Universidad de las Ciencias Informáticas

Facultad 7

“Componente para la configuración visual de los servicios de integración en el marco de trabajo Sauxe”

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

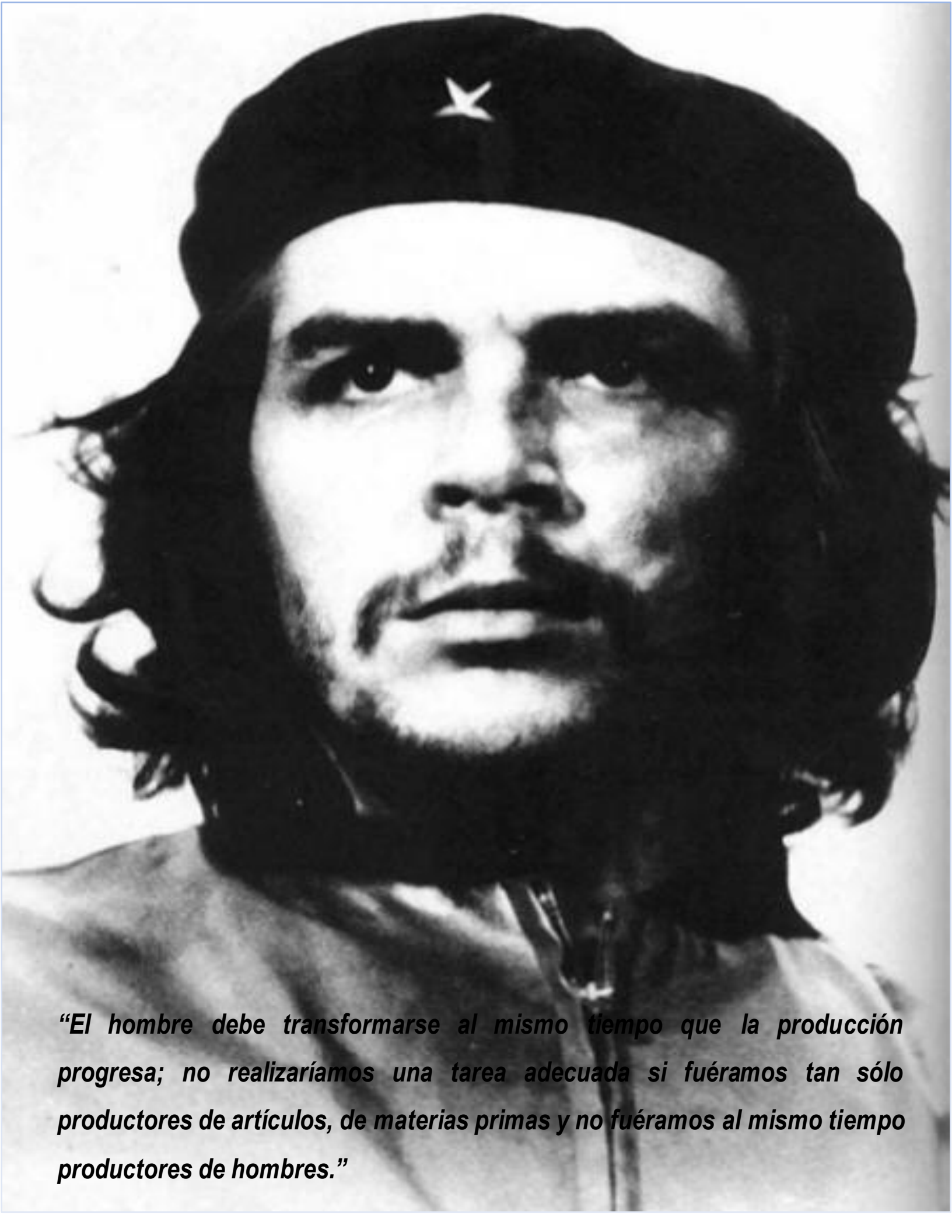
Autor(es): René Hernández Morera

Tutor(es): Ing. Pedro Manuel Nogales Cobas

Ing. Javier Ruiz Durán

La Habana, junio del 2011

"Año 53 de la Revolución"



“El hombre debe transformarse al mismo tiempo que la producción progresa; no realizaríamos una tarea adecuada si fuéramos tan sólo productores de artículos, de materias primas y no fuéramos al mismo tiempo productores de hombres.”

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo al departamento de tecnología de CEIGE de la Universidad de las Ciencias Informáticas; así como a dicho centro 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_____.

_____

René Hernández Morera.

Firma del Autor

_____

Ing. Pedro Manuel Nogales Cobas

Firma del Tutor

_____

Ing. Javier Ruiz Durán

Firma del Tutor

Datos de contacto

Tutor: Ing. Pedro Manuel Nogales Cobas

CEIGE, Facultad 3, Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: pmnogales@uci.cu

Tutor: Ing. Javier Ruiz Durán

CEIGE, Facultad 3, Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: jduran@uci.cu

Agradecimientos

…A la Revolución, a Fidel y a Raúl por la oportunidad y permitirme hacer un sueño realidad.

…A la Universidad de las Ciencias Informáticas (UCI) por formarme como un futuro profesional.

…A mis tutores Pedro y Javier por guiarme, ayudarme y apoyarme.

…A todos los que han contribuido de una forma u otra para el desarrollo de este trabajo. A todas las personas que me han apoyado y se han preocupado durante el desarrollo de mi carrera. Siéntanse parte de este logro.

René.

Dedicatoria

Me gustaría dedicar esta tesis a toda mi familia:

...a mis padres, que son el faro que me guía, mi razón de ser, mi vida entera, que me han dado todo lo que soy como persona, mis valores, mis principios, mi perseverancia y mi empeño, y todo ello con una gran dosis de amor incondicional.

...a mis hermanas Yanet y Mayelín con todo el cariño del mundo.

...a mi novia Aniuska, por su paciencia, por su comprensión, por su amor, por ser tal y como es,... porque la quiero y por estar siempre a mi lado en los buenos y en los malos momentos.

...a mi abuela Emilia y a mis abuelos Elena, Cicio y Celisdelio que aunque no estén físicamente siempre los llevo en mi corazón.

René

Resumen

El marco de trabajo Sauxe, desarrollado por el Centro de Informatización de la Gestión de Entidades (CEIGE), con el objetivo de soportar el desarrollo de aplicaciones web de gestión, es orientado a componentes y provee una estructura genérica para el desarrollo de aplicaciones web de gestión posibilitándole al desarrollador lograr una mayor estandarización, flexibilidad, integración y agilidad en el desarrollo del producto final. Para esto Sauxe se basa en la reutilización de componentes como Validaciones, Excepciones, Traza, IoC. Este último es quien establece la integración entre los sistemas que utilizan Sauxe.

La configuración de este componente se realiza sobre un XML donde se encuentran los servicios de integración de todos los sistemas en desarrollo sobre el marco de trabajo, este proceso se realiza totalmente manual por lo que resulta lento y engorroso además de que no se prevee el crecimiento del XML.

Con el trabajo realizado se desarrolló un componente que permite la configuración visual de los servicios de integración en el marco de trabajo Sauxe, brindándole a los desarrolladores una herramienta que facilite el proceso de configuración de los servicios.de integración, así tendrán una medida del crecimiento del XML, además estandarizarán la declaración de los servicios en el mismo y podrán realizar funciones como adicionar, modificar, eliminar y probar el servicio de forma visual, evitándose interactuar con el XML. Este componente se desarrolló sobre la base de políticas de software libre, utilizándose tecnologías web tales como (Apache, PostgresSQL, PHP, XML).

Palabras claves: Integración, servicios, componente, configuración.

Índice

Índice de figuras.

Índice de tablas

Índice de tablas

Introducción

Durante las dos últimas décadas, el desarrollo tecnológico muestra una convergencia entre la Informática, las Telecomunicaciones, la Electrónica y la Automatización, proceso que ha devenido una nueva rama del saber denominada Tecnologías de la Información y las Comunicaciones(TIC), de alta incidencia en la modernización y eficiencia de todos los sectores de la sociedad.

Cuba ha identificado la conveniencia y necesidad de dominar e introducir en la práctica social las TIC lo que facilitaría al país alcanzar un desarrollo sostenible. La Industria cubana del software inmersa en el perfeccionamiento continuo de la informática y las comunicaciones tiene como objetivo informatizar la sociedad, insertar a la isla en el mercado mundial del software y llegar a convertirse en puntera del desarrollo tecnológico, producto de dicho desarrollo surgió la Universidad de las Ciencias Informáticas (UCI), por idea del Compañero Fidel Castro Ruz y para impulsar el desarrollo de la informática en Cuba. Desde sus inicios la UCI desarrolla aplicaciones para la necesaria informatización de los principales sectores económicos y sociales del país.

El Centro de Informatización para la Gestión Entidades (CEIGE) de la facultad3 cuenta con un marco de trabajo para desarrollar aplicaciones web de gestión, denominado Sauxe; el cual es orientado a componentes y provee la estructura genérica para el desarrollo de aplicaciones web de gestión, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. Para lograr dicha integración Sauxe utiliza la Inversión de Control como patrón de diseño para permitir un menor acoplamiento entre todos los sistemas dentro de su desarrollo. Este patrón se implementa a través del componente IoC, que cuenta con un XML donde se encuentran los servicios de integración de todos los sistemas que utilizan el marco de trabajo.

Frecuentemente estos sistemas necesitan información de otros para su desarrollo, la que obtienen a través de los servicios de integración que se encuentran en el XML IoC. Cuando un sistema necesita un servicio de otro, para obtener información, el programador tiene que adicionar o modificar un servicio en el XML que dé respuesta a las exigencias del sistema que lo solicitó e informar cuando termina la tarea.

Este proceso trae consigo varios inconvenientes:

- Se realiza totalmente manual, por lo que resulta lento y engorroso.
- No se estandariza la declaración y descripción de los servicios en el XML.
- No se prevee el crecimiento del XML pues se pueden insertar servicios que ya existan en el mismo.

- No existe una vía de probar si el servicio cumple con el objetivo por el cual fue creado o modificado hasta el momento que sea utilizado.
- No existe un filtro para buscar un servicio en caso de que requiera modificación, lo que hace más trabajoso el proceso.

Teniendo en cuenta la situación planteada, el **problema a resolver** queda expresado de la siguiente forma: Necesidad de optimizar el proceso de configuración de la integración en el marco de trabajo Sauxe.

Definiéndose como **Objeto de estudio**: Marcos de trabajo para aplicaciones web de gestión.

Como **objetivo general** para resolver el problema planteado anteriormente se ha propuesto: Desarrollar un componente que permita la configuración visual de los servicios de integración en el marco de trabajo Sauxe.

Para dar cumplimiento al objetivo general se trazaron los siguientes **objetivos específicos**:

- Construir el marco teórico de la investigación.
- Realizar análisis y diseño de la solución.
- Realizar implementación y prueba de la solución.

Se identificó como **campo de acción**: Marco de trabajo Sauxe.
Se tiene como **idea a defender** en la presente investigación:

Con el desarrollo de un componente que permita la configuración visual de los servicios de integración en el marco de trabajo Sauxe, se optimiza el proceso configuración visual de los servicios de integración en dicho marco de trabajo.

Se identificaron como **Tareas de la investigación**:

- Estudiar los sistemas existentes relacionados con la configuración visual de los servicios de integración de aplicaciones obteniendo funcionalidades idóneas para el desarrollo del componente propuesto.
- Analizar los componentes del marco de trabajo Sauxe para una familiarización con la estructura y funcionamiento de los mismos.
- Desarrollar los artefactos que propone el Modelo de desarrollo orientado a componentes del proyecto ERP-Cuba enfocado al marco de trabajo Sauxe.
- Diseñar la interfaz de usuario del componente propuesto.

- Desarrollar las funcionalidades del componente teniendo en cuenta las buenas prácticas de programación.
- Validar las funcionalidades de la aplicación a través de pruebas unitarias y aplicación en un entorno real.

En la presente tesis el contenido está estructurado por tres capítulos que se muestran a continuación:

CAPÍTULO 1. Fundamentación Teórica: Se describen los aspectos y conceptos asociados al dominio del problema a resolver, siendo estos esenciales para entender el entorno del mismo. Se presenta el estado del arte de las técnicas implicadas en el objeto de estudio y el conjunto de tecnologías involucradas en el desarrollo de la propuesta.

CAPÍTULO 2. Propuesta de Solución: Se exponen los procesos de negocios y requisitos a cumplir por el componente además de los artefactos generados durante el diseño de la misma.

CAPÍTULO 3. Implementación y Prueba: Se exponen los artefactos generados durante la implementación de la solución así como las métricas y pruebas utilizadas para la validación de la misma

Capítulo 1: Fundamentación Teórica

En este capítulo se hace referencia a una serie de conceptos que son de vital importancia para la comprensión y desarrollo de la investigación. Se realiza un análisis del estado de los Frameworks que implementan y configuran el patrón IoC tanto en el ámbito nacional como internacional, además se exponen las principales características de las herramientas, tecnologías y la metodología utilizadas durante el desarrollo del sistema.

1.1 Aplicaciones web de gestión

Una aplicación web es un sistema informático de gestión accesible a través de Internet o de una red local. Las aplicaciones web son extremadamente populares debido a su practicidad y eficacia dentro del área de gestión y mantenimiento de una empresa**.** Las aplicaciones web permiten un estricto control de la información de una empresa, tanto en el plano de accesibilidad como en el de centralización de la misma. Otra de sus principales ventajas es la facilidad de actualización y mantenimiento**,** tanto de las propias aplicaciones como de la información gestionada con las mismas, dado que no es necesario instalar software a miles de potenciales usuarios. (1)

Ventajas:

- **Multiplataforma:** Las aplicaciones pueden ser utilizadas a través de variadas plataformas, tanto de hardware como de software, apoyado en gran medida por el uso de estándares.
- **Actualización instantánea:** Dado que los usuarios de la aplicación hacen uso de un solo programa que interactúa directamente con el servidor, siempre utilizarán la versión más actualizada del sistema.
- **Acceso no fijo:** El usuario puede acceder a la aplicación usando cualquier equipo provisto de red y navegador, estando ubicado en cualquier lugar y suponiendo siempre que cuenta con los permisos de acceso necesarios.
- **Fácil integración:** Debido a su basamento en protocolos estándares, la información manejada por el sistema puede ser accedida con mayor facilidad por otros sistemas.

1.1.1 Características principales de las aplicaciones web de gestión

En una aplicación Web la navegación y la entrada de datos por parte de un usuario, afectan el estado de la lógica del negocio; empleando para ello tecnologías que generan contenidos dinámicos. Si no existe lógica del negocio en el servidor, el sistema no es considerado aplicación Web.

Las interfaces de las páginas Web poseen ciertas limitantes en la funcionalidad del cliente. Métodos comunes en las aplicaciones de escritorio como dibujar en la pantalla, arrastrar o soltar no están soportados por las tecnologías Web estándar. Los desarrolladores utilizan generalmente lenguajes interpretados que se ejecutan en el cliente Web para obtener una mayor funcionalidad, así como tecnologías que se ejecutan en el servidor para no tener que recargar la página en su totalidad, algo que molesta mucho a los usuarios. Se han desarrollado técnicas para coordinar estos lenguajes con tecnologías del lado del servidor, como por ejemplo Personal Home Page Tools (PHP) y Asynchronous JavaScript And XML (AJAX). (2)

1.2 Inversión de Control

La Inversión de Control es una característica común de muchos marcos de trabajo; es un patrón de diseño que permite un menor acoplamiento entre componentes de una aplicación y fomenta así la reutilización de los mismos. El patrón IOC aplica un principio de diseño denominado principio de Hollywood (*No nos llames, nosotros te llamaremos*). (3)

1.2.1 Usos

El patrón IoC se puede utilizar cuando:

- Se desee desacoplar las clases de sus dependencias de manera de que las mismas puedan ser reemplazadas o actualizadas con muy pocos o casi ningún cambio en el código fuete de sus clases.
- Desea escribir clases que dependan de clases cuyas implementaciones no son conocidas en tiempo de compilación.
- Desea testar las clases aisladamente sin sus dependencias.
- Desea desacoplar sus clases de ser responsables de localizar y gestionar el tiempo de vida de sus dependencias.

1.2.2 Técnicas de implementación

Según diversos enfoques, este patrón puede implementarse de diversas maneras. Entre las más conocidas se tiene:

- ✓ **Service Locator**: Es un componente que contiene referencias a los servicios y encapsula la lógica que los localiza. Así, en las clases, se utiliza el Service Locator para obtener instancias de los servicios que realmente se necesitan. El Service Locator no instancia los servicios. Provee una manera de registrar servicios y mantener una referencia a dichos servicios. Luego de que el servicio es registrado, el Service Locator puede localizarlo.

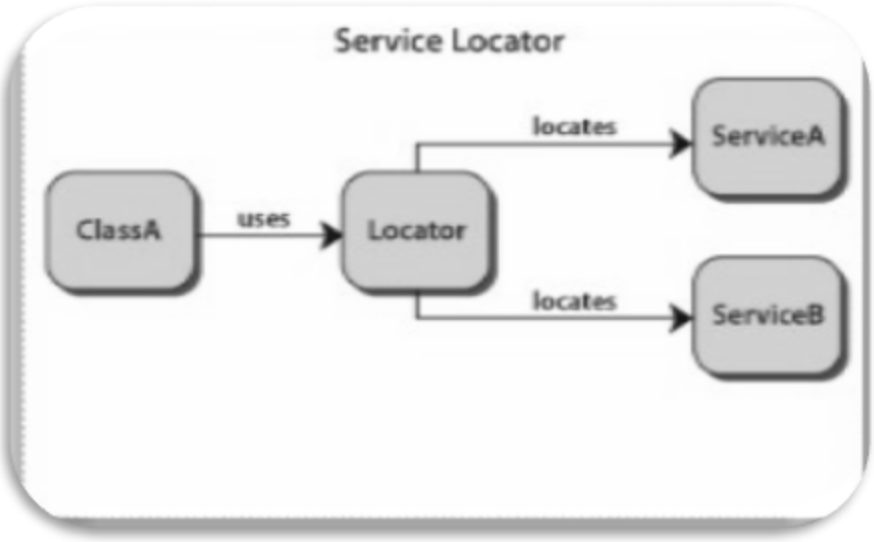


Figura 1 Service Locator

- ✓ **Inyección de dependencias (DI)**: del inglés dependecy injection; propone no instanciar las dependencias explícitamente en su clase, sino que declarativamente expresarlas en la definición de la clase. La esencia de la inyección de las dependencias es contar con un componente capaz de obtener instancias válidas de las dependencias del objeto y pasárselas durante la creación o inicialización del objeto. Se puede implementar de 3 formas, inyección basada en constructor, inyección basada en métodos setters e inyección basada en interfaces.

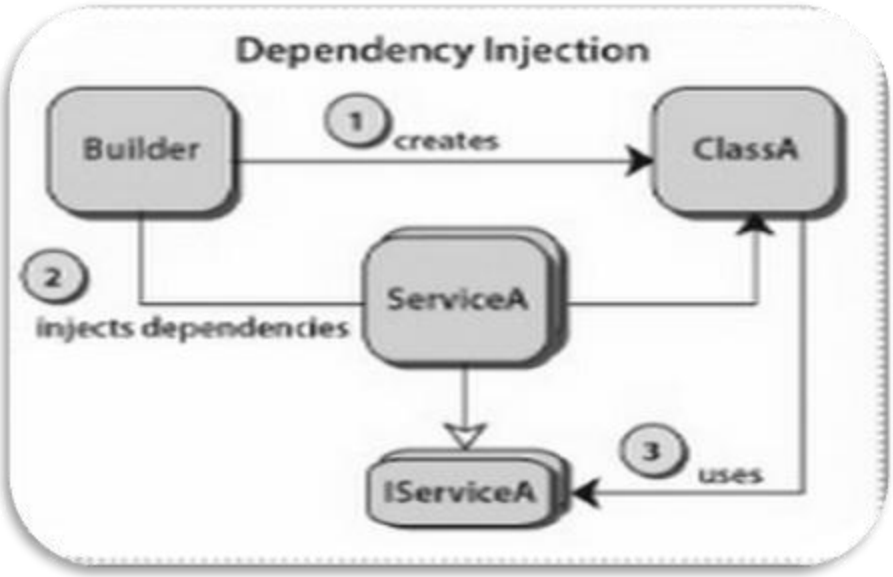


Figura 2 Dependency Injection

1.3 Marcos de Trabajo (Frameworks)

En el desarrollo de software, un marco de trabajo o framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Normalmente, un framework incluye soporte de programas, librerías además de un lenguaje de scripting que sirve de apoyo en el desarrollo y la integración de los diferentes componentes de una aplicación. Un framework representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. En general estos se diseñan y optimizan para resolver la mayor parte de la complejidad de los desarrollos, aprovechando las mejores prácticas y facilitando la construcción de nuevas aplicaciones. (4)

1.3.1 Características principales de los marcos de trabajo

A continuación se enuncian una serie de características que podemos encontrar en prácticamente todos los frameworks existentes.

- **Abstracción de URLs y sesiones:** No es necesario manipular directamente las URLs ni las sesiones, el framework ya se encarga de hacerlo.
- **Acceso a datos:** Incluyen las herramientas e interfaces necesarias para integrarse con herramientas de acceso a datos, en bases de datos, XML, etc.

- **Controladores:** La mayoría de frameworks implementa una serie de controladores para gestionar eventos, como una introducción de datos mediante un formulario o el acceso a una página. Estos controladores suelen ser fácilmente adaptables a las necesidades de un proyecto concreto.
- **Autentificación y control de acceso:** Incluyen mecanismos para la identificación de usuarios mediante login y password y permiten restringir el acceso a determinadas páginas de determinados usuarios.
- **Separación entre diseño y contenido**: La mayoría de los frameworks implementan el patrón MVC (modelo – vista -controlador). Este patrón organiza la aplicación en tres modelos separados, el primero es un modelo que representa los datos de la aplicación y sus reglas de negocio, el segundo es un conjunto de vistas que representa los formularios de entrada y salida de información, el tercero es un conjunto de controladores que procesa las peticiones de los usuarios y controla el flujo de ejecución del sistema.

1.3.2 Marcos de trabajo que implementan el patrón IoC

- ✓ **Spring**

Framework de código abierto bastante popular, que proporciona la posibilidad de integrarse con otras herramientas o frameworks, para esto brinda diferentes módulos, según la herramienta o el frameworks a integrar. Ofrece a los desarrolladores soluciones bien documentadas. Promueve la reutilización del código pues está diseñado con interfaces que pueden ser utilizadas por los desarrolladores. Está basado en el patrón Inversión de Control, el cual se implementa a través de la inyección de dependencias, la configuración de este se realiza en un archivo XML, donde se encuentran los beans que no son más que objetos creados y manejados por el contenedor Spring. Dicha configuración se puede realizar manualmente aunque este marco de trabajo cuenta con los BeanDefinition y BeanFactory que tienen varias implementaciones específicas para poder realizar la modificación dinámica de los beans. (5)

Ventajas:

- Facilita la manipulación de los objetos.
- Elimina la necesidad de usar distintos y variados tipos de ficheros de configuración.
- Permite el uso de la programación orientada a aspectos.

- ✓ **Symfony**

Symfony es un framework PHP que facilita el desarrollo de las aplicaciones web. Se encarga de todos los aspectos comunes en las aplicaciones web, dejando que el programador se dedique a aportar valor desarrollando las características únicas de cada proyecto. El mismo aumenta exponencialmente la productividad y ayuda a mejorar la calidad de las aplicaciones web aplicando todas las buenas prácticas y patrones de diseño que se han definido para la web. Uno de estos patrones es IoC, Symfony administra la inyección de dependencia de contenedores de servicios a través de una clase denominada sfServiceContainer cuando se tiene un número muy reducido de servicios para la gestión. Desde el momento que dichos servicios empiezan a aumentar progresivamente este marco de trabajo utiliza la clase sfServiceContainerBuilder para describirlos y así facilitar el proceso de definición de los mismos. La descripción de estos se realiza sobre archivos XML y YAML, Symfony tiene clases de ayuda para los servicios de carga de estos archivos cómo son sfServiceContainerLoaderFileXml para cargar los archivos XML, y sfServiceContainerLoaderFileYaml para cargar archivos YAML además de que utiliza los objetos dumper para poder convertir los contenedores de servicios a XML o YAML sin tener que realizarlo manualmente. (6)

Ventajas:

- ❖ Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares).
- ❖ Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- ❖ Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional.
- ❖ Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- ❖ Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- ❖ Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- ❖ Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros. (7)

- ✓ **Microsoft Unity**

Unity es un contenedor ligero de inyección de dependencia que facilita la creación de aplicaciones de acoplamiento flexible y proporciona a los desarrolladores:

Ventajas:

- ❖ Provee una manera simplificada de creación de objetos, especialmente para crear objetos de una jerarquía de clases, simplificando el código de la aplicación.
- ❖ Soporte abstracción de requerimientos, lo que permite a los desarrolladores especificar dependencias en tiempo de ejecución o mediante un fichero de configuración, simplificando el manejo del código transversal en las aplicaciones
- ❖ Dispone de capacidades para la localización de servicios. Esto permite a los clientes almacenar o cachear el contenedor
- ❖ Incrementa la flexibilidad, dejando la configuración de los componentes al contenedor. (8)

Unity es un framework de propósito general para su uso en cualquier tipo de framework .net, proporciona todas las funciones que se encuentran comúnmente en los mecanismos de inyección de dependencias, incluidos los métodos para registrar las asignaciones de tipos y las instancias de objetos además de inyectar objetos dependientes en los parámetros de los constructores, los métodos y el valor de las propiedades de los objetos que resuelve. Unity lee la información de configuración de un archivo XML, de forma predeterminada siempre establece que se realice de los archivos de la aplicación app.config o web.config, sin embargo, puede cargar la información de configuración desde cualquier otro archivo XML o de otras fuentes si es necesario. Esta configuración puede realizarse tanto en tiempo de ejecución como en tiempo de diseño. (9)

- ✓ **Marco de Trabajo Sauxe**

El Marco de Trabajo Sauxe se ha estructurado de manera tal que facilite la reutilización de los diferentes componentes y que sea de fácil entendimiento para todos los programadores que desarrollen sobre el mismo. Sauxe utiliza EXT para implementar la capa de presentación, se apoya en Zend, una extensión de Zend Framework para el desarrollo de la lógica del negocio y para la gestión de los datos, utiliza Doctrine.

Sauxe cuenta con un componente de transacciones mediante el cual serán salvados automáticamente los datos de modificaciones e inserciones, es decir, ya no será necesaria la implementación por parte del programador de las consultas de inserción, éste solamente deberá programar la obtención de los campos de la presentación y el Sauxe será el responsable de que los mismos sean guardados en la base de datos.

Ventajas:

- Los identificadores de cada tupla de las tablas serán generados automáticamente en la base de datos como una secuencia.
- El antiguo método de try y catch para el lanzamiento de excepciones desaparece, en el nuevo marco con registrar las excepciones en el manager de excepciones, el marco de trabajo será capaz de realizar un tratamiento óptimo de las mismas.
- No será necesario en cada método de inserción de información a la base de datos ejecutar el método correspondiente en la clase del negocio correspondiente, el marco de trabajo será capaz de tomar los datos de la presentación salvarlos en la base de datos directamente.

Es importante mencionar que Sauxe utiliza como patrón arquitectónico MVC, este marco de trabajo en su nueva configuración define que la conexión a la base de datos será configurada en un *XML* almacenado en la carpeta de recursos comunes del proyecto, además tiene como propósito insertar la programación orientada a aspectos así como la inversión de control, la cual se realizará con la implementación del patrón de diseño IoC. La configuración de este patrón se realiza sobre un archivo XML, donde se encuentran todos los servicios de integración de todos los sistemas que estén en desarrollo sobre el marco de trabajo, actualmente dicha configuración se realiza totalmente manual lo que resulta engorroso el trabajo.

Tras un análisis de todos estos sistemas se llegó a la conclusión de que la forma en que estos implementan y configuran el patrón IoC no es aplicable al Marco de trabajo Sauxe, aunque tienen características en común, se decidió realizar una solución a la medida que cumpliera con las exigencias del cliente como buscar y probar los servicios de forma eficiente, así como poder adicionar, modificar o eliminar un servicio en el momento en que se requiera.

1.4 Herramientas y Tecnologías

El presente trabajo forma parte del proceso productivo del CEIGE; el cual ha tomado decisiones tecnológicas que involucran la utilización de tecnologías de código abierto para el desarrollo de sus productos. A continuación se describen brevemente estas tecnologías.

1.4.1 Modelo de desarrollo orientado a componentes

El centro CEIGE dentro de sus propuestas tecnológicas para el desarrollo de sus aplicaciones, seleccionó como modelo de desarrollo el Modelo de desarrollo orientado a componentes del proyecto ERP-Cuba, el mismo está formado por 8 procesos y una estructura organizativa; cada proceso es especificado teniendo en cuenta su flujo de actividades, los roles involucrados y artefactos generados en cada actividad. Este modelo permite organizar el proceso de desarrollo de software tecnológico del CEIGE. Dicho modelo está basado en principios y buenas prácticas de metodologías ágiles, fue elaborado teniendo en cuenta las características especiales que presenta la UCI y tiene un valor social representativo, ya que implica mejoría en aspectos importantes como la planeación, formación y satisfacción del equipo de trabajo. (10)

Principales características de este modelo:

- Está orientado a la reutilización de componentes parametrizables.
- Se utilizan solamente los artefactos necesarios para documentar el producto.
- Se desarrollan partes pequeñas y se ensambla después el producto.
- Los flujos se integran a través de la arquitectura de software y de negocio. Todos los flujos de desarrollo de cada fase se integran siempre detrás de la arquitectura de software y de negocio.
- Existen áreas dedicadas a tareas específicas y especializadas en temas específicos.
- Se hacen pruebas continuas sobre los compones y/o productos y los cambios se hacen a tiempo, antes de poner un componente en el repositorio se hacen pruebas unitarias, y cuando se va a utilizar como parte de otro producto se hacen pruebas de integración, al igual que antes de liberar el producto también. Todo esto demuestra que se está probando en todo el proceso de desarrollo.
- Las áreas de proceso están especializadas, en temas de apoyo a la producción que es el elemento fundamental de la Subdirección y llevan unido al proceso productivo procesos tales como Investigación, Formación, Gestión del capital humano y calidad.
- Es un método muy estructurado que funciona bien con gente de poca experiencia.
- Reduce los riesgos ya que:

- Provee visibilidad sobre el progreso a través de sus nuevas versiones.
- Provee retroalimentación a través de la funcionalidad mostrada.
- Permite atacar los mayores riesgos desde el inicio.

- La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software.
- En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.
- Preocupación por el aprendizaje de los desarrolladores.

1.4.2 Visual Paradigm 6.4

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una construcción más rápida de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML.

Ventajas:

- Tiene disponible distintas versiones: Enterprise, Professional, Standard, Modeler, Personal y Community (que es gratuita).
- Ayuda a construir aplicaciones de calidad más rápidamente, mejor y a más bajo costo.
- Se pueden dibujar todos los tipos de diagramas de clase, código inverso, generar el código de diagramas y generar la documentación.
- Alta interoperabilidad: Los usuarios y proveedores de tecnología pueden integrar con Visual Paradigm modelos en sus soluciones con un mínimo esfuerzo.

1.4.3 Lenguaje Unificado de Modelado (UML) 2.1

Lenguaje estándar para el modelado de software, utilizado para visualizar, especificar, construir y documentar los artefactos de un sistema con gran cantidad de software. Además de que permite a los desarrolladores visualizar el producto de su trabajo (Artefactos) en esquemas o diagramas estandarizados.

Ventajas:

- Permite modelar sistemas utilizando técnicas orientadas a objetos.
- Permite especificar todas las decisiones de análisis y diseño, construyéndose así modelos precisos, no ambiguos y completos.
- Puede conectarse con lenguajes de programación (Ingeniería directa e inversa).
- Permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones, etc.).
- Es un lenguaje muy expresivo que cubre todas las vistas necesarias para desarrollar y luego desplegar los sistemas.
- Existe un equilibrio entre expresividad y simplicidad, pues no es difícil de aprender ni de utilizar.

1.4.4 Apache 2.2.9

Apache es el servidor Web hecho por excelencia y más difundido de Internet. Su robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa.

Ventajas:

- Es compatible con una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.
- Es una tecnología gratuita de código fuente abierta.
- Es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor.
- Trabaja con gran cantidad de lenguajes como Perl, PHP y otros lenguajes script.
- Permite la configuración de ficheros de logs, dándole al administrador un mayor control sobre lo que sucede en el servidor.
- Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor.

1.4.5 PostgreSQL 8.3

Es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS), basado en software libre y liberado bajo la licencia BSD, lo que significa que cualquiera puede disponer de su código fuente, modificarlo a voluntad y redistribuirlo libremente. Está considerado como el sistema de gestión de bases

de datos de código abierto más potente del mercado, ofrece nuevos conceptos como son: clases, herencia, tipos y funciones. Aporta en cierta forma potencia y flexibilidad adicional como son las restricciones, los disparadores, las reglas y la integridad transaccional, posee una amplia variedad de tipos nativos, o sea, presenta soporte para números de precisión arbitraria, texto de largo ilimitado, figuras geométricas con una variedad de funciones asociadas, direcciones IP, arreglos entre otros. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. (11)

Ventajas:

- Es estable.
- Es flexible.
- Se puede extender su funcionalidad.
- Tiene gran compatibilidad con sistemas operativos.

1.4.6 NetBeans 6.8

NetBeans es una herramienta para programadores pensada para 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 extenderlo. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso. (12)

Ventajas:

- Auto-completado y documentación de funciones PHP: Rápido acceso a la documentación de PHP, y si se necesita más información se provee el link directo a la función.
- Auto-completado de código propio: Esto es una consecuencia del punto anterior, al documentar código con el formato esperado, estos serán mostrados.
- Atajos de teclado muy útiles: Permite a través de varios comandos la ejecución de numerosas funcionalidades y existen muchas más como integración con Xdebug, soporte para Symfony, Zend Framework, Smarty, historia local para archivos. Entre otros.

1.4.7 Lenguajes de Programación.

Programación Orientada a Objetos

La tecnología orientada a objetos se define como una metodología de diseño de software que modela las características de objetos reales o abstractos por medio del uso de clases y objetos, la programación orientada a objetos (POO) es un paradigma que usa dichos objetos y sus interacciones para diseñar aplicaciones y programas informáticos que se fundamenta sobre las siguientes bases: (13)

- Abstracción: La abstracción es el proceso en el cual se separan las propiedades más importantes de un objeto, de las que no lo son.
- Modularidad: La modularidad, permite poder modificar las características de la clase que definen a un objeto, de forma independiente de las demás clases en la aplicación.
- Encapsulamiento: El encapsulamiento es la propiedad de la orientación a objetos que permite asegurar que la información de un objeto le es desconocida a los demás objetos en la aplicación.
- Jerarquía: La jerarquía "es un" se le conoce como herencia. La herencia simple es la propiedad que permite definir una clase nueva en términos de una clase ya existente mientras que la jerarquía de agregación, también conocida como inclusión, se puede decir que se trata del agrupamiento lógico de objetos relacionados entre sí dentro de una clase.
- Polimorfismo: El polimorfismo es la propiedad por la cual una entidad puede tomar diferentes formas.

Ventajas:

- Flexibilidad.
- Reusabilidad.
- Mantenibilidad.
- Extensibilidad.

✓ PHP (Lenguaje del lado del servidor)

Preprocesador de Hipertexto o PHP (por sus siglas en inglés) es un lenguaje de programación de alto nivel orientado al desarrollo de aplicaciones web, que es interpretado y de plataforma independiente. Sus

sintaxis son muy similares a lenguajes como C y PERL. Puede ser utilizado en casi todos los sistemas operativos existentes, permitiendo migrar las aplicaciones de un sistema a otro sin necesidad de realizar cambios en el código. Su rapidez en la ejecución y los bajos requerimientos de consumo en los sistemas donde es desplegado lo hacen uno de los preferidos por los desarrolladores. Es multiplataforma y presenta una alta capacidad de conexión con gran parte de los Sistemas Gestores de Bases de Datos.: MySQL, PostgreSQL, Oracle, MS SQL Server, además también tiene la posibilidad de extender sus potencialidades utilizando módulos. Su mayor ventaja radica en ser un lenguaje libre, por lo que se convierte en una alternativa de muy fácil acceso, además de poseer una comunidad de desarrolladores que intercambian experiencias, de esta forma cuando se presenta un problema, es muy fácil obtener documentación para darle solución de forma rápida y sin costo alguno. Quizás una de sus mayores desventajas radica en que promueve la creación de código desordenado, por lo que lo hace muy complejo de mantener. (14)

- ✓ **JavaScript (Lenguaje del lado del cliente)**

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. Este lenguaje permite interactuar con el navegador de manera dinámica y eficaz, proporcionando a las páginas web dinamismo y vida. (15)

1.4.10 Librerías y Marcos de Trabajo

- ✓ **Marco de trabajo Sauxe**

El desarrollo de la solución se realizará utilizando el marco de trabajo Sauxe, desarrollado por el Departamento de Tecnología del CEIGE, el cual contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. (16)

Sauxe está compuesto por varios frameworks, estructurados en niveles o capas como se puede apreciar en la siguiente figura 3, los cuales serán descritos a continuación.

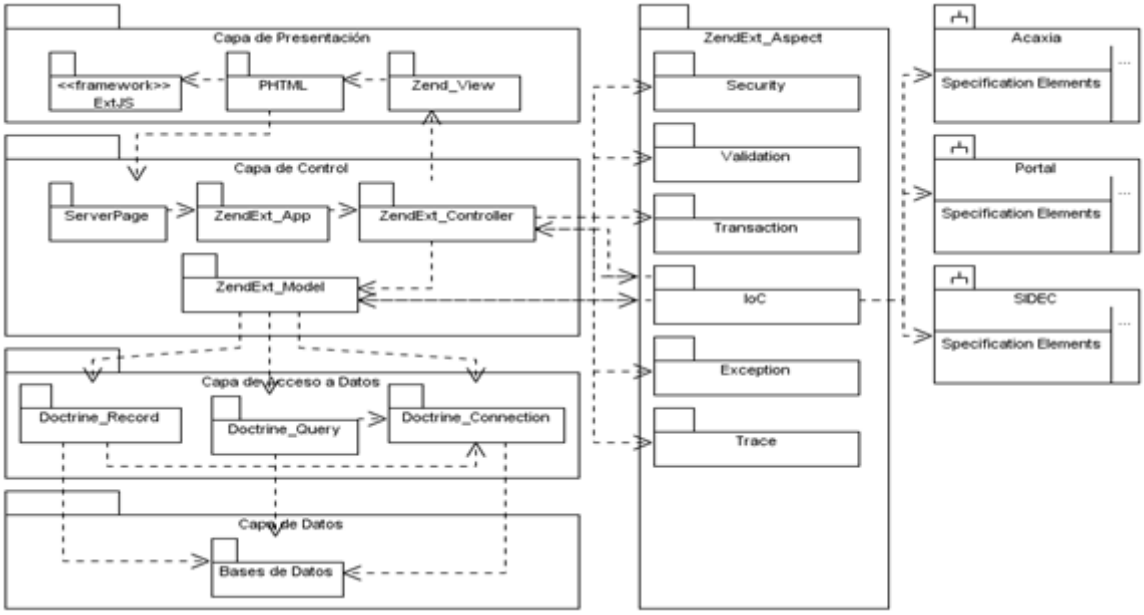


Figura 3 Estructura del marco de trabajo Sauxe.

✓ Zend Frameworks 1.8

Es un framework de código abierto para desarrollar aplicaciones web y servicios web con PHP5. Zend Framework es una implementación que usa código 100% orientado a objetos. La estructura de los componentes de Zend Framework 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, aunque se pueden utilizar de forma individual, los componentes de la biblioteca estándar de Zend Framework conforman un potente y extensible framework de aplicaciones web al combinarse. Zend Framework ofrece un gran rendimiento y una robusta implementación MVC, una abstracción de base de datos fácil de usar, y un componente de formularios que implementa la prestación de formularios HTML, validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos. (17)

Ventajas:

- Reduce el "time tomarket" de las aplicaciones, permitiendo ofrecer presupuestos más ajustados.
- Estandariza los procesos más frecuentes, dotándolos de gran robustez.
- Facilita el mantenimiento de las aplicaciones.

- ❖ Ofrece muchas facilidades para el acceso a recursos avanzados que de otro modo resultan bastante más costosos de desarrollar.
- ❖ A diferencia de otros frameworks, es posible utilizarlo en modo "desacoplado", es decir, aquellas clases o componentes que sean necesarios en cada proyecto, sin arrastrar todo el framework detrás para cualquier pequeña necesidad.
- ❖ Tiene el respaldo de la propia ZEND, creadora de PHP, lo que asegura su continuidad futura tanto como la del propio lenguaje PHP.

- ✓ **ExtJS 2.2**

Es una librería Java Script open-source de alto rendimiento para la creación y desarrollo de aplicaciones web dinámicas. Provee interfaces gráficas de usuario que brindan experiencias parecidas o iguales a las que se tienen con aplicaciones de escritorio. Permite la creación de aplicaciones complejas utilizando componentes predefinidos. Es extensible para la gran mayoría de los navegadores, evitando el tedioso problema de validar el código para cada uno de estos. Entre sus principales ventajas se encuentra el balance entre Cliente-Servidor, distribuyendo la carga de procesamiento en el último, y este al tener menor carga, maneja los clientes de manera más eficiente. La comunicación asíncrona permite el intercambio de información con el servidor sin necesidad de pedirle una acción al usuario, dando la libertad de cargar la información sin que este lo note. (18)

Ventajas:

- ❖ Nos permite crear aplicaciones complejas utilizando componentes predefinidos.
- ❖ Evita el problema de tener que validar el código para que funcione bien en cada uno de los navegadores (Firefox, IE, Safari, Opera etc.).
- ❖ El funcionamiento de las ventanas flotantes lo pone por encima de cualquier otro.
- ❖ Relación entre Cliente-Servidor balanceado: Se distribuye la carga de procesamiento entre, permitiendo que el servidor pueda atender más clientes al mismo tiempo.
- ❖ Eficiencia de la red: Disminuye el tráfico en la red pues las aplicaciones cuentan con la posibilidad de elegir que datos desea trasmitir al servidor y viceversa.

- ✓ **Doctrine 0.11**

Doctrine es un potente y completo sistema de mapeado de objetos relacionales (ORM) paraPHP 5.2 o superior, que posee una formidable capa de abstracción a base de datos. Entre sus principales características se encuentra la posibilidad de escribir de manera opcional las consultas a la base de datos. Lo que proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad, sin necesidad de la duplicación de código innecesaria. Permite también exportar una base de datos existente a sus clases correspondientes y también convierte clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos. (19)

1.5 Especificaciones de la arquitectura

Teniendo en cuenta que el presente trabajo de diploma está centrado en el desarrollo de un componente para el marco de trabajo Sauxe, no es propósito del autor crear una nueva arquitectura para dicho desarrollo y por tanto adopta la arquitectura definida por CEIGE.

1.5.1 Arquitectura Cliente-Servidor

La arquitectura Cliente/Servidor es una nueva tendencia en el desarrollo de redes, que tiene como objetivo optimizar el uso tanto del hardware como del software, a través de la separación de funciones: el cliente, quien inicia una determinada petición y el servidor, dedicado a responder dichas peticiones.

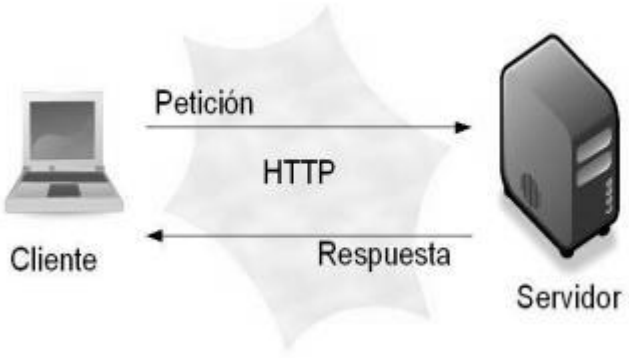


Figura 4 Arquitectura Cliente-Servidor.

Características de la arquitectura Cliente/Servidor:

- ✓ El servidor presenta una interfaz única y bien definida a todos sus clientes.
- ✓ El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.
- ✓ El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- ✓ Los cambios en el servidor no afectan al cliente.

1.6 Conclusiones parciales

En este capítulo realizó un análisis sobre las herramientas y técnicas a utilizar para darle solución al problema planteado. Se concibió un estudio sobre los frameworks que implementan y configuran el patrón IoC describiendo sus principales características y logrando establecer alguna similitud con la investigación en curso.Se decide desarrollar un componente web que facilite la gestión y configuración de los servicios que se encuentran en el XML IoC. Se utilizará el Modelo de desarrollo orientado a componentes, así como herramientas y tecnologías acordes al resultado que se quiere alcanzar.

Capítulo 2: Propuesta de Solución

El siguiente capítulo aborda sobre los resultados obtenidos durante el proceso de desarrollo de la solución, así como algunos de los artefactos generados por el mismo. En el mismo son descritos los procesos de y los requisitos funcionales de la solución, mostrando una propuesta de lo que se desea lograr. Posteriormente se realiza una descripción del diseño elaborado por el autor para alcanzar las metas trazadas, además de la estrategia a seguir durante el proceso de implementación.

2.1 Descripción de los procesos de negocios

Un proceso de negocio es un conjunto de tareas lógicamente relacionadas que existen para conseguir un resultado bien definido dentro de un negocio (20). La descripción de los procesos de negocio facilita las actividades del análisis debido a que posibilita una compresión más clara de los procesos en cuestión y contribuye a que los requisitos que se definan satisfagan las necesidades del usuario final.

El autor, teniendo en cuenta la opinión de especialistas y de los usuarios finales, ha identificado dos procesos de negocio que deben ser asistidos por la solución. Estos procesos son descritos a continuación.

2.1.1 Descripción del proceso: Adicionar servicios

La solución propuesta debe permitir a los usuarios adicionar un servicio al XML IoC a través de un componente web, en la figura 4 se muestra el diagrama que identifica dicho proceso partiendo del punto inicial en que el usuario tiene que buscar si el servicio que desea consumir existe ya en el XML, en caso de no existir el usuario tiene que notificar al desarrollador del subsistema del cual necesita el servicio la inexistencia del mismo, este crea e introduce el nuevo servicio en el XML, es importante destacar que debe existir al menos un método declarado en las clases Services para que el desarrollador pueda llevar a cabo estas acciones satisfactoriamente, posteriormente este informa al usuario que solicitó el servicio, que el mismo ya se encuentra en el XML IoC terminando así este proceso.

Como resultado de este proceso se debe obtener el XML IoC actualizado con el nuevo servicio recién adicionado por el desarrollador.

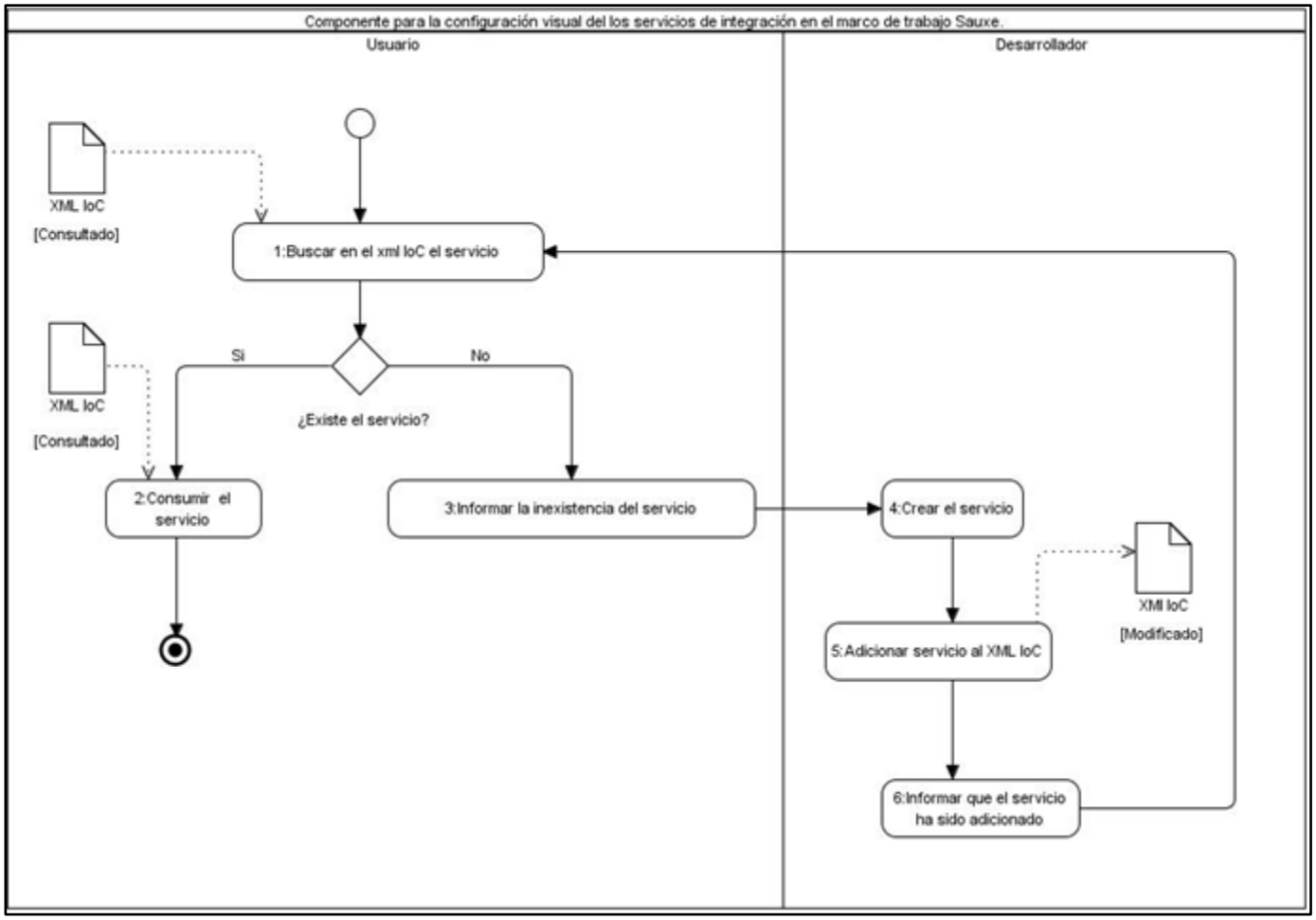


Figura 5 Proceso Adicionar Servicios

2.1.2 Descripción del proceso: Modificar Servicios

La solución propuesta debe permitir a los usuarios modificar un servicio en el XML IoC a través de un componente web, en la figura 5 se muestra el diagrama que identifica dicho proceso partiendo del punto inicial en que el usuario tiene que buscar el servicio en el XML y comprobar si es preciso para lo que se requiere, de no ser así este tiene que informar al desarrollador del subsistema del cual necesita el servicio que el mismo no es apropiado para la función que se tiene que realizar, posteriormente este modifica el servicio e informa al usuario que el servicio que necesita ya ha sido modificado terminando así este proceso.

Como resultado de este proceso se obtiene el XML IoC actualizado debido a la modificación realizada por el desarrollador sobre el servicio.

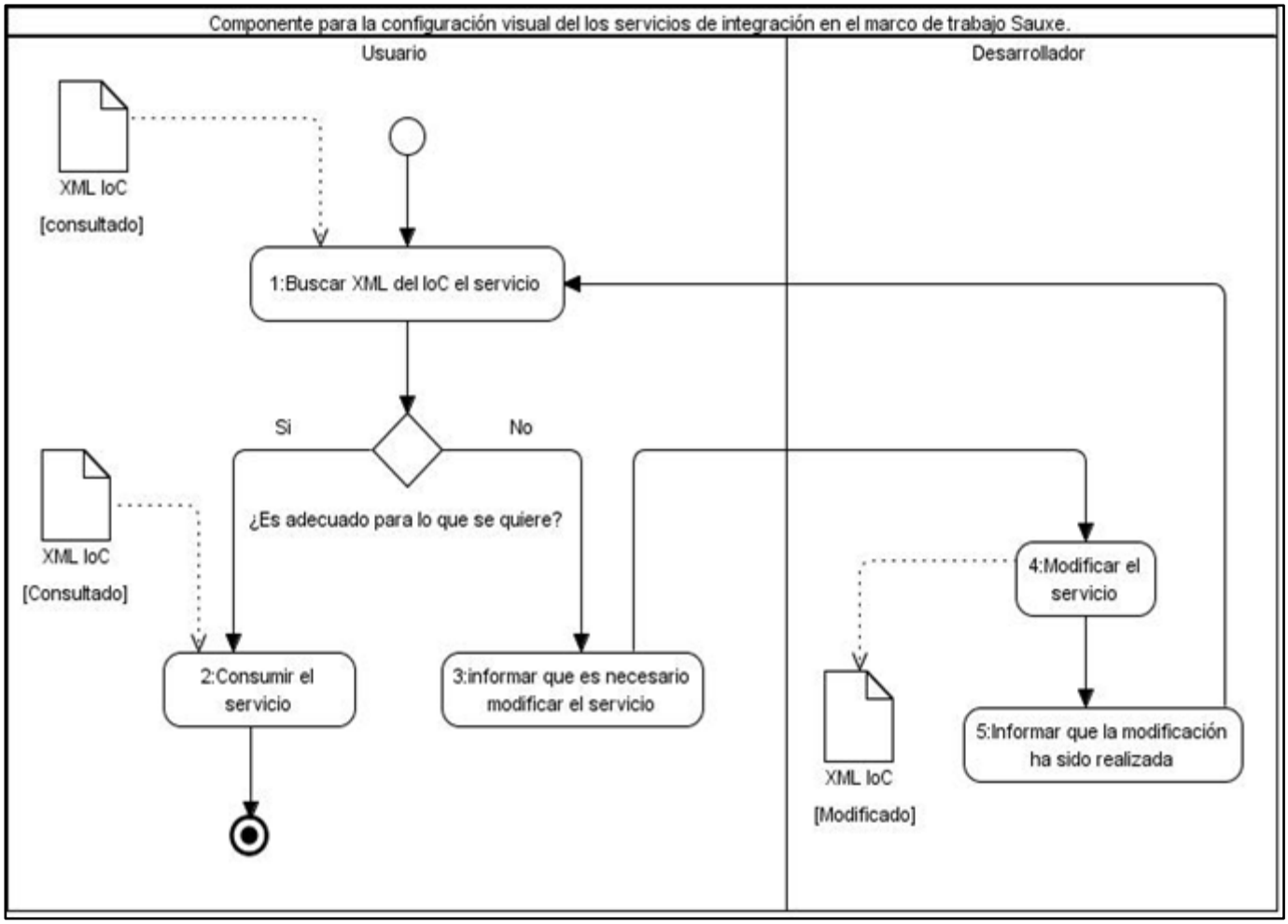


Figura 6 Proceso Modificar Servicios

2.2 Requisitos de la solución propuesta

Una vez que se realiza el modelado del negocio se puede comenzar a ejecutar el proceso de captura de requisitos del sistema, esta es una de las actividades fundamentales que se desarrolla en el proceso de desarrollo de software. Los requisitos son la condición o capacidad que tiene que ser alcanzada o poseída

por un sistema o componente de software para satisfacer un contrato, estándar, u otro documento impuesto formalmente. Los mismos pueden dividirse en requisitos funcionales y requisitos no funcionales. Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. A continuación se listan los requisitos que debe cumplir el componente a desarrollar.

2.2.1 Requisitos Funcionales

RF1 Gestionar servicios de integración.

RF 1.1 Adicionar servicios.

RF1.2 Modificar servicios.

RF1.3 Eliminar servicios.

RF1.5 Buscar servicios.

RF1.6 Listar servicios

RF2 Probar servicios.

2.2.1.1 Requisito funcional: Gestionar servicios de integración

Tabla 1 Especificación del requisito Adicionar servicios

Conceptos tratados	**Conceptos**	**Atributos**
	Servicios.	Nombre, descripción, retorno y dirección wdsl del servicio, así como el método y la clase y los parámetros que recibirá el mismo.
Precondiciones	**Precondiciones**	**Pre-requisito**

	El usuario se ha autenticado en el sistema y tiene permisos para ejecutar la acción adicionar servicio.	Listar servicios
	Debe existir al menos declarado un método para poder adicionar el servicio.	
Descripción	Para adicionar un servicio el usuario debe seleccionar el subsistema al cual quiere añadirle el mismo e introducir sus datos; debe elegir la clase y dentro de esta el método que implementará el servicio, luego de realizar esta acción el sistema automáticamente obtendrá los parámetros del mismo, dándole al usuario la oportunidad de modificar el tipo y la descripción de estos, además de los datos mencionados el usuario tiene que introducir de forma obligatoria el nombre, el retorno y la descripción del servicio siendo la dirección wsdl el único dato opcional que de no ser especificado tomará valor nulo. El sistema deberá detectar posibles errores en los datos especificados por el usuario y notificar al mismo para su corrección. En caso de que no existan errores en los datos, el sistema debe comprobar que los métodos están bien comentados y de ser así, adicionará el servicio y mostrará un mensaje de notificación. El sistema debe permitir la cancelación de esta acción.	

Validaciones	No se permite la creación de un servicio con el mismo nombre de otro ya existente. No se permiten valores nulos en los datos de carácter obligatorio.
Post-condiciones	Se ha adicionado un nuevo servicio al XML.
Post-requisito	El sistema lista los servicios.

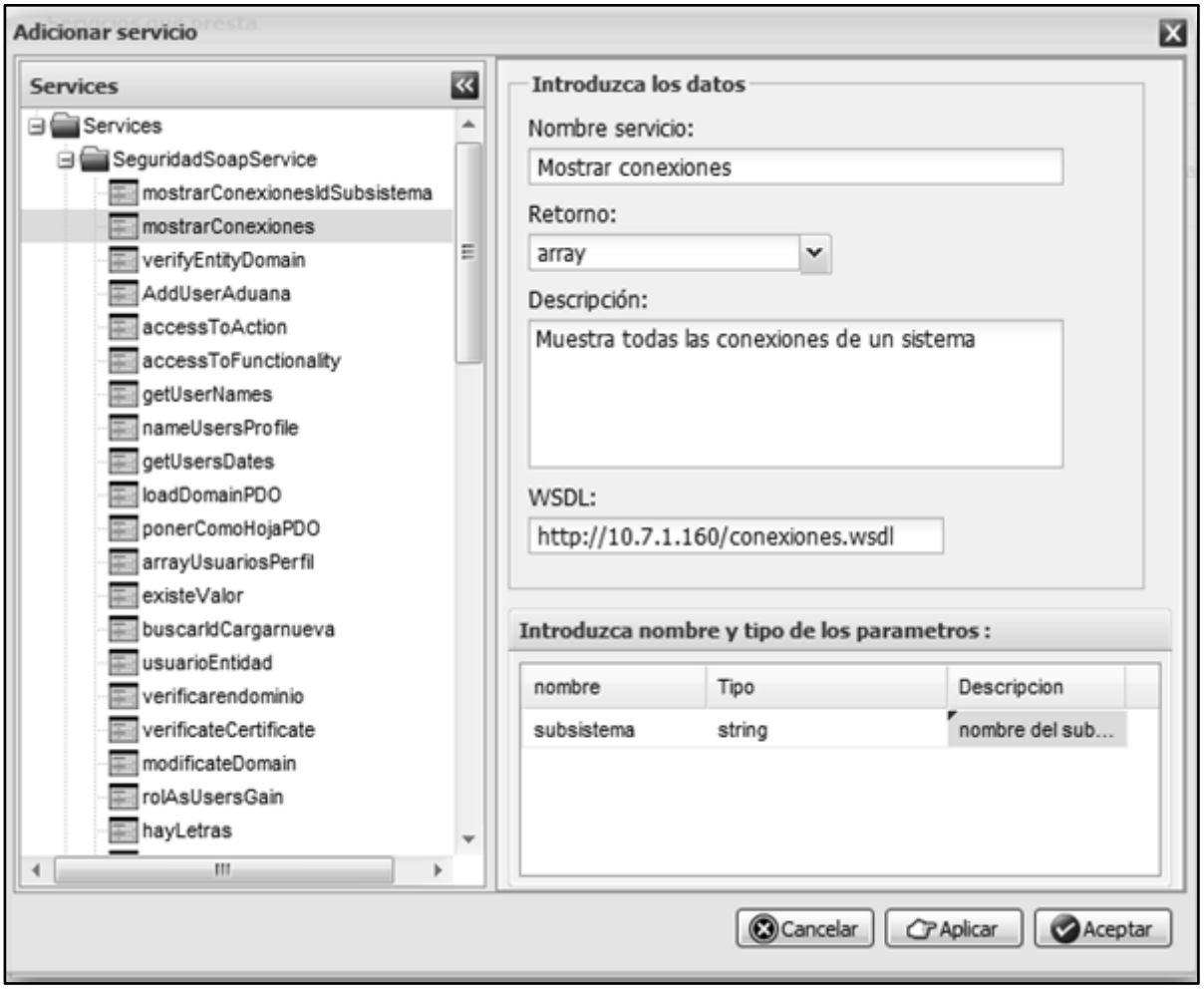


Figura 7 Interfaz de usuario del requisito Adicionar servicios.

Tabla 2 Especificación del requisito Modificar servicios.

Conceptos tratados	**Conceptos**	**Atributos**
	Servicios.	Nombre, descripción, retorno y dirección wdsl del servicio, así como el método y la clase y los parámetros que recibirá el mismo.
Precondiciones	**Precondiciones**	**Pre-requisito**
	El usuario se ha autenticado en el sistema y tiene permisos para ejecutar la acción modificar servicio.	Listar servicios
	Debe existir al menos un servicio adicionado para poderlo modificar.	
Descripción	Para modificar un servicio el usuario tiene que seleccionar el subsistema y luego de que el sistema liste los servicios escoger el que desea modificar. Luego debe introducir los datos que sea cambiar, excepto el nombre del servicio que este será único y luego de ser creado no se podrá modificar su valor. El sistema deberá detectar posibles errores en los datos especificados por el usuario y notificar al mismo para su corrección. En caso de que no existan errores en los datos, el sistema debe comprobar que los métodos están bien comentados, esto se realizará si el usuario escoge	

	nuevo método, de no ser así el sistema tomará el que tenía por defecto. Si todos los datos son correctos se modificará el servicio y se mostrará un mensaje de notificación
Validaciones	No se permiten valores nulos en los datos de carácter obligatorio.
Post-condiciones	Se ha modificado el servicio.
Post-requisito	El sistema lista los servicios.

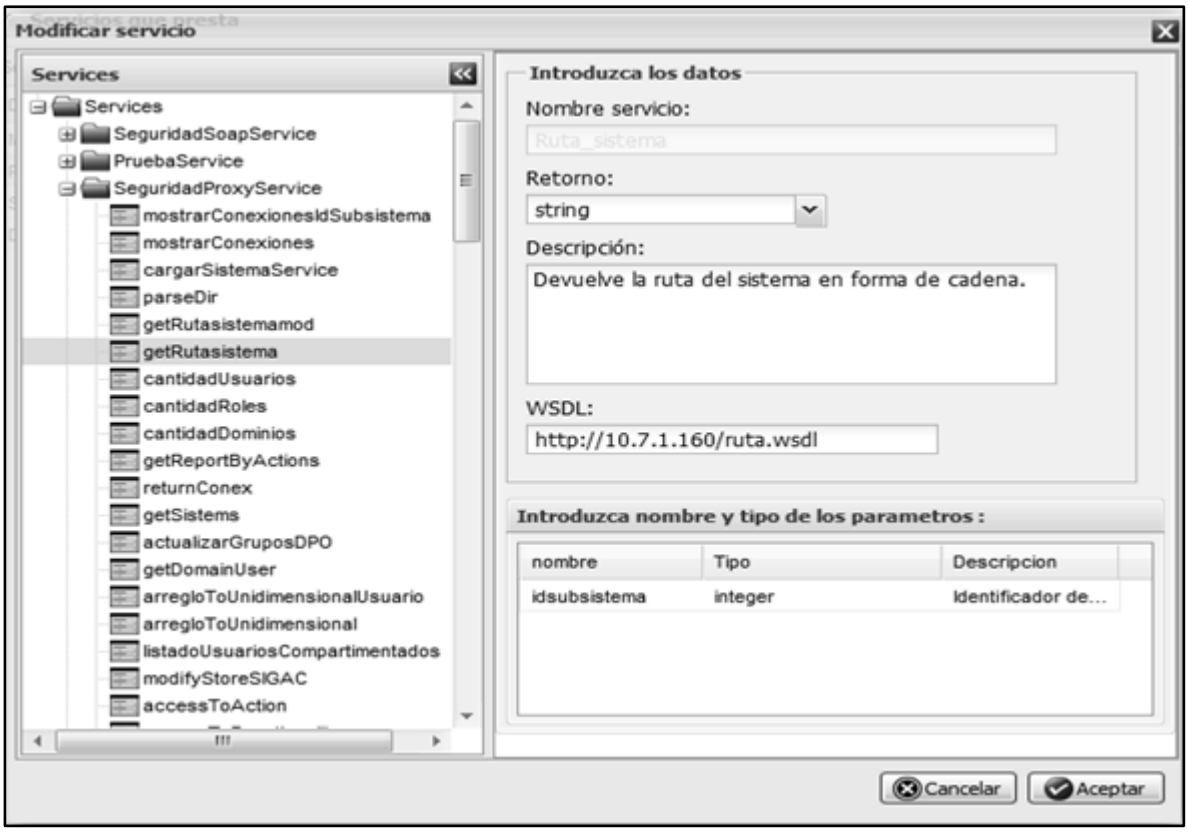


Figura 8 Interfaz de usuario del requisito Modificar servicios.

Tabla 3 Especificación del requisito Eliminar servicios.

Conceptos tratados	**Conceptos**	**Atributos**
	Servicios.	
Precondiciones	**Precondiciones**	**Pre-requisito**
	El usuario se ha autenticado en el sistema y tiene permisos para ejecutar la acción eliminar servicio.	Listar servicios
	Debe existir al menos un servicio para poderlo eliminar.	
Descripción	Para eliminar un servicio el usuario tiene que seleccionar el subsistema y luego de que el sistema liste los servicios escoger el que desea eliminar, el sistema debe mostrar un mensaje de confirmación. En caso de que el usuario confirme la acción, el sistema elimina el servicio y muestra un mensaje de notificación.	
Validaciones		
Post-condiciones	Se ha eliminado el servicio.	
Post-requisito	El sistema lista los servicios.	

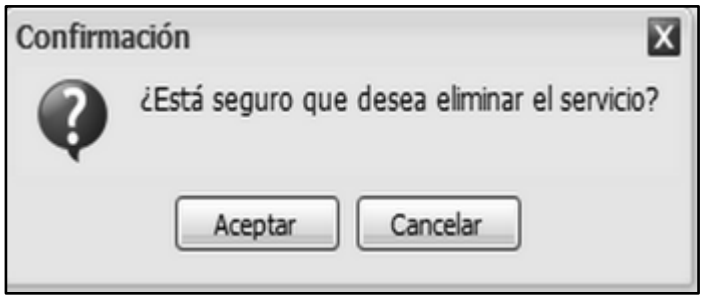


Figura 9 Interfaz de usuario del requisito Eliminar servicios.

Tabla 4 Especificación del requisito Buscar servicios.

Conceptos tratados	**Conceptos**	**Atributos**
	Servicio	Nombre del servicio
Precondiciones	**Precondiciones**	**Pre-requisito**
	El usuario se ha autenticado en el sistema y tiene permisos para ejecutar la acción buscar servicio.	Listar servicios
Descripción	Para buscar un servicio el usuario tiene que seleccionar el subsistema y luego de que el sistema liste los servicios debe introducir un criterio de búsqueda. El sistema busca el servicio por el criterio de búsqueda introducido e informa en caso de no existir ningún servicio.	
Validaciones	El nombre del servicio no debe ser nulo	
Post-condiciones		
Post-requisito		

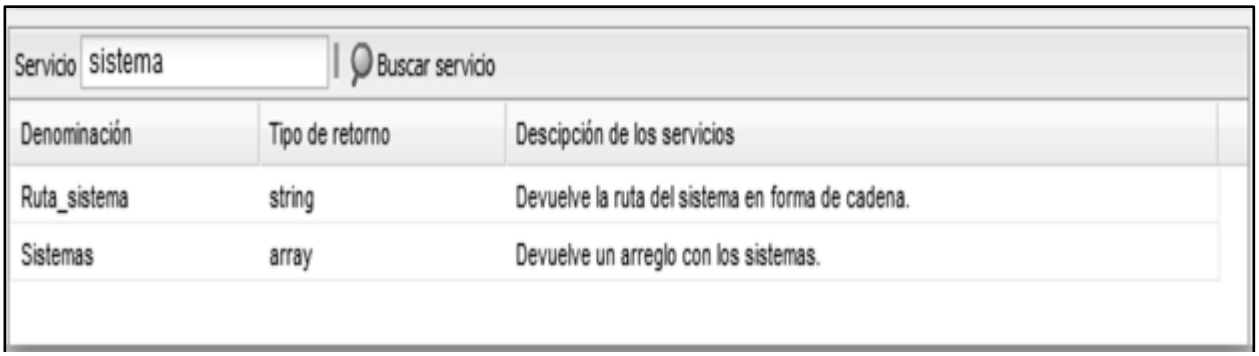


Figura 10 Interfaz de usuario del requisito Buscar servicios.

Tabla 5 Especificación del requisito Listar servicios.

Conceptos tratados	**Conceptos**	**Atributos**
	Servicio	Nombre del servicio
Precondiciones	**Precondiciones**	**Pre-requisito**
	Tiene que existir al menos un subsistema con al menos un servicio.	Cargar subsistemas
Descripción	Para listar un servicio el usuario tiene que seleccionar el subsistema y luego el sistema listará todos los servicios del subsistema seleccionado.	
Validaciones	Debe existir al menos un subsistema.	
Post-condiciones		
Post-requisito		

Figura 11 Interfaz de usuario del requisito Listar servicios.

2.2.1.2 Requisito funcional: Probar servicios

Tabla 6 Especificación del requisito Probar Servicios.

Conceptos tratados	**Conceptos**	**Atributos**
	Servicio.	Método, clase y parámetros del servicio
Precondiciones	**Precondiciones**	**Pre-requisito**
	El usuario se ha autenticado en el sistema y tiene permisos para ejecutar la acción probar servicio.	
	Debe existir al menos un servicio adicionado para que pueda ser probado.	
Descripción	Para probar un servicio el usuario tiene que seleccionar el subsistema y luego de que el sistema liste los servicios escoger el que desea probar. El sistema solicita al usuario que especifique los valores de los parámetros que requiere el servicio para su ejecución, siendo estos	

	de carácter obligatorio. Ejecuta la prueba del servicio, notifica al usuario de su resultado y modifica el estado del servicio en caso de que sea necesario.
Validaciones	Los valores de los parámetros deben ser del tipo de dato especificado. No se permiten valores nulos en los datos de carácter obligatorio.
Post-condiciones	
Post-requisito	

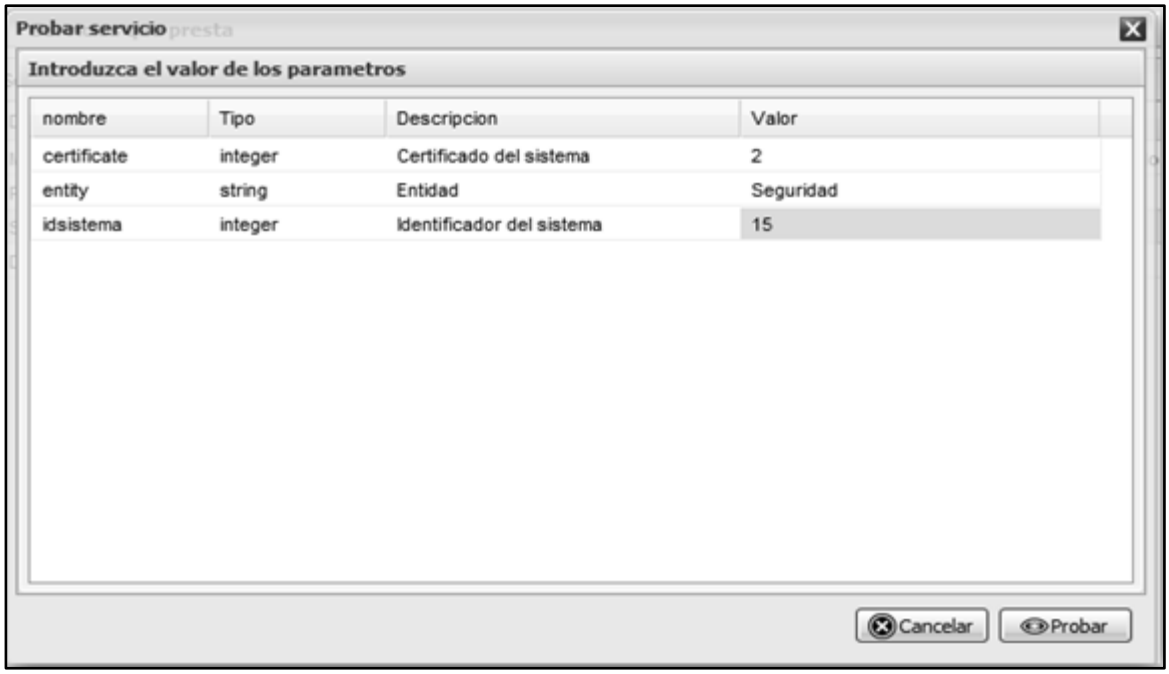


Figura 12 Interfaz de usuario del requisito Probar servicios.

2.2.2 Requisitos no Funcionales

Los requerimientos no funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con toda la funcionalidad requerida, las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación.

Como se ha mencionado con anterioridad, el presente trabajo forma parte de un proceso productivo iniciado por el CEIGE y los resultados que se obtengan formarán parte del marco de trabajo desarrollado en el mismo. Por tanto los requisitos no funcionales con los que debe cumplir la aplicación a desarrollar fueron establecidos por el centro al inicio del proceso de desarrollo, a continuación se describen los más importantes.

2.2.2.1 Rendimiento

Para un funcionamiento óptimo de la aplicación se seguirán las diferentes técnicas de elaboración de la Web. La eficiencia del producto estará determinada en gran medida por el aprovechamiento de los recursos que se disponen en el modelo cliente/servidor. La aplicación propuesta debe ser rápida y el tiempo de respuesta debe ser rápido; no mayor de 5 segundos para las actualizaciones y 20 para las recuperaciones.

2.2.2.2 Seguridad

Autenticación y Autorización (Contraseña de acceso). Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos. La atención al sistema así como la salva de la información, se realizará de forma centralizada por el administrador, además el sistema debe mostrar opción de advertencia antes de borrar cualquier elemento o información que pueda existir.

2.2.2.3 Software

Para el cliente:

- ✓ Navegador Mozilla Firefox 3.0 o superior.
- ✓ Sistema operativo Windows 98 o superior o Linux.

Para el servidor:

- ✓ Sistema operativo Linux en cualquiera de sus distribuciones.

- ✓ Un servidor Apache 2.0 o superior con módulo PHP 5.0 disponible. Este debe estar configurado con la extensión “pgsql” incluida.
- ✓ Un servidor de base de datos PostgreSQL 8.3.

2.2.2.4 Hardware

Para el servidor:

- ✓ Requerimientos mínimos: Procesador Pentium IV a 2GHz de velocidad de procesamiento y 1Gb de memoria RAM.
- ✓ Al menos 40Gb de espacio libre en disco duro.
- ✓ Tarjeta de red.

Para el cliente:

- ✓ Requerimientos mínimos: Procesador Pentium III a 1GHz con 256Mb de memoria RAM.
- ✓ Tarjeta de red.

2.2.2.5 Político-culturales

Se deberá hacer un uso correcto del idioma español en la Interfaz de la aplicación, con logotipos e imágenes que se encuentren en correspondencia con el carácter de la misma.

2.2.2.6 Confiabilidad

El subsistema debe ser confiable y preciso en la información que le suministra al usuario para evitar cualquier tipo de error. Estará disponible todo el tiempo, permitiendo el trabajo a los usuarios y las acciones de mantenimiento. Este debe ser estable, fiable y la velocidad de respuesta debe ser rápida durante la utilización del mismo. La información almacenada debe ser confiable en cuanto a su veracidad e integridad desde su recopilación y durante.

2.2.2.7 Portabilidad

El subsistema será multiplataforma (Linux-Windows) lo que permitirá ejecutarse sobre diferentes sistemas operativos sin importar sus versiones, y sin necesidad de modificar su código fuente.

2.3 Modelo Conceptual

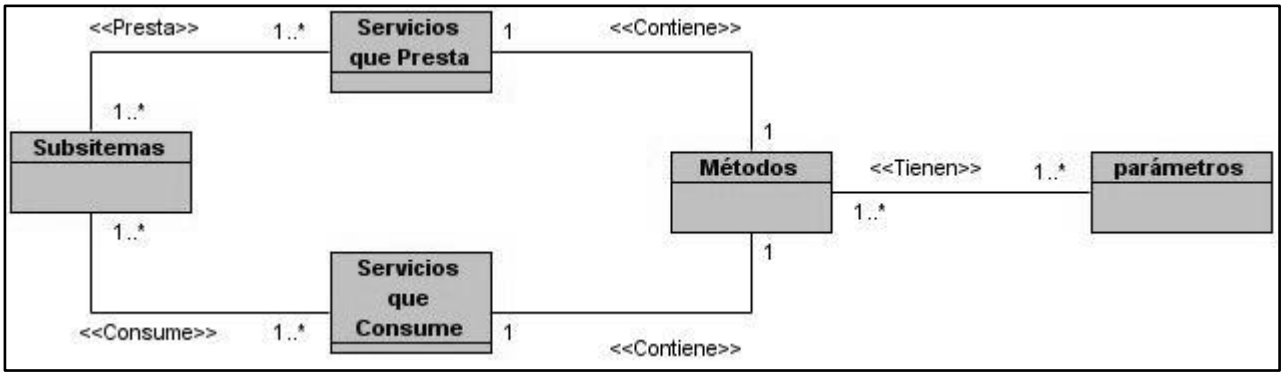


Figura 13 Modelo conceptual.

2.4 Modelo de diseño

El modelo de diseño describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Este modelo se puede utilizar para visualizar la implementación y para soportar las técnicas de programación gráfica de la aplicación. Para una mejor calidad del diseño fueron aplicados patrones de diseño durante la realización de los diagramas de clases permitiendo asignar las responsabilidades a los objetos y diseñar la colaboración entre ellos.

Dentro de este epígrafe se verán los elementos que se tuvieron en cuenta durante el diseño de la solución, dando paso así a la exposición de los resultados de este flujo de trabajo, que refleja cómo será implementado el sistema en términos de clases del diseño.

2.4.1 Diagrama de clases del diseño

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases y de las interfaces en una aplicación, contiene información como asociaciones, atributos, métodos y dependencias. A continuación se muestra el diagrama de clases del diseño basado en estereotipos web que se realizó durante el proceso de desarrollo.

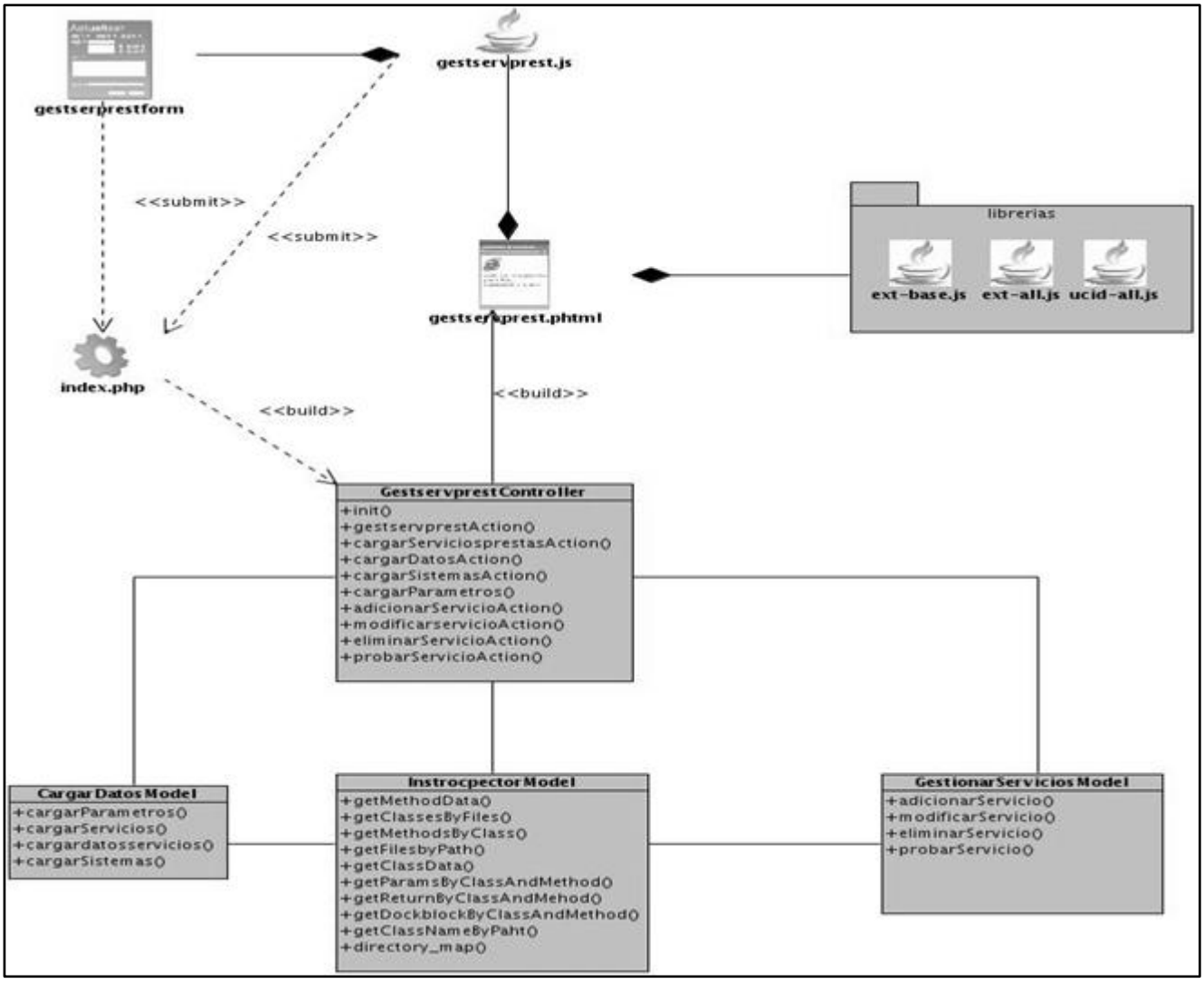


Figura 14 Diagrama de clases del diseño.

2.4.2 Patrón arquitectónico y patrones de diseño empleados

2.4.2.1 Modelo-Vista-Controlador (MVC)

MVC es un patrón de arquitectura utilizado en sistemas Web para separar los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos, permitiendo flexibilidad y facilidad a la hora de hacer futuros cambios.

La **Vista** es la información presentada al usuario. Una vista puede ser una página Web o una parte de una página.

El **Controlador** actúa como intermediario entre el Modelo, la Vista y cualquier otro recurso necesario para generar una página, es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo.

El **Modelo** representa las estructuras de datos. Típicamente el modelo de clases contendrá funciones para consultar, insertar y actualizar información de la base de datos.

Este patrón es empleado en la capa de control del marco de trabajo Sauxe y determina la estructura de los paquetes internos de los componentes a desarrollar.

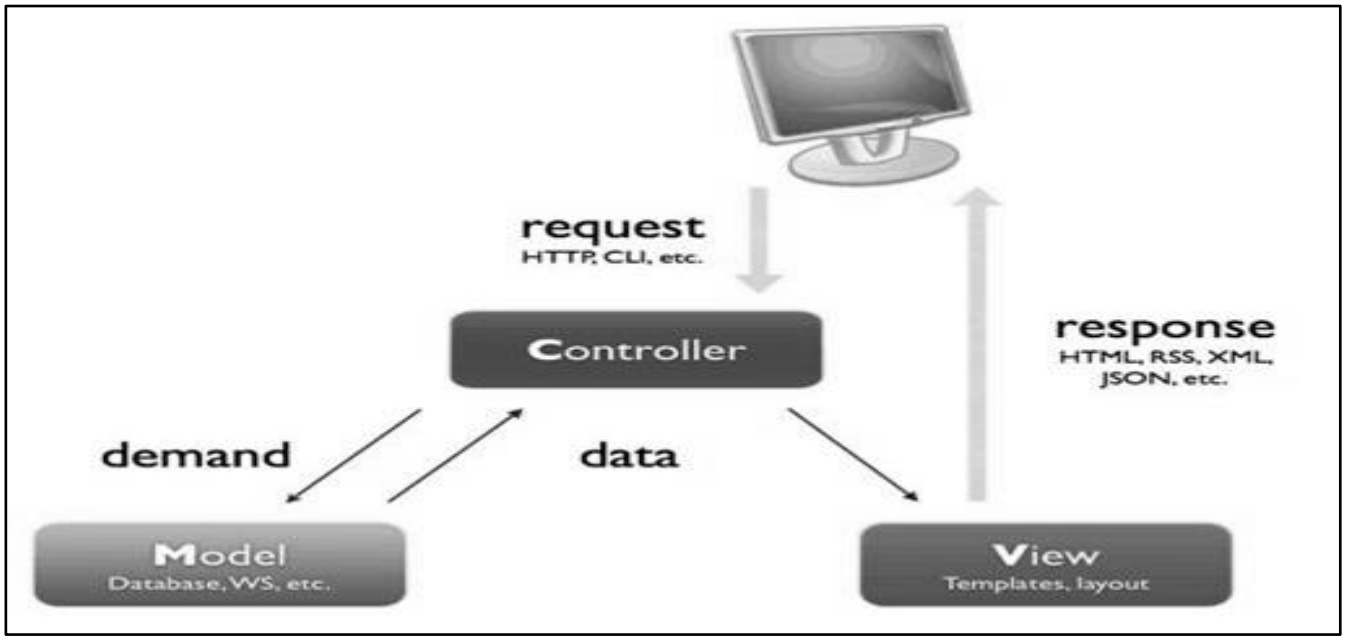


Figura 15 Modelo-Vista-Controlador.

2.4.2.2 Otros patrones utilizados

Los patrones de diseño no son más que la descripción de un problema y la solución del mismo, de forma que se pueda utilizar en diferentes contextos dando respuesta a interrogantes comunes. Durante la realización del presente solamente se utilizó en patrón Singleton.

- ✓ **Singleton:** Es un patrón creacional que tiene como propósito garantizar una única instancia de una clase, proporcionando un punto de acceso global a la misma (21). En la solución se hacen varios usos de este patrón, un ejemplo de estos lo constituye el IoC que es utilizado por los componentes para integrarse entre sí.

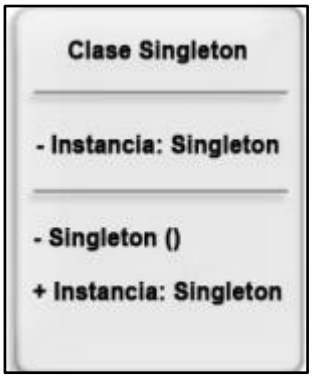


Figura 16 Patrón Singleton.

2.5 Conclusiones parciales

Durante el desarrollo del capítulo se argumentaron los aspectos fundamentales que se llevan a cabo durante el proceso de análisis, comenzando la descripción de los procesos de negocio y el levantamiento de requisitos funcionales del componente propuesto, además de los requisitos no funcionales para garantizar una ejecución eficaz de la aplicación. También se reflejó el diseño de clases para una mejor comprensión de cómo están estructuradas las clases que conforman el componente.

Capítulo 3: Implementación y prueba

En este capítulo se muestra el modelo de implementación que pone en práctica el diseño de la solución realizado en el capítulo anterior y se especifica el conjunto de validaciones y pruebas que evalúan la calidad del sistema.

3.1 Modelo de implementación

El modelo de implementación es una colección de componentes y los subsistemas que los contienen. Estos componentes incluyen: ficheros de código fuente, y otros tipos de ficheros necesarios para la implantación y despliegue del sistema. Describe también, como se organizan los componentes de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y como dependen de los componentes unos de otros. (22)

3.1.1 Codificación

- ✓ **Nomenclatura**

Clases en las vistas

Se definió para las clases que se encuentran dentro de Views que el nombre se escribirá sin ningún sufijo o prefijo. Los nombres se escribirán en notación camello (Camel Case). Ejemplo: GestServPrest.js

Clases Controladoras

Se definió para las clases controladoras que después del nombre se les colocará como sufijo la palabra:"Controller". Ejemplo: GestservprestController.php.

Clases de los modelos

Business (Negocio) Se definió para las clases que se encuentran dentro de Business que después del nombre se les colocará como sufijo la palabra:"Model" Ejemplo: InstrocpectorModel.php

Domain (Dominio) y Bases del Dominio (Generated).El sistema no cuenta con clases de este tipo debido a que los datos se persisten en un xml, al no interactuar con ninguna base de datos estas clases no existen.

Nomenclatura de las funciones

Se definió que el nombre de las funciones se escribe con la primera palabra en minúscula, para el caso, específico de que sea un nombre compuesto se empleará la notación CamelCasing (notación camello) y de forma tal que al leerlo se conozca el objetivo de la misma. Ejemplo: adicionarServicios(). En caso particular que la función sea una acción de la clase controladora u otra se escribe después del nombre de la función la palabra "Action" como sufijo. Ejemplo: cargarservprestAction().

Nomenclatura de las variables

Se definió que el nombre a emplear para las variables se escribe en minúscula y en caso de ser un nombre compuesto se escriben ambas palabras en minúscula separadas por un guión bajo.

Ejemplo:

variable.

variable_compuesta.

- ✓ **Normas de los comentarios**

Con el objetivo de garantizar un código más legible y reutilizable se convierte en una necesidad el comentariado de todo el código que se implemente dentro del desarrollo, con el objetivo de aumentar su mantenibilidad a lo largo del tiempo.

Nomenclatura de los comentarios

Se definió el uso de comentarios claros y precisos que ayudarán a una mejor comprensión del código implementado para que se entiendan sus objetivos específicos.

En las clases

Para implementar una clase se escribe una pequeña descripción donde se expliquen los propósitos de la misma de forma general, así como su autor y sistema al que pertenece. Dicha descripción se escribe de la siguiente forma:

```
/**
* Nombre de la clase *
* Descripción *
* @author *
```

* @package *(módulo)
* @subpackage *(sub módulo)
* @copyright *
* @version (versión - parche)
*/

```
<?php
/**
*Nombre de la clase Gestservprest
*Descripcion * Esta clase es la encargada de gestionar los servicios
*@author * Rene Hernandez Morera
*@package * Herramientas
*@subpackage * configuracion  de IoC
*@copyringht * CEIGE
*@version 1.0
*/
```

Figura 17 Ejemplo de comentario de clase

Comentarios por líneas

Se utilizan también los comentarios para líneas de código en específico que ayudan a aclarar la complejidad de algunos algoritmos. Para ellos se utiliza //, y seguidamente el comentario que describe la línea de código, tal y como se representa a continuación.

```
if (!file_exists($direccion)) {              //Preguntar si existe un arvhivo en esa direccion

 $dom = new DOMDocument('1.0', 'utf-8');     //Creo un objeto de tipo DOMDocument
 $xml = $dom->createElement("servicios",''); //Creo el nombre de la primera etiqueta
 $dom->appendChild($xml);                    //Se la adiciono al objeto que cree anteriormente
 $fs = fopen ( $direccion, "a+");             //Abro el fichero dada la direccion
 $f = fputs($fs, $dom->saveXML());            //Escribo el  XMl en la direccion
 $f = fclose($fs);                             // Cierro el fichero
 }
```

Figura 18 Ejemplo de comentario por líneas de código.

3.1.2 Métricas de diseño.

Para entender mejor el concepto de métrica es necesario aclarar que los términos, métricas, medición y medida no tienen el mismo significado.

Medida: Proporciona una indicación cuantitativa de la extensión, cantidad, dimensiones, capacidad o tamaño de algunos atributos de un proceso o producto. (23)

Medición: La medición es el acto de determinar una medida. (23)

Métrica: Es una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado. (23)

Se definen las métricas de software como "La aplicación continua de mediciones basadas en técnicas para el proceso de desarrollo del software y sus productos, para suministrar información relevante a tiempo, así el administrador junto con el empleo de estás técnicas mejorará el proceso y sus productos" (24).

Para una definición más completa debe incluirse los servicios relacionados al software como la respuesta a los resultados del cliente: Ver Figura 19

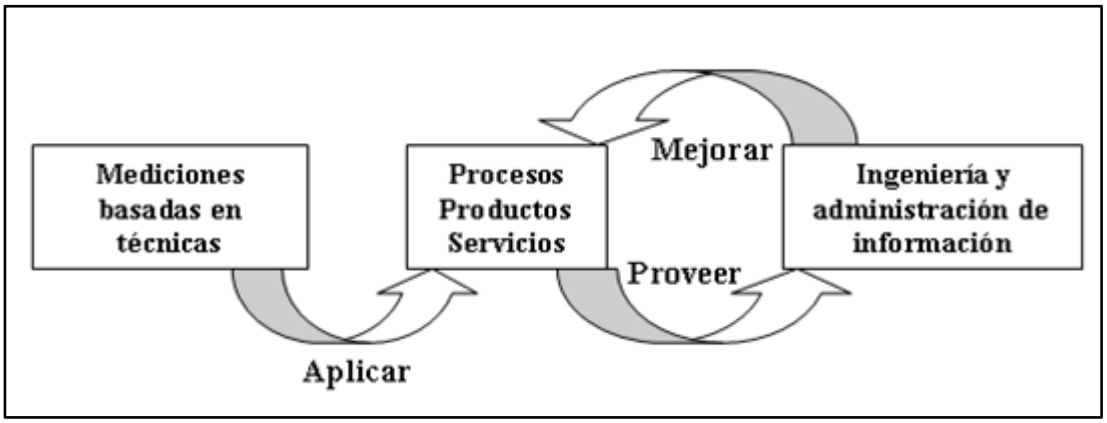


Figura 19 Concepto de Métricas.

Atributos de la calidad que abarcan.

A continuación se detallan los atributos que se miden en la validación de la solución:

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

- ✓ **Complejidad del diseño**: Consiste en la complejidad que posee una estructura de diseño de clases.
- ✓ **Complejidad de implementación**: Consiste en el grado de dificultad que tiene que implementar un diseño de clases determinado.
- ✓ **Reutilización**: Consiste en el grado de reutilización que presente una clase o estructura de clase, dentro de un diseño de software.
- ✓ **Acoplamiento**: Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.
- ✓ **Complejidad de mantenimiento**: Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.
- ✓ **Cantidad de pruebas**: Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (Componente, módulo, clase, conjunto de clases, etc.) diseñado.
- ✓ **Eficiencia**: Se refiere al esfuerzo que un usuario tiene que hacer para conseguir un objetivo
- ✓ **Efectividad**: Variables que permiten medir la exactitud y la plenitud con la que se alcanzan los objetivos de una tarea concreta.

Las **métricas** aplicadas al diseño descrito en el capítulo anterior para la evaluación de la solución propuesta son las siguientes:

Tamaño operacional de clase (TOC): Está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

Tabla 7 Atributos de calidad evaluados por la métrica TOC.

Atributo de calidad	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.

Reutilización	Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

Tabla 8 Criterios de evaluación para la métrica TOC.

Atributo	Categoría	Criterio
Responsabilidad	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Complejidad implementación	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio

Relaciones entre clases (RC): Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

Tabla 9 Atributos de calidad evaluados por la métrica RC.

Atributo de calidad	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del Acoplamiento de la clase.
Complejidad de mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

Tabla 10 Criterios de evaluación para la métrica RC.

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

3.2 Resultados obtenidos de la aplicación de la métrica TOC

Una vez obtenidos los resultados de la evaluación del instrumento de medición de la métrica TOC, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que el 50% de las clases empleadas en el sistema posee 5 operaciones o menos lo que conlleva a evaluaciones positivas de los atributos de calidad involucrados (responsabilidad, complejidad de implementación y reutilización). A continuación se muestran los resultados obtenidos.

Tabla 11 Instrumento de evaluación de la métrica TOC.

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
GestservprestController	11	Media	Media	Media
InstrospectorModel	12	Media	Media	Media
CargardDatosModel	4	Baja	Baja	Alta
GestionarServicioModel	4	Baja	Baja	Alta

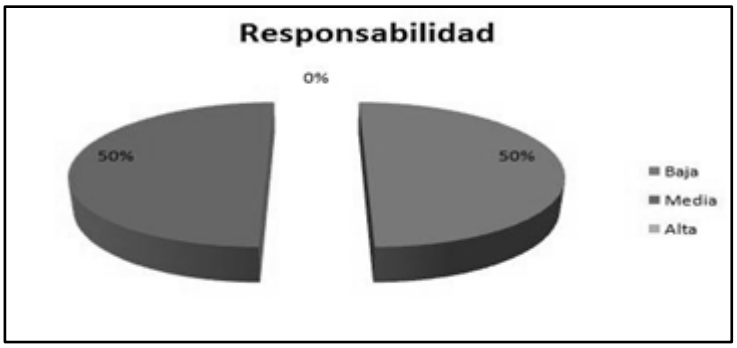


Figura 20 Resultados de la evaluación de la métrica TOC para el atributo Responsabilidad.

Figura 21 Resultados de la evaluación de la métrica TOC para el atributo Reutilización

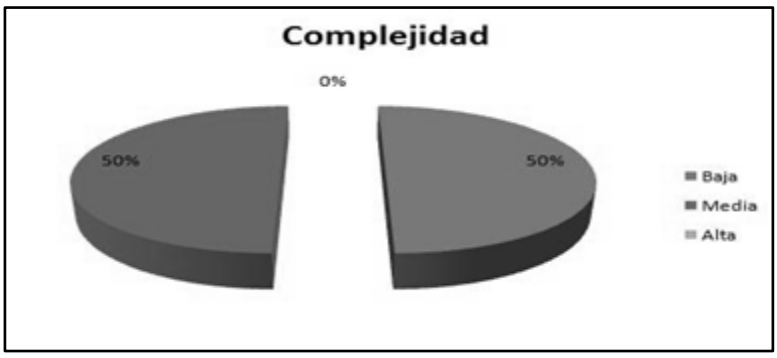


Figura 22 Resultados de la evaluación de la métrica TOC para el atributo Complejidad.

3.3 Resultados obtenidos de la aplicación de la métrica RC

Una vez obtenidos los resultados de la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que el 50% de las clases empleadas posee menos de 3 dependencias de otras clases lo que conlleva a evaluaciones positivas de los atributos de calidad involucrados (acoplamiento, complejidad de mantenimiento, cantidad de pruebas y reutilización). A continuación se muestran los resultados obtenidos.

Tabla 12 Instrumento de evaluación de la métrica RC.

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
GsetservprestController	9	Alta	Media	Media
InstrospectorModel	12	Alta	Alta	Alta
CargardDatosModel	0	Ninguno	Baja	Baja
GestionarServicioModel	0	Ninguno	Baja	Baja

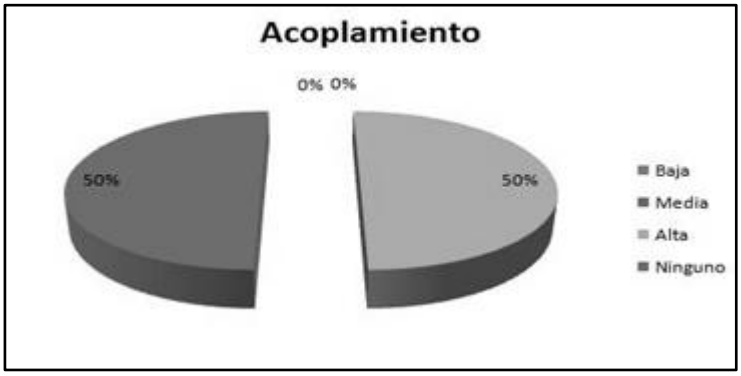


Figura 23 Resultados de la evaluación de la métrica RC para el atributo Acoplamiento.

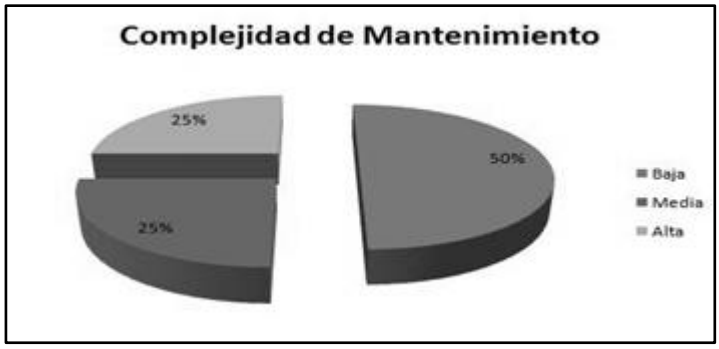


Figura 24 Resultados de la evaluación de la métrica RC para el atributo Complejidad de Mantenimiento.

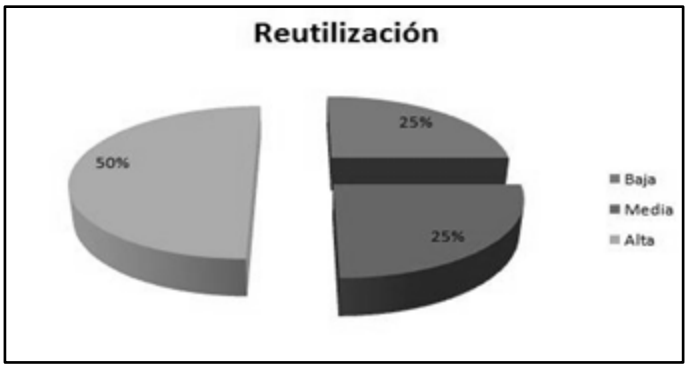


Figura 25 Resultados de la evaluación de la métrica RC para el atributo Reutilización.

