Online] BRiNDYS Software, 1996. [Cited: 12 21, 2010.] http://www.brindys.com/gedex/sammenu.html?ibl=es-mx..

Caja Blanca. 2008. [Online] 2008. [Cited: diciembre 21, 2010.] http://www.lcc.uma.es/~av/Libro/CAP1.pdf.

Collado, Manuel. 2003. [Online] marzo 2003. [Cited: abril 26, 2010.] http://www.lml.ls.fi.upm.es/ftp/ed2/0203/Apuntes/pruebas.ppt.

Criado, Alfonso Blanco. 2008. [Online] febrero 04, 2008. [Cited: febrero 10, 2010.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=xampp.

D'Onofrio, Diego Lucio. 2009. [Online] marzo 2009. [Cited: febrero 10, 2010.] http://www.elguille.info/Clipper/probando.htm.

El código K. 20009. El código K. [Online] 20009. [Cited: 12 22, 2010.] http://www.elcodigok.com.ar/2009/10/netbeans-6-8-soporta-symfony/.

Equipo de desarrollo de PostgreSQL. 1999. *Tutorial de PostgreSQL.* Berkeley, California : Thomas Lockhart, 1999.

Fernández, Juan Antonio Vega. 1999. [Online] 1999. [Cited: abril 26, 2010.] www.rogeliodavila.com/tcs/.../Parte_16_TestWhite.ppt.

Fernando. 2006. lacoctelera. [Online] diciembre 1, 2006. [Cited: abril 26, 2010.] http://www.inwebwetrust.net/post/2006/12/01/testeando-varios-valores-un-atributo-un-test-unidad.

García, Yunesky del Rio. 2009. *Implementación del módulo Indice de Peligrosidad Pre-Delictiva del Proyecto Sistema de Gestión Fiscal.* La Habana : UCI, 2009.

Hidalgo, Daira Figueroa, Ortiz, Yurisbel Vega and Fernández, Vladimir Martell. 2010.. *Propuesta de diseño para proyectos informáticos que utilizan Symfony como Framework.* La Habana : Universidad de las Ciencias Informáticas. , 2010.

IDE php. 2007. tufuncion. [Online] 2007. [Cited: febrero 10, 2010.] http://www.tufuncion.com/ide-php.

Jacobson y Rumbaugh. 2000. *El Proceso Unificado de Desarrollo de Software.* s.l. : Pearson Education SA, 2000.

Javascript. 2005. PubliSpain. [Online] 2005. [Cited: 12 21, 2010.] http://www.publispain.com/supertutoriales/Javascript/Intro.doc.

Lago, Ramiro. 2007. [Online] Abril 2007. [Cited: febrero 10, 2010.] http://www.proactiva-calidad.com/java/patrones/index.html#algunos_patrones.

Lagos, Manuel Torres. 2002. Introducción al diseño con patrones. . [Online] 2002. [Cited: 1 22, 2011.] http://www.elrincondelprogramador.com/default.asp?id=29&pag=articulos/leer.asp.

Leopoldo, Carlos. 2008. [Online] septiembre 19, 2008. [Cited: febrero 10, 2010.] http://techtastico.com/post/7-sistemas-control-versiones/.

Mairelys y Osley. 2010. *Diseño e implementación del componente Ajuste al Costo del.* La Habana : UCI, 2010.

Muñoz y Rodríguez. 2010. *Nodo Virtual de Procesos Versión 2.0.* La Habana : UCI, 2010.

Naberezny, Mike and O'Phinney, Matthew Weier . 2006. *Best Practices of PHP Development.* s.l. : Zend Technologies, 2006.

Peláez, Juan. 2009. [Online] mayo 29, 2009. [Cited: febrero 10, 2010.] http://geeks.ms/blogs/jkpelaez/archive/2009/05/29/arquitectura-basada-en-capas.aspx.

PHP. 2001. PHP. [Online] 2001. [Cited: 12 23, 2010.] http://www.php.net/.

phpmvc. [Online] [Cited: febrero 10, 2010.] http://www.phpmvc.net/.

Postigo, Javier Vidal. 2008. [Online] febrero 11, 2008. [Cited: febrero 10, 2010.] http://javiervidal.net/los-servidores-web-mas-utilizados/.

Potencier y Zaninotto. 2008. *Symfony, la guía definitiva. .* Francia : s.n., 2008.

Potencier, Fabien. 2007. Guía definitiva de Symfony. [Online] octubre 21, 2007. [Cited: marzo 8, 2010.] www.librosweb.es.

Pressman, R.S. 2002. *Ingenieria del Software.Un enfoque práctico.* 2002.

Programación Estructurada. 2004. Lenguajes de Programación. [Online] 2004. [Cited: 12 20, 2010.] http://www.lenguajes-de-programacion.com/programacion-estructurada.shtml.