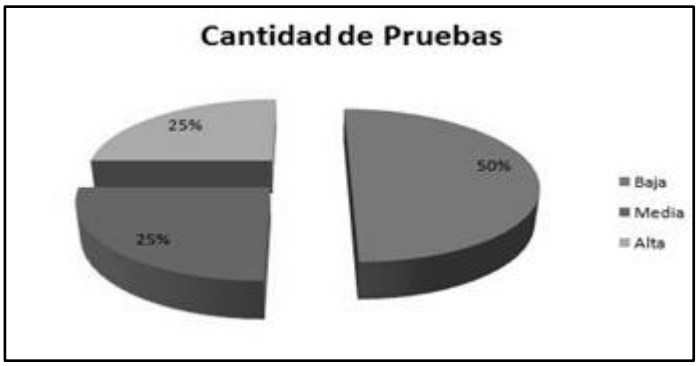


Figura 26 Resultados de la evaluación de la métrica RC para el atributo Cantidad de Pruebas.

3.4 Matriz de inferencia de indicadores de calidad

La matriz inferencia de indicadores de calidad, también llamada matriz de cubrimiento permite conocer si los resultados obtenidos de las relaciones atributo/métrica es positivo o no, llevando estos resultados a una escalabilidad numérica donde, si los resultados son positivos se le asigna el valor de 1, si son

negativos toma valor 0 y si no existe relación es considerada como nula y es representada con un guión simple (-). Luego se puede obtener un resultado

A continuación se muestran los resultados obtenidos.

Tabla 13 Resultados de la evaluación de la relación atributo/métrica.

Atributos\Métricas	TOC	RC	Promedio
Responsabilidad	1	(-)	1
Complejidad de Implementación	1	(-)	1
Reutilización	1	1	1
Acoplamiento	(-)	1	1
Complejidad de Mantenimiento	(-)	1	1
Cantidad de pruebas	(-)	1	1

Tabla 14 Rango de valores para la evaluación de la relación atributo/métrica

Categoría	Rango de valores
Malo	<=0.4
Regular	>0.4 y <0.7
Bueno	>=0.7

Figura 27 Resultados obtenidos de la evaluación de los atributos de calidad.

3.5 Diagrama de despliegue

Un diagrama de despliegue muestra la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. A continuación se muestra una propuesta de despliegue, así quedará distribuido el subsistema.

Figura 28 Diagrama de despliegue.

3.6 Diagrama de componentes

La solución que se propuso consta de un solo componente llamado Configuración de IoC el cual interactuar con otros componentes del marco de trabajo Sauxe. A continuación se muestra el diagrama de componentes elaborado.

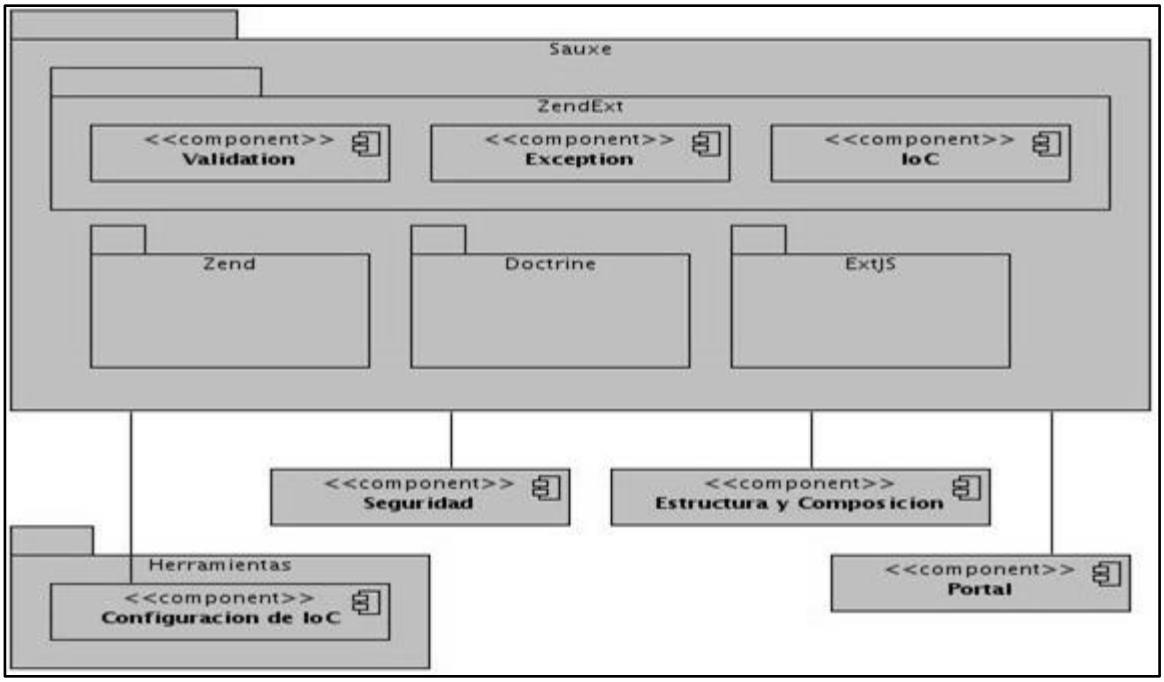


Figura 29 Diagrama de componentes

3.7 Pruebas estructurales o de caja blanca

Las llamadas pruebas de caja blanca, también conocidas como técnicas de caja transparente o de cristal, proponen una manera de diseñar los casos de prueba centrándose en el comportamiento interno y la estructura del programa. De esta manera se examina solo la lógica interna del programa, dejando fuera los aspectos de rendimiento del mismo.

A partir del empleo de estas pruebas se podrá diseñar los casos de prueba que comprueben que todas las sentencias del sistema se ejecuten al menos una vez, además de todas las condiciones. Para lograrlo lo primero es enumerar las sentencias del código y a partir del mismo se construye el grafo de flujo correspondiente.

```php
<?php
public  function cargarsistemasAction()
    {
    $certif = $this->global->Perfil->certificado; //1
    $entidad = $this->global->Estructura->idestructura; //1

    $sistemas = $this->integrator->seguridad->getSistems($certif, $entidad, $this->_request->getPost('node')); 1
       if(count($sistemas)) //2
          {
              foreach($sistemas as $valores=>$valor) //3
               {
                   $sistemaArr[$valores]['id'] = $valor['id']; //4
                   $sistemaArr[$valores]['text'] = $valor['text']; //4
                   $sistemaArr[$valores]['abreviatura'] = $valor['abreviatura']; //4
                   $sistemaArr[$valores]['descripcion'] = $valor['descripcion']; //4
                   $sistemaArr[$valores]['idpadre'] = $valor['idpadre']; //4
                   $sistemaArr[$valores]['icono'] = $valor['icono']; //4
                   $sistemaArr[$valores]['leaf'] = true; //4
               } //5

                echo json_encode ($sistemaArr);return; //6
          }
       else //7
       {
           $sist = $sistemas->toArray(); //8
           echo json_encode($sist);return; //8
       }
    } //9
```

Figura 30 Código fuente de la funcionalidad cargarsistemasAction.

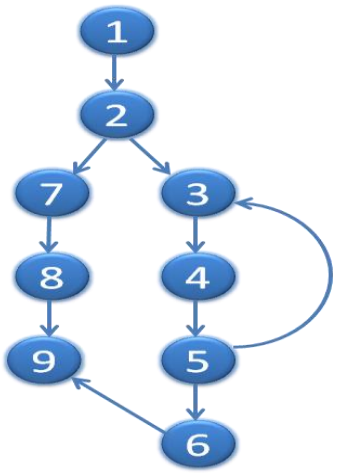


Figura 31 Grafo de flujo correspondiente a la funcionalidad cargarsistemasAction.

Luego de haber construido el grafo se realiza el cálculo de la complejidad ciclomática mediante las tres fórmulas descritas a continuación, las cuales deben arrojar el mismo resultado para asegurar que el cálculo de la complejidad es correcto.

1. **V (G) = (A – N) + 2**

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

V (G) = (10 - 9) + 2

V (G) = 3

2. **V (G) = P + 1**

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

V (G) = 2 + 1

V (G) = 3

3. **V (G) = R**

Siendo "R" la cantidad total de regiones, para cada formula "V (G)" representa el valor del cálculo.

V (G) = 3

El cálculo efectuado anteriormente sobre las fórmulas ha dado el mismo valor, dando como resultado 3, lo que indica que existen 3 posibles caminos por donde el flujo puede circular, y determina el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo:

- ✓ **Camino básico #1:** 1-2-3-4-5-6-9
- ✓ **Camino básico #2:** 1-2-3-4-5-3-4-5-6-9
- ✓ **Camino básico #3 :** 1-2-7-8-9

Para cada camino se realiza un caso de prueba.

- ✓ **Caso de prueba para el Camino básico #1:**

Descripción: Los datos de entrada se obtienen a través del método Post y de las variables globales Perfil y Estructura y cumplirán con los siguientes requisitos: El certificado, la entidad y el nodo tienen que ser cadenas de números.

Condición de ejecución: Ningún dato debe ser nulo.

Obtenido: $certificado = $certif, $entidad = $entidad, $nodo = $nodo.
Resultados esperados: Un sistema

✓ **Caso de prueba para el Camino básico #2:**

Descripción: Los datos de entrada se obtienen a través del método Post y de las variables globales Perfil y Estructura y cumplirán con los siguientes requisitos: El certificado, la entidad y el nodo tienen que ser cadenas de números.
Condición de ejecución: Ningún dato debe ser nulo.
Obtenido: $certificado = $certif, $entidad = $entidad, $nodo = $nodo.
Resultados esperados: Un arreglo de sistemas

✓ **Caso de prueba para el Camino básico #3:**

Descripción: Los datos de entrada se obtienen a través del método Post y de las variables globales Perfil y Estructura y cumplirán con los siguientes requisitos: El certificado, la entidad y el nodo tienen que ser cadenas de números.
Condición de ejecución: Ningún dato debe ser nulo.
Obtenido: $certificado = $certif, $entidad = $entidad, $nodo = $nodo.
Resultados esperados: Un arreglo vacío.

3.8 Diseño de casos de prueba

Tabla 15 Escenarios de prueba.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Gestionar Servicios.	Permite adicionar un servicio al XML IoC	EP 1.1: Listar Servicios.	Desplegar el botón Inicio y buscar el escenario donde se desarrolla que se encuentra en Herramientas, IoC,

			Servicios que presta.
		EP 1.2: Adicionar Servicio	En este escenario se debe seleccionar un subsistema. Una vez seleccionado este el sistema listará todos los servicios del subsistema seleccionado. Presionar el botón Adicionar e introducir los datos que requiere el nuevo servicio, así como seleccionar una clase Services y posteriormente un método dentro de esta clase y presionar el botón aceptar o aplicar. Al final el sistema muestra una interfaz de confirmación.
		EP 1.3: Modificar Servicio	En este escenario se debe seleccionar un servicio en el grid y posteriormente presionar el botón Modificar, debe introducir los datos que sean necesarios modificar del servicio. El sistema desplegará el árbol y marcará automáticamente un método, se es

			necesario sustituirlo por otro solo tiene que una clase Services y posteriormente un método dentro de esta clase y presionar el botón modificar o aplicar. Al final el sistema muestra una interfaz de confirmación.
		EP 1.4: Eliminar Servicio	En este escenario se debe seleccionar un servicio en el grid. Y presionar el botón Eliminar, al final el sistema muestra una interfaz de confirmación
2: Probar Servicios.		EP 1.1: Probar Servicio	En este escenario se debe seleccionar un servicio en el grid. Y se presiona el botón probar, posteriormente se introducen los parámetros para probar el servicio y se presionar el botón probar.

3.9 Conclusiones parciales

En este capítulo fueron expuestos los artefactos generados como parte del modelo de implementación de la solución propuesta, tales como el diagrama de componentes y los estándares de codificación. Se realizó una validación del diseño expuesto en el capítulo anterior mediante la aplicación de métricas para la evaluación de atributos de calidad, lo cual permitió valorar el diseño propuesto de muy bueno dado los excelentes resultados obtenidos. Además se describieron las pruebas estructurales y funcionales realizadas que permitieron comprobar el correcto funcionamiento del sistema.

Conclusiones

Con el desarrollo del componente utilizando las herramientas y tecnologías propuestas se logró brindar a los desarrolladores del CEIGE así como a los de otros grupos de trabajo que utilicen Sauxe para el desarrollo de sus productos una herramienta que posibilite la configuración visual de los servicios de integración. Esto fue posible gracias al estudio del arte realizado sobre las herramientas, tecnologías y técnicas aplicadas a nivel mundial sobre la configuración del patrón IoC, lo cual sentó las condiciones necesarias para el posterior análisis, diseño e implementación del componente realizado. Los resultados obtenidos fueron analizados mediante la validación de su diseño y una etapa de pruebas que reveló el cumplimiento de los requerimientos propuestos. El componente hace posible que los desarrolladores agilicen el proceso de configuración de los servicios de integración, acortando el tiempo de desarrollo y facilitando dicho proceso de configuración haciéndolo menos engorroso.

Recomendaciones

- Se recomienda que el resultado alcanzado sea implantado como uso cotidiano e introducido en el desarrollo de las aplicaciones web de gestión que se desarrollen sobre el Marco de Trabajo Sauxe.

- Se recomienda continuar perfeccionando la herramienta a partir de los nuevos requisitos que puedan surgir como resultado de su explotación.

- Se propone que se utilice el componente como punto de partida para la implementación de futuros componentes relacionados con la configuración de servicios de integración.

Bibliografía

1. Kameleweb. *Kameleweb, web based management home page.* [En línea] OneXperience, S.L, 2010. [Citado el: 12 de 2 de 2011.] http://es.kameleweb.com/aplicaciones-web-a-medida.htm.
2. **Daylin Real Madrigal, Reinaldo Alain Hernández Valdés.** *Sistema de Log de Auditoría.* La Habana : s.n., 2008.

3. **Microsotf.** Microsoft MSDN. *Microsoft MSDN.* [En línea] Microsoft . [Citado el: 10 de 2 de 2011.] http://msdn.microsoft.com/es-ar/library/cc707904.aspx.

4. **Informática, Centro Técnico de.** *El Framework de desarrollo del Consejo Superior de Investigaciones Científicas.* Sevilla : Centro Técnico de Informática, 2006.

5. **Programadores, Comunidad de.** Comunidad Programadores. *Comunidad programadores.* [En línea] [Citado el: 2011 de 2 de 12.] http://www.lawebdelprogramador.com/temas/enlace.php?idp=3545&id=44&texto=java. .

6. symfony.es. *symfony.es.* [En línea] febrero de 2011. [Citado el: 12 de febrero de 2011.] http://www.symfony.es/2009/04/02/inyeccion-de-dependencias-en-symfony-2/.

7. librosweb.es. *librosweb.es.* [En línea] [Citado el: 12 de 2 de 2011.] http://www.librosweb.es/symfony/capitulo1/symfony_en_pocas_palabras.html.

8. **Alcántara, Ing. Samuel Mestanza.** Scribd. *Scribd.* [En línea] Laureate (International Universities). [Citado el: 12 de 2 de 2011.] http://es.scribd.com/doc/40764319/IoC-Unity-Sem-10.

9. **Microsoft.** Microsoft MSDN. *Microsoft MSDN.* [En línea] Micrsoft, abril de 2010. [Citado el: 12 de 2 de 2011.] http://msdn.microsoft.com/en-us/library/ff660846%28PandP.20%29.aspx.

10. **Pérez, Mileidy Magalys Sarduy.** *Propuesta de modelo de desarrollo de software tecnológico del centro de soluciones de gestión.* Ciudad Habana : UCI, 2009.

11. PostgresSQL - es Portal en español sobre PostgresSQL. [En línea] [Citado el: 2 de 12 de 2011.] http://www.postgresql-es.org/sobre_postgresql.

12. **symfony.es.** Netbeans. [En línea] [Citado el: 16 de 12 de 2010.] http://www.symfony.es/2009/10/05/netbeans-ya-incluye-soporte-para-symfony/.

Bibliografía.

13. **Marley, Jimi.** programacion en castellanos. *Programacion en Castellano.* [En línea] 12 de 12 de 2010. [Citado el: 2011 de 1 de 10.] http://www.programacion.com/articulo/programacion_orientada_a_objetos_279.

14. **Grupo de documentación de PHP.** *Manual de PHP*. 2001.

15. **Pérez, Javier Eguíluz.** LibrosWeb.es. *LibrosWeb.com*. [En línea] Introducción a JavaScript. [Citado el: 10 de 1 de 2011.] http://www.librosweb.es/javascript/.

16. **Gómez Baryolo, Oiner, Tenrero Cabrera, Marianela y Nemuris, Silega Martínez.** *Plantilla Registro de la Propiedad intelectual (Sauxe).* La Habana : s.n., 2008.

17. **Gonzales, Benjamín.** Zend Frameworks DES.com. *Zend Frameworks DES.com*. [En línea] Zend Technologies Inc, 28 de 11 de 2010. [Citado el: 14 de 1 de 2011.] http://manual.zfdes.com/es/introduction.overview.html.

18. **Frederick, Shea, Ramsay, Colin y Blades, Steve 'Cutter'.** *Learning Ext JS.* Birmingham - Mumbai : PACKT, 2008. 978-1-847195-14-2.

19. *Un método para el diseño de la base de datos a partir del modelo orientado a objetos.* **Hernández González, Dra. Anaisa.** 4, México DF : s.n., 2004, Computación y Sistemas, Vol. 7. 1405-5546.

20. **Barros, Oscar.** Reingeniería de Procesos de negocio. Chile : Editorial Dolmen, 1994.

21. **Gamma, Erich, y otros.** *Design Patterns: Elements of Reusable Object-Oriented Software.* s.l. : Addison-Wesley, 1994. 0-201-63361-2.

22. **Letelier, Patricio.** Rational Unified Process (RUP). *Rational Unified Process (RUP).* Valencia : Universidad Politécnica de Valencia.

23. **Pressman, Roger S.** *Ingeniería del Software un Enfoque Práctico* . Mexico : México McGraw-Hill/Interamericana de Mexico , MEXICO , 2005. 970-10-5473-3 .

24. **González Doria, H.** *Las Métricas de Software y su Uso en la Región.* Mexico : Departamento de Ingeniería en Sistemas Computacionales, 2001.

25. **Microsotf.** msdn. *msdn.* [En línea] Microsotf. [Citado el: 10 de 2 de 2011.] http://msdn.microsoft.com/es-ar/library/cc707904.aspx.

26. **Amaro Calderón, Sarah Dámaris Valverde Rebaza,Jorge Carlos.** *Metodologías Ágiles.* Perú : Trujillo, 2007.

27. **Gómez Baryolo, Oiner, Tenrero Cabrera, Marianela y Nemuris, Silega Martínez.** *Plantilla Registro de la Propiedad intelectual (Sauxe).* La Habana : s.n., 2008.

28. **Group, Object Management.** Unified Modeling Language. *Unified Modeling Language.* [En línea] OMG, 2010 de 10 de 21. [Citado el: 10 de 1 de 2011.] http://www.uml.org/.

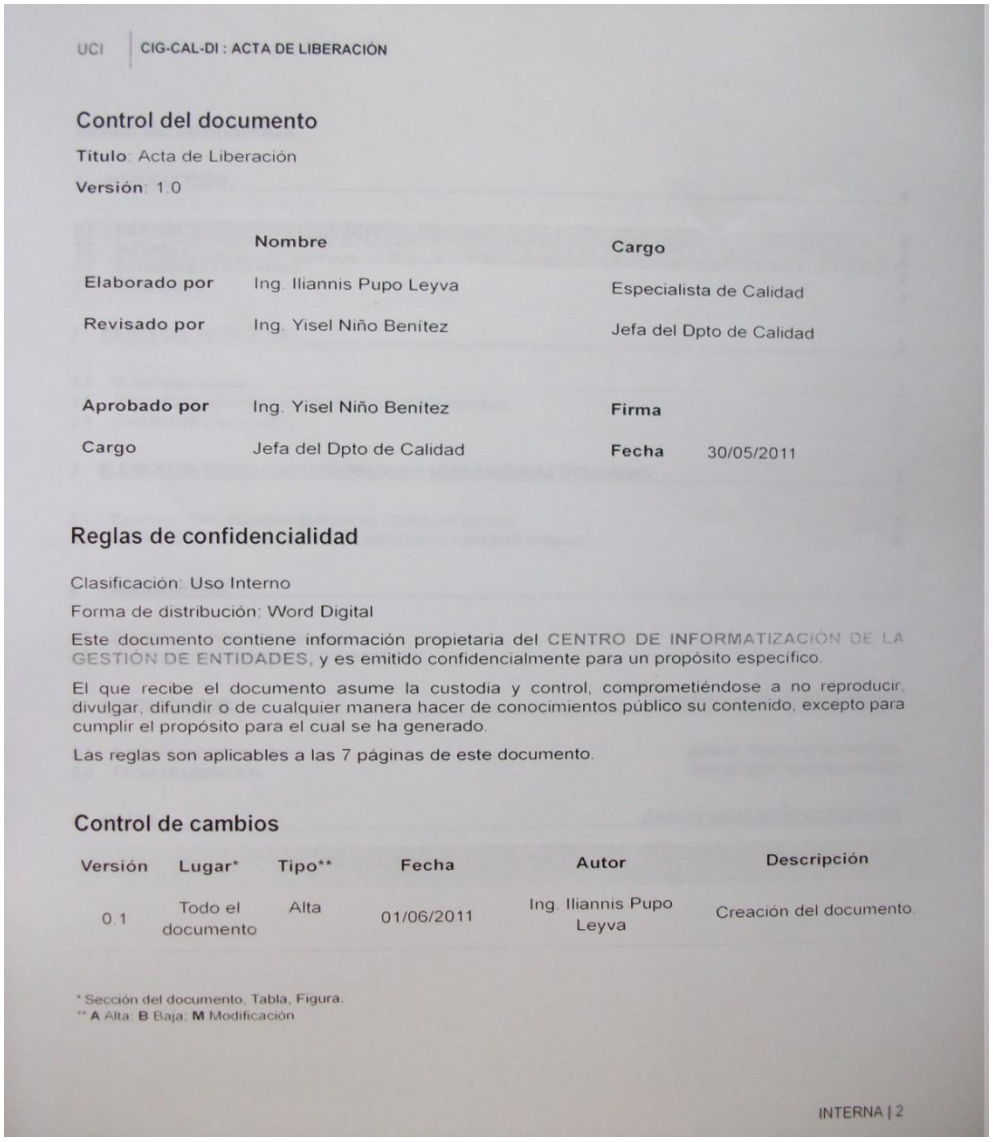
29. **Pérez, Javier Eguíluz.** LibrosWeb.es. *LibrosWeb.com*. [En línea] Introducción a JavaScript. [Citado el: 10 de 1 de 2011.] http://www.librosweb.es/javascript/.

30. **Riola, Jose Carlos Carvajal.** *Metodologias Ágiles: Herramientas y modelo de desarrollo para aplicaciones.* Barcelona, España : s.n., 2008.

31. **Sanchez, María A. Mendoza.** Informatízate. *Informatízate.* [En línea] 7 de Junio de 2007. [Citado el: 21 de Diciembre de 2010.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

Anexos

Anexo 1: Acta de Liberación del producto.



UCI | CIG-CAL-DI : ACTA DE LIBERACIÓN

Control del documento

Título: Acta de Liberación

Versión: 1.0

	Nombre	Cargo
Elaborado por	Ing. Iliannis Pupo Leyva	Especialista de Calidad
Revisado por	Ing. Yisel Niño Benítez	Jefa del Dpto de Calidad

Aprobado por	Ing. Yisel Niño Benítez	**Firma**	
Cargo	Jefa del Dpto de Calidad	**Fecha**	30/05/2011

Reglas de confidencialidad

Clasificación: Uso Interno

Forma de distribución: Word Digital

Este documento contiene información propietaria del CENTRO DE INFORMATIZACIÓN DE LA GESTIÓN DE ENTIDADES, y es emitido confidencialmente para un propósito específico.

El que recibe el documento asume la custodia y control, comprometiéndose a no reproducir, divulgar, difundir o de cualquier manera hacer de conocimientos público su contenido, excepto para cumplir el propósito para el cual se ha generado.

Las reglas son aplicables a las 7 páginas de este documento.

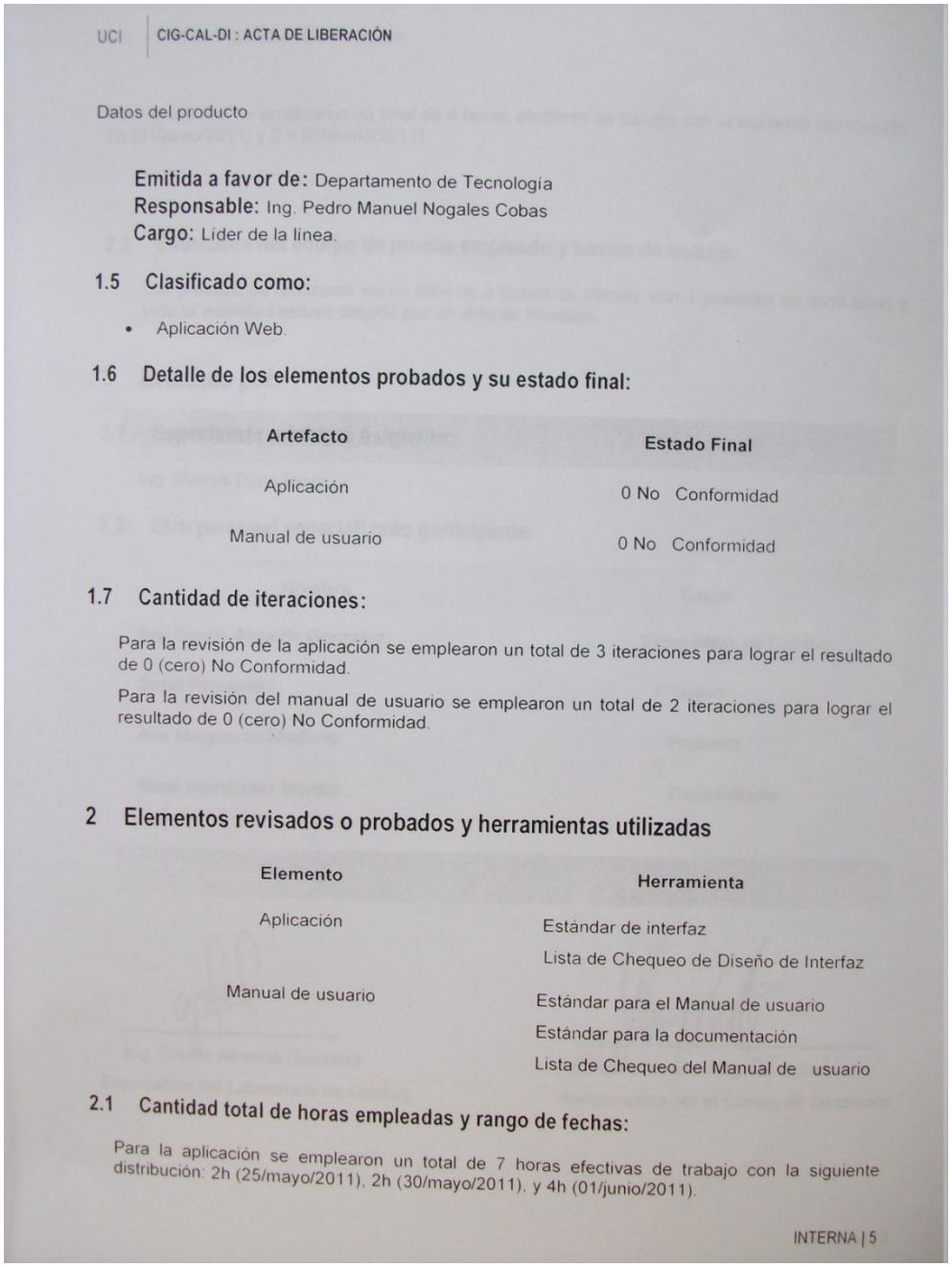
Control de cambios

Versión	Lugar*	Tipo**	Fecha	Autor	Descripción
0.1	Todo el documento	Alta	01/06/2011	Ing. Iliannis Pupo Leyva	Creación del documento.

* Sección del documento, Tabla, Figura.
** **A** Alta; **B** Baja; **M** Modificación

INTERNA | 2

Figura 32 Acta de Liberación del producto, página 1 de 3.



UCI | CIG-CAL-DI : ACTA DE LIBERACIÓN

Datos del producto

Emitida a favor de: Departamento de Tecnología
Responsable: Ing. Pedro Manuel Nogales Cobas
Cargo: Líder de la línea.

1.5 Clasificado como:

- Aplicación Web.

1.6 Detalle de los elementos probados y su estado final:

Artefacto	Estado Final
Aplicación	0 No Conformidad
Manual de usuario	0 No Conformidad

1.7 Cantidad de iteraciones:

Para la revisión de la aplicación se emplearon un total de 3 iteraciones para lograr el resultado de 0 (cero) No Conformidad.

Para la revisión del manual de usuario se emplearon un total de 2 iteraciones para lograr el resultado de 0 (cero) No Conformidad.

2 Elementos revisados o probados y herramientas utilizadas

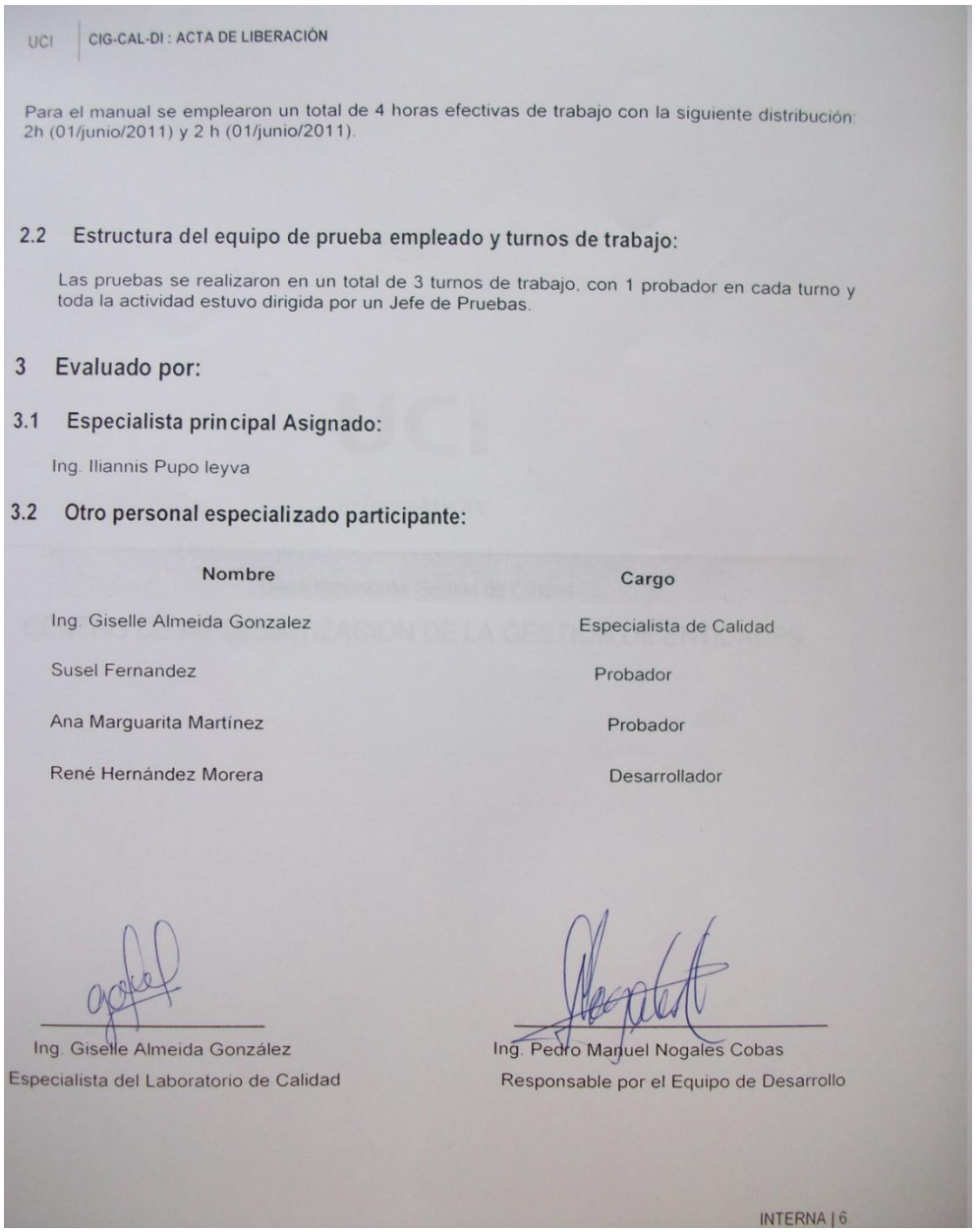
Elemento	Herramienta
Aplicación	Estándar de interfaz
	Lista de Chequeo de Diseño de Interfaz
Manual de usuario	Estándar para el Manual de usuario
	Estándar para la documentación
	Lista de Chequeo del Manual de usuario

2.1 Cantidad total de horas empleadas y rango de fechas:

Para la aplicación se emplearon un total de 7 horas efectivas de trabajo con la siguiente distribución: 2h (25/mayo/2011), 2h (30/mayo/2011), y 4h (01/junio/2011).

INTERNA | 5

Figura 33 Acta de Liberación del producto, página 2 de 3.



UCI | CIG-CAL-DI : ACTA DE LIBERACIÓN

Para el manual se emplearon un total de 4 horas efectivas de trabajo con la siguiente distribución: 2h (01/junio/2011) y 2 h (01/junio/2011).

2.2 Estructura del equipo de prueba empleado y turnos de trabajo:

Las pruebas se realizaron en un total de 3 turnos de trabajo, con 1 probador en cada turno y toda la actividad estuvo dirigida por un Jefe de Pruebas.

3 Evaluado por:

3.1 Especialista principal Asignado:

Ing. Iliannis Pupo leyva

3.2 Otro personal especializado participante:

Nombre	Cargo
Ing. Giselle Almeida Gonzalez	Especialista de Calidad
Susel Fernandez	Probador
Ana Marguarita Martínez	Probador
René Hernández Morera	Desarrollador

Ing. Giselle Almeida González
Especialista del Laboratorio de Calidad

Ing. Pedro Manuel Nogales Cobas
Responsable por el Equipo de Desarrollo

INTERNA | 6

Figura 34 Acta de Liberación del producto, página 3 de 3.