Programacion Orientada a Objetos. 2004. lenguajes-de-programacion. [Online] 2004. [Cited: 12 20, 2010.] http://www.lenguajes-de-programacion.com/programacion-orientada-objetos.shtml.

Rodríguez y Harrison. 2007. *MEDICIÓN EN LA ORIENTACIÓN A OBJETOS.* Reading, UK : School of Computer Science, Cybernetics & Electronic Engineering, 2007.

Ruíz y Morejon. 2010. *Análisis y Diseño del Módulo Inspeccion a Locales de Detencion y Centros Penitenciaros de el proyecto Sistema de Gestión Fiscal.* La Habana : UCI, 2010.

Sistemas Jurídicos SLR. 1999. Lex Doctor Gestión Jurídica. [Online] Sistemas Jurídicos SLR, 1999. [Cited: 10 20, 2010.] http://www.lex-doctor.com.

Subversion. 2005. Guía de Subversion: ¿Qué es? ¿Qué hace? [Online] 2005. [Cited: 12 22, 2010.] http://www.osmosislatina.com/subversion/basico.htm. .

Sun Microsystem. 2006. Netbeans. [Online] Sun Microsystem, 2006. [Cited: 12 21, 2010.] http:\\www.netbeans.org\index_es.html.

The JQuery Project. 2010. JQuery. [Online] 2010. [Cited: 1 26, 2011.] www.JQuery.com.

Torre, Aníbal de la. [Online] [Cited: febrero 10, 2010.] http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html.

Universidad Politécnica de Valencia. 2004. Rational Unified Procces. [Online] Departamento de Sistemas Informáticos y Computación, 2004. [Cited: 12 10, 2010.] https://pid.dsic.upv.es/C1/Material/Documentos%20Disponibles/Introducci%C3%B3n%20a%20RUP.doc.

Victoria. 2009. [Online] febrero 25, 2009. [Cited: febrero 10, 2010.] http://www.definicionabc.com/tecnologia/mysql.php.

Vitoria-Gasteiz. [Online] [Cited: febrero 10, 2010.] http://www.google.com.cu/url?sa=t&source=web&ct=res&cd=10&ved=0CB0QFjAJ&url=http%3A%2F%2Fwww.ejie.net%2Fdocumentos%2FHerramientas%2FTortoiseSVN.%2520Manual%2520de%2520usuario.doc&rct=j&q=TortoiseSVN+%2B+concepto&ei=9pFzS4PjN6imtgf6gsWDCg&usg=AFQjCNH1uMJCj.

Watson y McCabbe. 1996. *Structured testing: a testing methology using the cyclomatic complexity metric.* s.l. : NIST, 1996. 500-225.

Zaninotto, Fabien Potencier y François. 2007. [Online] octubre 21, 2007. [Cited: febrero 10, 2010.] http://www.symfony-project.org/.

Zend Studio. 2006. Zend Framework. [Online] Zend Studio, 2006. [Cited: febrero 10, 2010.] http://framework.zend.com.

GLOSARIO DE TÉRMINOS

IDE: es un programa informático compuesto por un conjunto de herramientas de programación para la creación, diseño y pruebas de programas.
HTML: es el lenguaje de marcado predominante para la elaboración de páginas web.

HTTP: Hypertext Transfer Protocol, en español protocolo de transferencia de hipertexto, es el protocolo usado en cada transacción de la Web.

Sun: Sun Microsystems es una empresa informática fabricante de semiconductores y software. Recientemente comprada por Oracle, Sun era la desarrolladora del Lenguaje Java entre otros productos.

GPL: GNU General Public License, en español Licencia Pública General de GNU, es una licencia creada por la Free Software Foundation en 1989 y está orientada principalmente a proteger la libre distribución, modificación y uso de software.
BSD: es la licencia de software otorgada principalmente para los sistemas BSD (*Berkeley Software Distribution*). Es una licencia de software libre permisiva como la licencia de OpenSSL o la MIT License. Esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre.
MIT: es una de tantas licencias de software que ha empleado el Instituto Tecnológico de Massachusetts (MIT, *Massachusetts Institute of Technology*) a lo largo de su historia, y quizás debería llamarse más correctamente **licencia X11**, ya que es la licencia que llevaba este software de muestra de la información de manera gráfica X Windows System originario del MIT en los años 1980. Pero ya sea como MIT o X11, su texto es idéntico.
Framework: herramienta que simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, un framework proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, un framework facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas.
Lenguajes: Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una computadora. Consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Un lenguaje de programación permite a uno o más

programadores especificar de manera precisa: sobre qué datos una computadora debe operar, cómo deben ser estos almacenados, transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural.

Las pruebas: es una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requisitos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente.