**Universidad de las Ciencias Informáticas.**


# “Diseño e implementación de una herramienta para la gestión de servicios web sobre aplicaciones PHP”

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

**Autores: Javier Ruiz Durán**

**Eyonys González Marcaida**

**Tutores: Ing. René Lazo Ochoa**

**Lic. Manuel Vázquez Acosta**

**Ing. César Lage Codorniu**

Ciudad de la Habana, Junio de 2010

## *De Javier:*

*A mi familia y amigos, por estar junto a mí en todo momento. Sin su apoyo el camino se hubiese hecho más largo.*

*A mi novia, por brindarme su apoyo incondicional y llenar mi vida de felicidad a lo largo de estos cinco años.*

*A los profesores que contribuyeron a mi formación como profesional.*

*A mis compañeros de equipo, por toda la ayuda que nos brindaron.*

*A los tutores de este trabajo sin los cuales no hubiese podido llevarse a cabo.*

*A todas las personas que de una forma u otra han colaborado en la realización de esta investigación.*

## *De Eyonys:*

*En primer lugar agradecer a mi mamá por todos estos años de sacrificios y desvelos, amor y comprensión, por estar ahí siempre que he necesitado su compañía, pues sin ella hoy no sería lo que soy. Te quiero mucho mi gorda y un besote bien grandote.*

*A mis abuelos por sus sabios consejos, de veras los quiero mucho y sepan que estoy eternamente agradecido por cuanto han significado para mí.*

*Un abrazo grande a mi papá, gracias por preocuparte por mí, por ser ese gran amigo que en tantos momentos he necesitado y por lo que me has ayudado.*

*Al resto de mi familia, a mis tíos y primos por poder contar con ellos.*

*A mi hermano Yankiel, mi compañero de travesuras desde pequeños y ese gran amigo con el que he tenido el gusto de compartir gran parte de mi vida, te quiero mi herma y gracias por todo.*

*A mi compañero de tesis El Javie, amigo ejemplar y persona magnífica, un gusto tremendo para mí este año que hemos compartido juntos y gracias por toda la ayuda que he recibido de ti.*

*Al piquete del brood, Tony, Alain y Adrian, desde 1er año juntos compartiendo momentos inolvidables. A mis amigos del grupo y del apto, al Gilbe, a Mary, Dianabel, Daniel, Osniel, Gustavo. A mi gente de los camilitos que venimos juntos desde allá, a Edel, Uriser "el Kangry". Gracias a todos y los quiero mucho.*

*A mis tutores Rene, Cesar y Manuel por toda la ayuda que recibimos de ustedes.*

*A mi oponente Oiner quien lo considero un tutor más por estar todo el tiempo pendiente de qué necesitábamos y cómo solucionar los problemas.*

*Al tribunal por sus consejos y sus observaciones, ayudando a superarnos profesionalmente.*

*A todos mis amigos, a los que he mencionado y a los que no, gracias por ser parte de mi vida y sepan que esté donde esté siempre los recordaré. Un beso grande los quiero.*

*A mis familiares y amigos.*

***Javier***

*Dedico este trabajo a mi hermanita para que lo tome como ejemplo y vea en él una motivación a seguir esforzándose en los estudios y pueda llegar a ser una persona bien preparada.*

***Eyonys***

# RESUMEN

Dentro de los nuevos paradigmas de desarrollo de software la integración juega un papel clave para lograr el éxito de la solución, y no constituye una tarea fácil. Dicha integración se debe lograr tanto entre los componentes de la propia aplicación como con otros sistemas ya desarrollados, sin importar las directrices tecnológicas de los mismos.

En el Centro de Informatización de la Gestión de Entidades, de la Universidad de las Ciencias Informáticas, se ha tomado como base tecnológica a PHP, una tecnología que brinda numerosas ventajas pero los marcos de trabajo existentes dentro de esta, son inmaduros en términos de integración.

Una de las estrategias más utilizadas para llevar a cabo la integración de aplicaciones son los servicios web, tecnología que permite intercambiar mensajes entre estas pero carece de herramientas que permitan el modelado y exportación de los mismos en PHP, lenguaje de programación muy utilizado en el mundo.

Este trabajo tiene como objetivo principal el desarrollo de una herramienta para la gestión de servicios web sobre aplicaciones PHP, la cual permitirá agilizar los procesos de integración con otros sistemas informáticos, sin importar el lenguaje de programación o plataforma sobre la que son ejecutados los mismos.

**Palabras claves:** integración de aplicaciones, servicios web.

# ÍNDICE

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# INTRODUCCIÓN

En la actualidad el entorno cambiante en el que se desenvuelven las empresas hace que las mismas necesiten de nuevas soluciones de software con una prontitud cada vez mayor. Esto ha traído como consecuencia que la industria de software haya evolucionado a nuevos paradigmas de desarrollo que apuestan por la escalabilidad, con componentes que se combinan para dar solución a grandes problemáticas, dejando a un lado las macro-soluciones que ofrecen mayor resistencia a los constantes cambios mencionados.

Dentro de esta nueva filosofía la integración desempeña un papel clave para lograr el éxito de la solución. Dicha integración se debe lograr tanto entre los componentes de la propia aplicación, como con otras aplicaciones ya desarrolladas, lo cual complica aún más la tarea, pues no siempre estas soluciones comparten directrices tecnológicas.

Como solución a esta problemática surgen las plataformas de integración que se encargan de gestionar la integración de las aplicaciones a nivel de servicios, abstrayendo al usuario de la complejidad y la heterogeneidad de las arquitecturas, protocolos, sistemas operativos y lenguajes de programación.

Por otra parte, el Centro de Informatización de la Gestión de Entidades (CEIGE) de la Universidad de las Ciencias Informáticas (UCI) ha tomado como base tecnológica a PHP, una tecnología de probada productividad y eficiencia que además tiene la ventaja de ser software libre, lo cual lo hace altamente atractivo a los ojos de los productores de software en Cuba, dada la situación político-económica del país. Pero cuando se realiza una comparación de los marcos de trabajo existentes dentro de esta tecnología con otros propios de .NET[1] o J2EE[2], se puede detectar la inmadurez de los primeros en términos de integración.

Una de las soluciones más empleadas ante la problemática de la integración son los servicios web, que no son más que una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar mensajes entre aplicaciones, independientemente del lenguaje de programación en el que fueron construidas o la plataforma sobre la que se ejecutan.

[1] .NET es una plataforma de programación que involucra un conjunto de tecnologías desarrolladas por Microsoft.
[2] Java 2 Enterprise Edition es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java.

PHP, a diferencia de las otras tecnologías antes mencionadas, no cuenta con herramientas que permitan el trabajo con servicios web de forma fácil, obligando a sus desarrolladores a realizarlo de forma artesanal, lo cual conduce al siguiente **problema**: Existe un déficit de herramientas que permitan la modelación y exportación de servicios web de aplicaciones PHP lo que dificulta el proceso de integración con otras aplicaciones, independientemente de sus directrices tecnológicas.

Los autores consideran que con la construcción de un sistema informático que permita la modelación y exportación de servicios web de aplicaciones PHP, podría facilitar el proceso de integración de estas aplicaciones con otras, elemento este que se enarbola en el presente trabajo a razón de ***idea a defender***. Para la concreción de la misma se traza como ***objetivo general*** diseñar e implementar una herramienta para la gestión de servicios web sobre aplicaciones PHP lo que para lograrse se desglosa en los siguientes ***objetivos específicos***:

- Construir el marco teórico.
- Diseñar la herramienta
- Implementar la herramienta

Para la solución del problema planteado se define como **objeto de estudio** las plataformas de integración entre aplicaciones y como **campo de acción** las herramientas para la gestión de servicios web, enfocando estos como estrategia de integración.

El presente trabajo consta de tres capítulos. En el Capítulo 1 se realiza la fundamentación teórica de la investigación, la cual incluye un estudio de las diferentes estrategias utilizadas en el mundo para la integración de aplicaciones, así como un análisis de las tecnologías que serán empleadas en la construcción de la solución. El Capítulo 2 constituye la propuesta de solución, en el mismo se exponen los procesos de negocios y requisitos a cumplir por la herramienta además de los artefactos generados durante el diseño de la misma. Mientras que en el Capítulo 3 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

El presente capítulo constituye el marco teórico de la investigación a realizar. En él se definen una serie de conceptos necesarios para entender el objetivo fundamental del trabajo, profundizando en temas como la integración de aplicaciones y los servicios web, haciendo énfasis en las principales plataformas de integración que existen en el mundo. Además se describen las tecnologías que serán utilizadas durante la implementación.

## 1.1 Integración de aplicaciones

La integración de aplicaciones resulta, cada vez más, un factor estratégico a tener en cuenta por las empresas. Pero el proceso de integrar dos o más sistemas no resulta sencillo, dada la gran cantidad de factores involucrados y los riesgos financieros debido a la inversión necesaria en tiempo y recursos (1).

Los equipos de desarrollo involucrados en proyectos de integración se deben enfrentar a diversas problemáticas que surgen con frecuencia, tales como:

- Sistemas aislados, heterogéneos y solapados en funcionalidad.
- Necesidad de soportar procesos que involucran a varios sistemas.
- Información inconsistente en distintos sistemas.
- Costo de implementación de los mecanismos de integración.

De igual manera luego de lograda la integración pueden aparecer otros problemas con los cuales se debe lidiar, tales como:

- Dependencias entre sistemas.
- Pérdida de información.
- Integraciones aisladas sin usar patrones unificados.

### 1.1.1 Niveles de integración

La complejidad de los procesos que cubren las aplicaciones de software de la actualidad provoca que las funcionalidades no siempre puedan ser cubiertas por un solo componente o incluso que excedan los límites de la propia aplicación. Este problema es resuelto con la integración a distintos niveles utilizando diferentes estrategias (2).

En el desarrollo basado en componentes se pueden identificar tres niveles de integración:

- Componente a componente: es el nivel más simple de integración, se presenta cuando se combinan componentes para dar soporte a las funcionalidades de un subsistema.
- Subsistema a subsistema: se presenta cuando los procesos abarcan funcionalidades que son responsabilidad de distintos subsistemas. Es un tanto más complejo que el nivel anterior, pues los subsistemas involucrados pueden estar distribuidos en diferentes entornos y se deben evitar las dependencias fuertes.
- Nivel de servicios: se presenta cuando las aplicaciones no son capaces de cubrir por sí solas todos los procesos que necesitan las entidades y deben integrarse con otras aplicaciones que las soporten. Es el nivel más complejo, pues las aplicaciones a integrar no siempre comparten directrices tecnológicas.

### 1.1.2 Técnicas de integración

Como respuesta a la necesidad de integrar las diversas aplicaciones en la industria del software, se han creado estrategias o técnicas que potencian dicha integración en los diferentes niveles. A continuación se realiza un estudio de algunas de las técnicas más utilizadas por los sistemas de la actualidad para lograr la integración a nivel de servicios.

#### 1.1.2.1 Protocolos de comunicación entre aplicaciones

Una de las técnicas más empleadas por los sistemas de la actualidad es el uso de estándares o protocolos, los cuales son utilizados como base para lograr una comunicación rápida y segura entre las diversas aplicaciones. A continuación se describen algunos de los más utilizados.

**Arquitectura Común de Intermediarios en Peticiones a Objetos (CORBA)**

CORBA, es un modelo de soporte para la programación distribuida orientada a objetos. Hace posible que los objetos interactúen a través de lenguajes de programación, protocolos de comunicación y plataformas heterogéneas (2).

Es una tecnología que oculta la programación a bajo nivel de aplicaciones distribuidas, de tal forma que el programador no se tiene que ocupar de tratar con sockets, flujos de datos, paquetes, sesiones, etc. Brinda al programador una tecnología orientada a objetos, las funciones y los datos se agrupan en objetos, los

cuales pueden estar en diferentes máquinas, pero el programador accederá a ellos a través de funciones normales dentro de su programa (2).

Los métodos y datos CORBA se agrupan formando lo que se denominan interfaces, las cuales son definidas utilizando un lenguaje de descripción de interfaces IDL. Este lenguaje es estándar y lo soportan todas las implementaciones CORBA (2).

**Modelo de Objetos Componentes (COM)**

COM, es una plataforma de Microsoft para comunicar diferentes aplicaciones. Es utilizada por los desarrolladores para crear componentes de software reutilizables. Los objetos pueden ser creados en distintos lenguajes (2).

COM define un binario estándar, en el cual se compilarán los módulos binarios, tales como las DLLs y los ejecutables, en una estructura específica. Este estándar especifica exactamente cómo los objetos COM deben ser organizados en memoria. Dichos binarios no deben depender de ninguna característica específica del lenguaje en el que está implementado. Una vez hecho esto, los módulos pueden ser accedidos por diferentes aplicaciones implementadas en otros lenguajes de programación. El estándar pone toda la carga de compatibilidad en el compilador que produce los binarios, lo que facilita el uso de estos por parte de las personas que requieran de sus servicios (2).

**Protocolo de Acceso Simple a Objetos (SOAP)**

SOAP (siglas en inglés de “protocolo de acceso simple a objetos”) es un protocolo que proporciona un mecanismo estándar para empaquetar mensajes. Este protocolo ha sido diseñado con el fin del intercambio de información en entornos descentralizados y distribuidos, con independencia de cualquier modelo de programación. Usa las tecnologías relacionadas con XML a fin de definir un marco de trabajo extensible para los mensajes (3).

Algunas de las ventajas que ofrece SOAP son las siguientes:

- No está asociado a ningún lenguaje. SOAP no especifica una API, por lo que la implementación de esta se deja al lenguaje de programación que se elija.
- No se encuentra fuertemente asociado a ningún protocolo de transporte. Un mensaje de SOAP no es sino un documento en XML, por lo que la única característica que se le exigirá al protocolo de transporte es que tenga la capacidad de transmitir cadenas de texto.

- No está ligado a ninguna infraestructura de objetos distribuidos. SOAP brinda cierto grado de interoperabilidad entre sistemas diferentes, en los que se ejecutan componentes middleware de distintos fabricantes.
- Aprovecha los estándares existentes en la industria. Al crear SOAP, no se reinventaron conceptos, sino que se tomaron estándares existentes y se extendieron (4).

#### 1.1.2.2 Servicios web

Existen múltiples definiciones sobre lo que son los Servicios Web (WS por sus siglas en inglés), lo que dificulta la tarea de dar una adecuada definición que englobe todo lo que son e implican. El término Servicio Web, al tratarse de una tecnología relativamente reciente, es un concepto del cual se abusa con cierta frecuencia, ya que no se tiene una idea clara y concisa acerca de los requisitos que debe cumplir un sistema software para que sea realmente un WS. Algunas de las definiciones que dan las publicaciones sobre el tema son las siguientes:

- "Un servicio web es una aplicación software identificada por una URI, cuyas interfaces y vinculaciones son capaces de ser definidas, descritas y descubiertas como artefactos XML. Un WS soporta la interacción con otros agentes software mediante el intercambio de mensajes basado en XML a través de protocolos basados en Internet." (5)
- "Los WS son interfaces Web genéricas a servicios componente. A fin de soportar la interoperabilidad entre todas las arquitecturas, los WS utilizan protocolos del World Wide Web Consortium (W3C) como pueden ser XML, WSDL y SOAP." (6)
- "Los WS son aplicaciones basadas en la Web compuestas por funcionalidades de granularidad gruesa que son accesibles a través de Internet. Desde una perspectiva técnica, los WS son una forma estandarizada de integrar aplicaciones basadas en la Web mediante estándares abiertos, incluyendo XML, SOAP, WSDL, y UDDI." (7)

De modo general se puede definir que los servicios web son una tecnología que utiliza un conjunto de estándares y protocolos para el intercambio de mensajes entre aplicaciones de software, con independencia de las directrices tecnológicas de las mismas.

Los servicios web son muy empleados en la actualidad debido a la facilidad de su uso y las ventajas que conlleva, entre las que se encuentran:

- Aportan interoperabilidad entre aplicaciones de software, independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Al apoyarse en HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.
- Permiten que servicios y software de diferentes compañías, ubicadas en diferentes lugares geográficos, puedan ser combinados fácilmente para proveer servicios integrados.
- Las especificaciones son gestionadas por una organización abierta, la W3C, por tanto no hay secretismos por intereses particulares de fabricantes concretos y se garantiza la plena interoperabilidad entre aplicaciones.

Las organizaciones OASIS y W3C son las responsables de la arquitectura y reglamentación de los servicios Web. Para mejorar la interoperabilidad entre distintas implementaciones de servicios Web se ha creado el organismo WS-I, encargado de definir de manera más exhaustiva estos estándares (3).

**Estándares utilizados en los servicios web**

Los servicios web tienen interfaces tipadas, soporte multilenguaje, usan los estándares de Internet existentes, cumplen las propiedades de interoperabilidad, entre otras propiedades que son alcanzables gracias a la base en que están sustentados.

Bajo el concepto de servicio web, subyacen una serie de estándares que ofrecen seguridad y confiabilidad a todos los aspectos que rodean a los mismos (3).

Estos estándares definen cómo se describen los servicios, cómo se localizan, cómo se representa la información y la forma mediante la cual se deben comunicar. A continuación serán tratados algunos de los estándares de mayor importancia.

- **Lenguaje de marcas extensibles (XML)**

XML (siglas en ingles de “lenguaje de marcas extensibles”) es un metalenguaje extensible de etiquetas, desarrollado por el W3C, que no se considera como un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades (3).

Su aplicación no está solamente enmarcada en Internet a pesar de ser este el medio donde más se utiliza. También puede ser utilizado en bases de datos, editores de texto, hojas de cálculo, entre otras aplicaciones. Sus creadores lo proponen como un estándar para el intercambio de información estructurada entre diferentes plataformas (3). A continuación se listan algunas de las ventajas que brinda el uso de XML:

- Es extensible, de manera que es posible la adición de nuevas etiquetas para ser adaptado a las necesidades sin problemas mayores.
- El parser (analizador) es un componente estándar, de modo que no es necesario crear un analizador específico para cada versión de lenguaje XML. Esto posibilita el empleo de cualquiera de los analizadores disponibles. De esta manera se evitan errores y se acelera el desarrollo de aplicaciones.

➢ **Lenguaje de descripción de servicios web (WSDL)**

WSDL (siglas en inglés de "lenguaje de descripción de servicios web") es una aplicación de XML que se utiliza para describir servicios web. WSDL define un conjunto de elementos mediante los cuales se puede describir los servicios de la red como colecciones de nodos de comunicación que tienen la capacidad de intercambiar mensajes. La versión 2.0 es la recomendación actual por parte del W3C (3).

Un documento WSDL utiliza los siguientes elementos en la definición de los servicios de red:

- *Tipos*: definición de los tipos de datos utilizados en los mensajes.
- *Mensaje*: definición abstracta de los datos que están siendo transmitidos.
- *Operación*: descripción abstracta de una operación soportada por el servicio.
- *Tipo de Puerto*: conjunto abstracto de operaciones soportadas por uno o más nodos.
- *Vinculación*: protocolo concreto y especificación del formato de los datos de un *tipo de puerto* en particular.
- *Puerto*: nodo definido como la combinación de una vinculación y una dirección de red.
- *Servicio*: colección de nodos relacionados. (3)

**Herramientas para la gestión de servicios web**

- **NetBeans IDE**

NetBeans es una herramienta de desarrollo libre y de código abierto creada por la compañía Sun MicroSystems. La misma es capaz de generar el código necesario para la creación tanto de WS como de Clientes WS para Java. Todo esto es posible hacerlo mediante Asistentes que permiten la configuración visual de los WS, de forma tal que le ahorre tiempo al desarrollador (8).

- **Microsoft Visual Studio**

Es una potente herramienta para el desarrollo de aplicaciones basadas en la tecnología .NET. Es un software privativo creado por la compañía Microsoft, el cual permite la generación de todo el código necesario para la creación de WS escritos en Asp.net. Permite además crear clientes que consuman dichos servicios en lenguajes como C#, Visual Basic o C++ (9).

- **Zend Studio**

Creado por la compañía Zend, es una herramienta propietaria que permite el desarrollo de aplicaciones Web mediante PHP. Permite generar el WSDL correspondiente a una clase o función escrita en PHP, basándose en elementos de configuración previamente definidos por el desarrollador, como la localización del WSDL, convención de nombres y opciones de enlaces, entre otros (10).

#### 1.1.2.3 Plataformas de integración

Tradicionalmente la integración de sistemas se realizaba punto a punto (P2P); sin embargo, este enfoque requiere de grandes esfuerzos que se ven incrementados por el número de aplicaciones que se desean integrar (11).

Como solución a esta problemática, surgen las tecnologías llamadas Middleware, que se encargan de gestionar la integración de las aplicaciones a nivel de servicios.

Se define Middleware, como la capa de software que se coloca entre el usuario y el entorno distribuido, abstrayendo al usuario de la complejidad y la heterogeneidad de las arquitecturas, protocolos, sistemas operativos, lenguajes de programación; es decir, otorga la posibilidad de intercomunicar aplicaciones desarrolladas en distintos lenguajes de programación, sistemas operativos y plataformas (12).

A continuación son descritas algunas de estas tecnologías:

**Thrift**

Thrift es una librería y un conjunto de herramientas de generación de código fuente para el desarrollo rápido y seguro de servicios escalables y eficientes entre aplicaciones. Su principal objetivo es posibilitar la comunicación eficiente y confiable entre lenguajes de programación, abstrayéndose de la porción más difícil de configurar de cada lenguaje con una librería implementada en cada uno de los mismos. Permite a los desarrolladores definir tipos de datos e interfaces de servicios en un fichero de lenguaje neutral y generar el código necesario para construir clientes y servidores RPC (13). Aunque fue desarrollado por Facebook, actualmente es un proyecto open-source de la Apache Software Foundation Incubator.

A continuación se muestran algunas de las características de Thrift:

- Lenguajes de programación soportados: C++, Java, Python, Erlang y PHP. Soporte limitado para XSD, Ruby, C#, Perl, Objective C, Smalltalk, OCaml, JavaScript y Haskell.
- Tipos de datos primitivos: bool, byte, i16 (entero de 16 bits), i32 (entero de 32 bits), i64 (entero de 64 bits), double, string, map<t1,t2>, list<t>, set<t>.
- Formatos de serialización: binario, json.
- Otras características: enumerativos, constantes, excepciones tipadas, interfaces RPC.
- Licencia: Apache License (13).

**Protocol Buffers**

Protocol Buffers es un formato de serialización con un lenguaje de descripción de interfaces desarrollado por Google. Su diseño fue enfocado con el objetivo de lograr simplicidad y eficiencia, de hecho fue diseñado para que fuera más eficiente que XML, aunque no existen comparaciones publicadas que lo confirmen. Es muy similar a Thrift y es ampliamente utilizado en Google para el intercambio y almacenamiento de todo tipo de información estructurada (14).

A continuación se muestran algunas de las características de Protocol Buffers:

- Lenguajes de programación soportados: C++, Java y Python, por el momento.
- Tipos de datos primitivos: bool, interger (32 y 64 bits), float, double, string, byte sequence.
- Formatos de serialización: binario.
- Otras características: enumerativos, interfaces RPC.
- Licencia: BSD-style (14).

**ZendExt_Service**

Las plataformas estudiadas con anterioridad, han sido construidas por corporaciones de renombre internacional y constituyen tecnologías de punta, pero en un estudio realizado en el CEIGE se determinó que resultan inviables para el desarrollo de las líneas de productos de la Dirección Tecnológica del centro, debido al elevado coste que implicaría su utilización en términos de capacitación del personal y mantenimiento de los productos construidos (15).

Por lo antes expuesto, el CEIGE determinó que la solución más viable a las necesidades de integrar sus productos era la construcción de una plataforma que soportara la integración de aplicaciones a alto nivel para proyectos desarrollados en PHP haciendo uso de las herramientas y tecnologías definidas por el Departamento de Tecnología del centro. La principal función de esta plataforma es la publicación de interfaces interoperables para la exportación de servicios web, basándose en el estándar SCA (Service Component Architecture). Dicha plataforma fue desarrollada como un componente más del marco de trabajo del centro, lo cual permite que se complemente con otros componentes del mismo, dándole fortaleza a la integración pues inserta elementos como la seguridad y el control de trazas (15).

## 1.2 Tecnologías empleadas

El presente trabajo forma parte de un proceso productivo iniciado en el CEIGE en el cual se tomaron decisiones tecnológicas que involucran la utilización de tecnologías Open-Source para el desarrollo de sus productos. A continuación se describen brevemente estas tecnologías.

### 1.2.1 Lenguajes de programación

**PHP**

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 del lado del servidor. 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. Se integra perfectamente a la mayoría de los Sistemas Gestores de Bases de Datos. 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 (16).

**JavaScript**

JavaScript es un lenguaje de scripting basado en objetos, que se utiliza principalmente para crear páginas web dinámicas y permite el desarrollo de interfaces de usuario mejoradas. Una página web dinámica es aquella que permite la interacción entre el contenido de la misma y el usuario. JavaScript permite incorporar a dichas páginas 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. A pesar de su nombre, este no guarda relación directa con el lenguaje Java, sino que simplemente la compañía dueña del mismo lo adoptó por una cuestión de mercado (17).

### 1.2.2 Librerías y marcos de trabajo

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 (18).

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

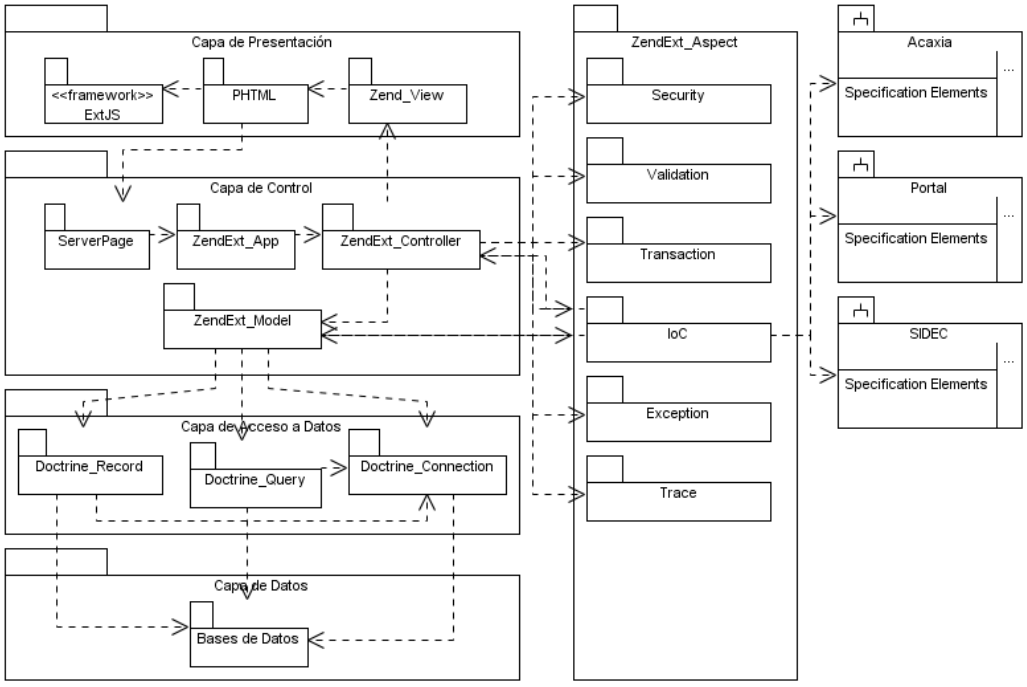



**Figura 1 Estructura del marco de trabajo Sauxe.**

### Zend Framework 1.8

Zend Framework no es más que un framework MVC (Modelo-Vista-Controlador) open-source para el desarrollo de aplicaciones Web con PHP. Brinda soluciones para construir sitios web modernos, robustos y seguros. Posee un bajo acoplamiento entre sus componentes, lo que posibilita la utilización de los mismos a conveniencia, aunque todos estos en conjunto conforman un potente y extensible framework para aplicaciones web. Brinda una alta abstracción de bases de datos, haciendo extremadamente simple la interacción con estas, sin necesidad de escribir ninguna consulta SQL. Cuenta con módulos para el manejo de ficheros PDF, canales RSS, entre otros. Cuenta con clientes para el acceso a WS y robustas clases para la autenticación y el filtrado de entrada; completa documentación y tests de alta calidad (19).

### Doctrine 0.11

Doctrine es un potente y completo sistema ORM (siglas en inglés de “mapeador relacional de objetos”) para PHP con un DBAL (siglas en inglés de “capa de abstracción de bases de datos”) incorporado que permite exportar una base de datos a sus clases correspondientes y viceversa, o sea, a partir de las

clases creadas y siguiendo las especificaciones de ORM, generar las tablas de la base de datos. Se encuentra en la parte superior de una poderosa Capa de Abstracción de Base de Datos (DBAL). Una de sus principales características es la opción de escribir las consultas de base de datos en un objeto con una propiedad orientada al dialecto SQL llamada Doctrine Query Language (DQL), inspirada en Hibernate HQL. Esto proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad, sin necesidad de la duplicación de código innecesaria (20).

**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 (17).

### 1.2.3 Herramientas de desarrollo

**Apache 2.2.9**

Apache es el servidor web por excelencia, su facilidad de configuración, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. Se ejecuta en gran cantidad de sistemas operativos, lo que lo hace prácticamente universal. Es una tecnología gratuita, open-source y altamente configurable de diseño modular por lo que resulta muy sencillo ampliar las sus capacidades. Actualmente existen muchos módulos para Apache que son adaptables a este, y están disponibles para su instalación cuando sean necesarios. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor y es posible configurarlo para que ejecute un determinado script cuando esto suceda (21).

**PostgreSQL 8.3**

Es un sistema de gestión de bases de datos libre basado en el proyecto Postgres, perteneciente a la Universidad de Berkeley. Es un sistema objeto-relacional, que incluye características como la herencia, valores no atómicos (atributos basados en vectores y conjuntos), funciones, disparadores, entre otras. Es altamente extensible, permitiendo el uso de operadores, funciones y tipos de datos definidos por el usuario. Soporta la integridad referencial garantizando la integridad de los datos en la base de datos. PostgreSQL permite realizar múltiples conexiones desde procesos clientes, existiendo un proceso maestro en el servidor que siempre se ejecuta y que está a la espera de nuevas conexiones clientes, de forma tal que cuando alguien se conecta, se inicia un nuevo proceso, asegurando que el cliente y la nueva conexión no necesitan del proceso postgres original. Una de sus principales características es la alta concurrencia. Esto permite que mientras se realizan cambios en una tabla, otros procesos accedan a la misma sin la necesidad de bloqueos, además de que cada usuario tiene visión de la última modificación. Presenta el inconveniente de que para bases de datos pequeñas su velocidad de respuesta no es muy eficiente en comparación con otras relativamente grandes (20).

**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 (8).

**Visual Paradigm 4.3**

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientado a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. 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. Soporta UML versión 2.1, permite modelado colaborativo con CVS y Subversion, generación de código, ingeniería inversa, generación de bases de datos (transformación de diagramas entidad-relación en tablas de la base de

datos), importación y exportación a ficheros XML, distribución automática de diagramas, entre otras características (22).

## 1.3 Conclusiones parciales

Una vez realizado el estudio del marco teórico de la presente investigación se arribó a las siguientes conclusiones:

- Existen tecnologías para la integración de aplicaciones de eficiencia probada como CORBA, Thrift y Protocol Buffers pero al no estar basadas en WS no se ajustan a las necesidades del CEIGE.
- Las herramientas disponibles que permiten la gestión de WS no resuelven las necesidades expuestas en la situación problémica.
- La plataforma ZendExt_Service está basada en WS con lo que resuelve las necesidades de integración de las aplicaciones desarrolladas en el CEIGE pero carece de una herramienta para la gestión de las interfaces interoperables que esta provee.

Por lo antes expuesto se propone la construcción de una herramienta que potencie la integración de aplicaciones en PHP, mediante la gestión visual de la plataforma de integración desarrollada en el CEIGE.

# CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

El siguiente capítulo recoge los resultados obtenidos durante el proceso de desarrollo de la solución, así como algunos de los artefactos generados por el mismo. Primeramente son descritos los procesos de negocio relativos al campo de acción y los requisitos funcionales de la solución para lograr un entendimiento de lo que se quiere lograr. Posteriormente se realiza una descripción del diseño elaborado por los autores para alcanzar las metas trazadas, además de la estrategia a seguir durante el proceso de implementación.

## 2.1 Descripción de los procesos de negocio

Un proceso de negocio es un conjunto de tareas relacionadas lógicamente llevadas a cabo para lograr un resultado de negocio definido. La descripción de los procesos de negocio hace más viable el paso a las actividades del análisis ya 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.

Los autores, teniendo en cuenta la opinión de especialistas y de los usuarios finales, han identificado tres procesos de negocio que deben ser asistidos por la solución. Estos procesos son descritos a continuación.

### 2.1.1 Descripción del proceso: Crear paquete de servicios

La solución debe permitir a los usuarios la creación de paquetes de servicios. Estos no son más que un grupo de funcionalidades agrupadas según el criterio del usuario, las cuales son expuestas como servicios web. Este es el proceso principal de la solución y para llevarlo a cabo el usuario debe especificar las funcionalidades que desea incluir en el paquete, así como otros datos relativos al paquete, como nombre y descripción.

Es de gran importancia que los métodos seleccionados por el usuario tengan definidos en el comentario los tipos de datos involucrados, tanto de los parámetros recibidos como del valor de retorno, debido a que es un requisito de los servicios web que los datos involucrados sean tipados y esta es una característica ajena a PHP, base tecnológica de las aplicaciones con las que se trabajará.

Como resultado de este proceso se debe obtener una clase o paquete con un grupo de métodos o servicios, los cuales constituyen interfaces a las funcionalidades escogidas por el usuario, además de un WSDL con la descripción de dicha clase. En la siguiente figura se muestra el diagrama del proceso.

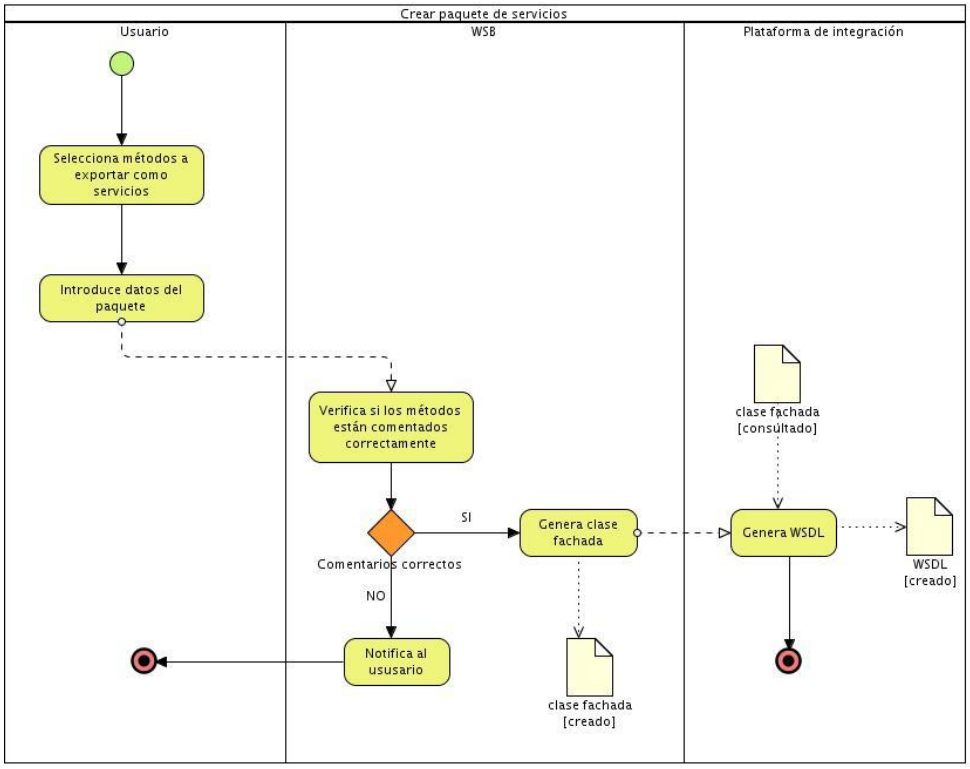



**Figura 2 Proceso Crear Paquete de Servicios.**

### 2.1.2 Descripción del proceso: Probar servicio

La solución debe permitir a los usuarios poner a prueba los servicios previamente creados, para garantizar el correcto funcionamiento de los mismos. Para llevar a cabo esto se hace necesario que el usuario especifique el juego de datos con el cual se desea probar el servicio. Una vez terminado el proceso, el usuario podrá comprobar si el servicio en cuestión presenta algún tipo de error. En la siguiente figura se muestra el diagrama del proceso.

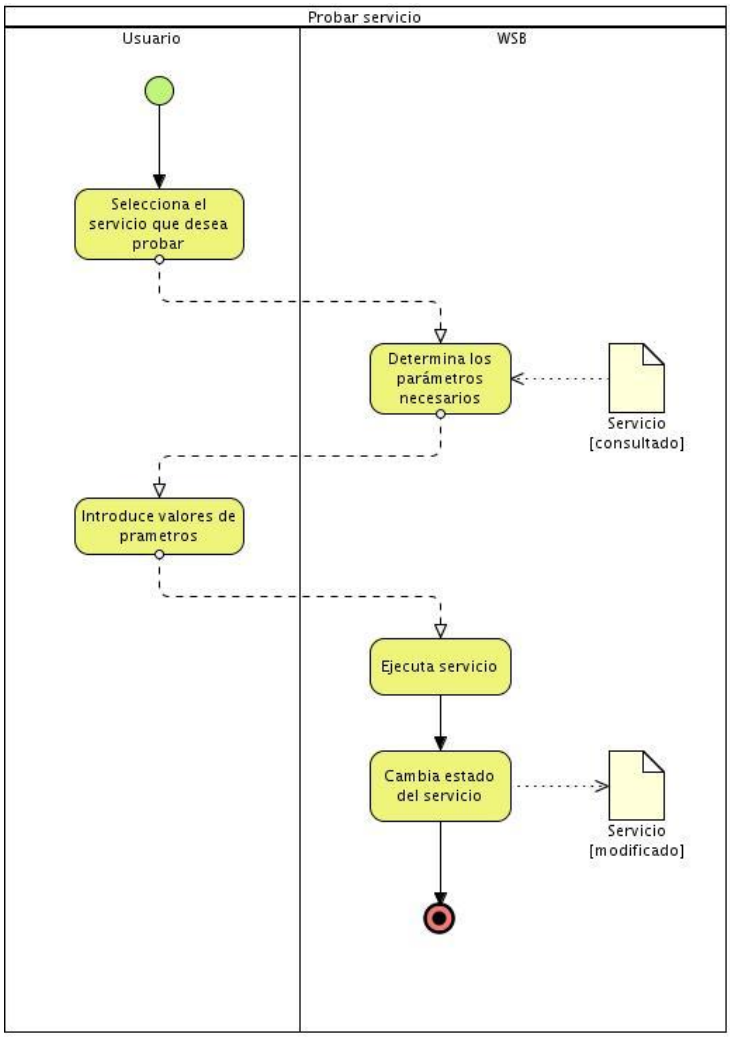



**Figura 3 Proceso Probar Servicio.**

### 2.1.3 Descripción del proceso: Crear proxy

Con el objetivo de hacer el proceso de integración aun más transparente a los ojos de los desarrolladores, la solución debe permitir la creación de proxies, que no son más que una clase que encapsula toda la lógica de manejo de la interfaz interoperable. Para llevar a cabo el proceso, el usuario deberá especificar qué servicios desea que sean consumidos, así como datos relativos al proxy en sí, como nombre y

descripción. Una vez terminado el proceso el usuario podrá descargar el proxy creado para su posterior utilización en cualquier aplicación PHP. En la siguiente figura se muestra el diagrama del proceso.

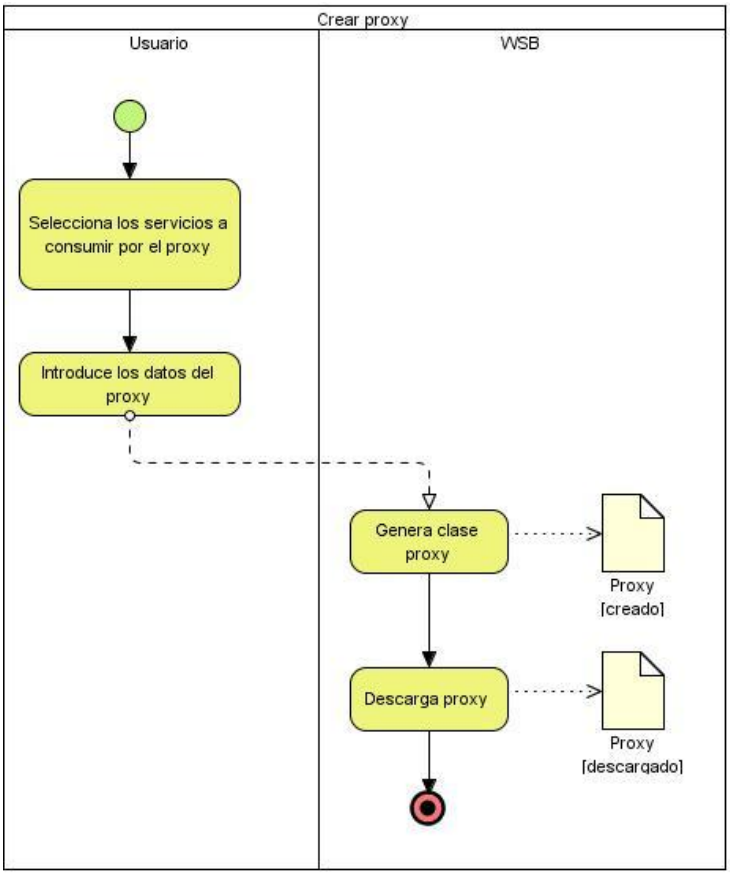



**Figura 4 Proceso Crear Proxy.**

## 2.2 Requisitos de software

Uno de los elementos más importantes en un proceso de desarrollo de software lo constituyen los requisitos, pues estos permiten una comunicación efectiva entre los usuarios y el equipo de desarrollo, con el objetivo de llegar a un entendimiento de lo que hay que realizar, siendo así la clave del éxito en la producción de un software.

Existen distintas definiciones de requisito de software dadas por diversos autores entre las que podemos citar:

- Los requisitos son expresiones de las necesidades de *stakeholders* para alcanzar una meta particular (23).
- Un requisito es una condición o capacidad necesaria dada por un usuario con el objetivo de resolver un problema o alcanzar un objetivo (24).
- Los requisitos expresan las necesidades y restricciones atribuibles a un producto de software que contribuye a la solución de algún problema del mundo real (25).

Los requisitos de software se clasifican en funcionales y no funcionales. Los requisitos funcionales describen qué es lo que el sistema debe hacer para dar soporte a las funciones y objetivos del usuario. Los requisitos no funcionales imponen restricciones de cómo los requisitos funcionales deben ser implementados (26).

### 2.2.1 Requisitos funcionales

Los requisitos funcionales son los que definen las funciones que el sistema será capaz de realizar. Los mismos se centran en *qué* y no *cómo* se deben hacer esas funciones.

Una vez descritos los procesos de negocio, apoyándose en la experiencia de los analistas del CEIGE y con participación de los usuarios, se identificaron los requisitos a cumplir por el sistema, los cuales son listados a continuación:

R 1. Gestionar servicios.
- R 1.1. Crear paquete de servicios.
- R 1.2. Eliminar paquete de servicios.
- R 1.3. Adicionar servicio.
- R 1.4. Eliminar servicio.
- R 1.5. Probar servicio.

R 2. Gestionar proxy.
- R 1.1. Crear proxy.
- R 1.2. Eliminar proxy.
- R 1.3. Descargar proxy.

R 3. Gestionar solución.
- R 1.1. Adicionar solución.

R 1.2. Eliminar solución.

R 1.3. Cargar solución.

R 4. Gestionar comentario.

R 1.1. Adicionar comentario.

R 1.2. Modificar comentario.

#### 2.2.1.1 Requisito funcional: Gestionar servicios

**Tabla 1 Especificación del requisito Crear Paquete de Servicios**

| **Conceptos tratados** | **Conceptos** | **Atributos** |
|---|---|---|
| | Paquete de servicios | Nombre, autor, URI, descripción. |
| **Precondiciones** | **Precondiciones** | **Pre-requisito** |
| | Tener una solución cargada. | Cargar solución. |
| | La solución debe tener métodos declarados en su código fuente. | No procede. |
| **Descripción** | Para crear un paquete de servicios el usuario debe seleccionar los métodos que desea incluir como servicios en el paquete, así como especificar el nombre del paquete, siendo estos datos de carácter obligatorio. Opcionalmente puede introducir el autor y la descripción del paquete, datos que de no ser especificados tomarán 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, crear el paquete de | |

| | |
|---|---|
| | servicios y mostrar un mensaje de notificación.<br>El sistema debe permitir la cancelación de esta acción. |
| **Validaciones** | El nombre del paquete debe tener las características del nombre de una clase en el lenguaje PHP, por lo que solo permite el empleo de letras (mayúsculas y minúsculas), números y guión bajo.<br>No se permite la creación de un paquete con el mismo nombre de otro ya existente.<br>No se permiten valores nulos en los datos de carácter obligatorio. |
| **Post-condiciones** | Se ha adicionado un nuevo paquete de servicios. |
| **Post-requisito** | No procede. |

**Tabla 2 Especificación del requisito Eliminar Paquete de Servicios**

| **Conceptos tratados** | **Conceptos** | **Atributos** |
|---|---|---|
| | Paquete de servicios | Nombre, autor, URI, descripción. |
| **Precondiciones** | **Precondiciones** | **Pre-requisito** |
| | Tener paquetes de servicios creados. | Crear paquete de servicios. |

| | |
|---|---|
| **Descripción** | El usuario selecciona el paquete que desea eliminar, el sistema debe mostrar un mensaje de confirmación. En caso de que el usuario confirme la acción, el sistema elimina el paquete de servicios y muestra un mensaje de notificación. |
| **Validaciones** | No procede. |
| **Post-condiciones** | Se ha eliminado el paquete de servicios. |
| **Post-requisito** | No procede. |

**Tabla 3 Especificación del requisito Adicionar Servicio**

| **Conceptos tratados** | **Conceptos** | **Atributos** |
|---|---|---|
| | Servicio | Nombre, descripción, URI, estado. |
| | Paquete de servicios | Nombre, autor, URI, descripción. |
| **Precondiciones** | **Precondiciones** | **Pre-requisito** |
| | Tener una solución cargada. | Cargar solución. |
| | La solución debe tener métodos declarados en su | No procede. |

| | | |
|---|---|---|
| | código fuente. | |
| **Descripción** | El usuario selecciona el método que desea exportar como servicio web, el sistema solicita al usuario especificar si desea adicionar el servicio a un paquete existente o crear uno nuevo. En caso de que el usuario seleccione la opción de crear un paquete nuevo, se da paso al requisito Crear Paquete de Servicios. En caso contrario, el sistema comprueba que el método esté bien comentado, adiciona el servicio al paquete especificado por el usuario y muestra un mensaje de notificación.<br>El sistema debe permitir la cancelación de esta acción. | |
| **Validaciones** | No procede. | |
| **Post-condiciones** | Se ha adicionado un nuevo servicio. | |
| **Post-requisito** | No procede. | |

**Tabla 4 Especificación del requisito Eliminar Servicio**

| **Conceptos tratados** | **Conceptos** | **Atributos** |
|---|---|---|
| | Servicio | Nombre, descripción, URI, estado. |
| **Precondiciones** | **Precondiciones** | **Pre-requisito** |
| | Tener servicios creados. | Crear paquete de servicios. |

| | | |
|---|---|---|
| | | Adicionar servicio. |
| **Descripción** | El usuario selecciona el servicio que desea eliminar. El sistema solicita la confirmación del usuario para realizar esta acción. En caso de que el usuario confirme la acción el sistema elimina el servicio especificado. | |
| **Validaciones** | No procede. | |
| **Post-condiciones** | Se ha eliminado el servicio. | |
| **Post-requisito** | No procede. | |

**Tabla 5 Especificación del requisito Probar Servicio**

| **Conceptos tratados** | **Conceptos** | **Atributos** |
|---|---|---|
| | Servicio | Nombre, descripción, URI, estado. |
| **Precondiciones** | **Precondiciones** | **Pre-requisito** |
| | Tener servicios creados. | Crear paquete de servicios.<br>Adicionar servicio. |
| **Descripción** | El usuario selecciona el servicio 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.<br>No se permiten valores nulos en los datos de carácter obligatorio. |
| **Post-condiciones** | No procede. |
| **Post-requisito** | No procede. |

#### 2.2.1.2 Requisito funcional: Gestionar proxy

**Tabla 6 Especificación del requisito Crear Proxy**

| **Conceptos tratados** | **Conceptos** | **Atributos** |
|---|---|---|
| | Proxy | Nombre, descripción. |
| | Servicio | Nombre, descripción, URI, estado. |
| **Precondiciones** | **Precondiciones** | **Pre-requisito** |
| | Tener servicios creados. | Crear paquete de servicios.<br>Adicionar servicio. |

| | |
|---|---|
| **Descripción** | Para crear un proxy el usuario debe seleccionar los servicios que desea que sean consumidos desde el mismo, así como especificar el nombre del proxy, siendo estos datos de carácter obligatorio. Opcionalmente puede introducir la descripción del proxy, dato 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 crear el proxy y mostrar un mensaje de notificación.<br>El sistema debe permitir la cancelación de esta acción. |
| **Validaciones** | El nombre del proxy debe tener las características del nombre de una clase en el lenguaje PHP, por lo que solo permite el empleo de letras (mayúsculas y minúsculas), números y guión bajo.<br>No se permiten valores nulos en los datos de carácter obligatorio. |
| **Post-condiciones** | Se ha adicionado un nuevo proxy. |
| **Post-requisito** | No procede. |

**Tabla 7 Especificación del requisito Eliminar Proxy**

| **Conceptos tratados** | **Conceptos** | **Atributos** |
|---|---|---|
| | Proxy | Nombre, descripción. |

| **Precondiciones** | **Precondiciones** | **Pre-requisito** |
|---|---|---|
| | Tener proxies creados. | Crear proxy. |
| **Descripción** | El usuario selecciona el proxy que desea eliminar. El sistema solicita la confirmación del usuario para realizar esta acción. En caso de que el usuario confirme la acción el sistema elimina el proxy especificado. | |
| **Validaciones** | No procede. | |
| **Post-condiciones** | Se ha eliminado el proxy. | |
| **Post-requisito** | No procede. | |

**Tabla 8 Especificación del requisito Descargar Proxy**

| **Conceptos tratados** | **Conceptos** | **Atributos** |
|---|---|---|
| | Proxy | Nombre, descripción. |
| **Precondiciones** | **Precondiciones** | **Pre-requisito** |
| | Tener proxies creados. | Crear proxy |
| **Descripción** | El usuario selecciona el proxy que desea descargar. El sistema envía el proxy desde el servidor hacia el cliente en un archivo comprimido. | |

| | |
|---|---|
| **Validaciones** | No procede. |
| **Post-condiciones** | Se ha descargado el proxy. |
| **Post-requisito** | No procede. |

#### 2.2.1.3 Requisito funcional: Gestionar solución

**Tabla 9 Especificación del requisito Adicionar Solución**

| **Conceptos tratados** | **Conceptos** | **Atributos** |
|---|---|---|
| | Solución | Nombre, ruta de acceso, fichero de configuración. |
| **Precondiciones** | **Precondiciones** | **Pre-requisito** |
| | No procede. | No procede. |
| **Descripción** | Para adicionar una nueva solución el usuario debe especificar el nombre de la misma, así como la ruta física donde se encuentra su código, siendo estos datos de carácter obligatorio.<br>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 adicionar la solución y mostrar un mensaje de notificación. | |

| | |
|---|---|
| | El sistema debe permitir la cancelación de esta acción. |
| **Validaciones** | No se permiten valores nulos en los datos de carácter obligatorio. |
| **Post-condiciones** | Se ha adicionado una nueva solución. |
| **Post-requisito** | No procede. |

**Tabla 10 Especificación del requisito Eliminar Solución**

| **Conceptos tratados** | **Conceptos** | **Atributos** |
|---|---|---|
| | Solución | Nombre, ruta de acceso, fichero de configuración. |
| **Precondiciones** | **Precondiciones** | **Pre-requisito** |
| | Tener soluciones creadas. | Adicionar solución. |
| **Descripción** | El usuario selecciona la solución que desea eliminar. El sistema solicita la confirmación del usuario para realizar esta acción. En caso de que el usuario confirme la acción el sistema elimina la solución especificada. | |
| **Validaciones** | No procede. | |

| | |
|---|---|
| **Post-condiciones** | Se ha eliminado la solución. |
| **Post-requisito** | No procede. |

**Tabla 11 Especificación del requisito Cargar Solución**

| **Conceptos tratados** | **Conceptos** | **Atributos** |
|---|---|---|
| | Solución | Nombre, ruta de acceso, fichero de configuración. |
| **Precondiciones** | **Precondiciones** | **Pre-requisito** |
| | Tener soluciones creadas. | Adicionar solución. |
| **Descripción** | El usuario selecciona la solución que desea cargar. El sistema muestra los datos de la solución especificada (árbol de introspección del código fuente, listado de servicios agrupados por paquetes y listado de proxies). | |
| **Validaciones** | No procede. | |
| **Post-condiciones** | Se ha cargado la solución. | |
| **Post-requisito** | No procede. | |

#### 2.2.1.4 Requisito funcional: Gestionar comentario

**Tabla 12 Especificación del requisito Adicionar Comentario**

| **Conceptos tratados** | **Conceptos** | **Atributos** |
|---|---|---|
| | No procede. | No procede. |
| **Precondiciones** | **Precondiciones** | **Pre-requisito** |
| | Tener soluciones creadas. | Adicionar solución. |
| **Descripción** | El usuario selecciona el método que desea comentar. El sistema le muestra un formulario para que especifique los datos (tipo de retorno, descripción y tipo de datos de los parámetros), todos con carácter opcional. El sistema adiciona un comentario al método especificado.<br>El sistema debe permitir cancelar esta acción. | |
| **Validaciones** | No procede. | |
| **Post-condiciones** | Se ha adicionado un comentario al método. | |
| **Post-requisito** | No procede. | |

**Tabla 13 Especificación del requisito Modificar Comentario**

| **Conceptos tratados** | **Conceptos** | **Atributos** |
|---|---|---|
| | No procede. | No procede. |

| **Precondiciones** | **Precondiciones** | **Pre-requisito** |
|---|---|---|
| | Tener soluciones creadas. | Adicionar solución. |
| **Descripción** | El usuario selecciona el método al cual desea modificar el comentario. El sistema muestra un formulario con los valores de los datos (tipo de retorno, descripción y tipo de datos de los parámetros) que posee el comentario actual, para que el usuario los modifique a conveniencia. En caso de que exista alguna modificación el sistema modifica el comentario del método.<br>El sistema debe permitir cancelar esta acción. | |
| **Validaciones** | No procede. | |
| **Post-condiciones** | Se ha modificado el comentario del método. | |
| **Post-requisito** | No procede. | |

### 2.2.2 Requisitos no funcionales

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. De esta manera los requisitos no funcionales a los que se debe acoger la aplicación a desarrollar son los que fueron establecidos por el centro al inicio del proceso de desarrollo, a continuación son descritos algunos de los más importantes.

#### 2.2.2.1 Rendimiento

Los tiempos de respuesta y velocidad de procesamiento de la información serán rápidos, no mayores 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 incluyendo el mantenimiento de las bases de datos, así como la salva de la información, se realizará de forma centralizada por el administrador.

#### 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 o superior.

#### 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.

## 2.3 Modelo de diseño

El Modelo de diseño es una abstracción del Modelo de Implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño. Es utilizado como entrada esencial en las actividades relacionadas a la implementación. El Modelo de Diseño puede contener: diagramas, clases, paquetes, subsistemas, cápsulas, protocolos, interfaces, relaciones, colaboraciones y atributos.

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.

### 2.3.1 Diagrama de clases

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación, contiene información como clases, 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.

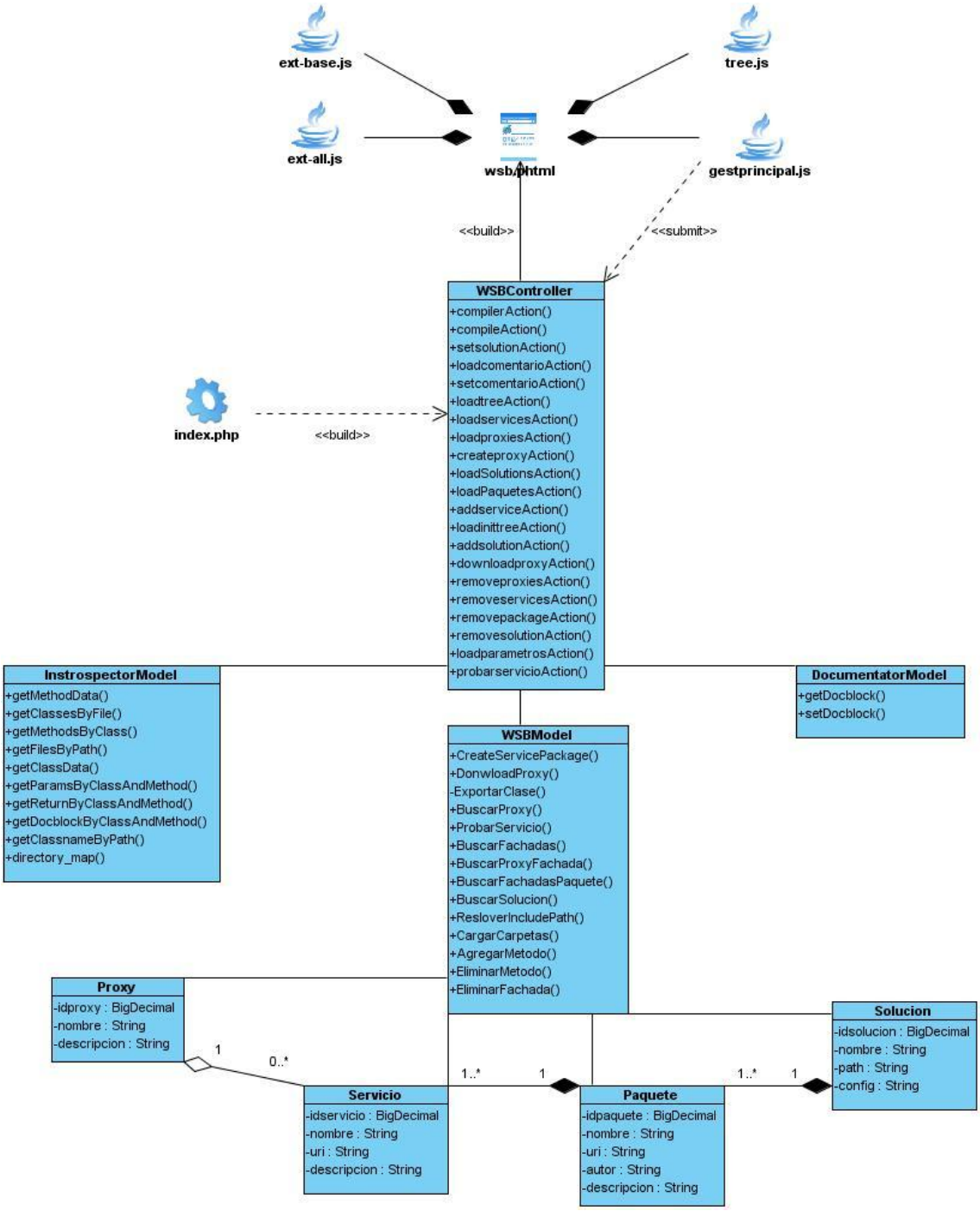



**Figura 5 Diagrama de Clases del Diseño.**

### 2.3.2 Patrones utilizados

#### 2.1.1.1 Patrón arquitectónico 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.

#### 2.1.1.2 Patrones de diseño

Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores y debe ser aplicable a diferentes problemas de diseño en distintas circunstancias (27).

Para hacer un diseño eficiente se tomaron en cuenta un conjunto de patrones, que al ser experiencias de diseñadores expertos en orientación a objetos permiten dar soluciones eficientes a problemas existentes, facilitando notablemente el trabajo posterior. Algunos de esto patrones son:

- Fachada

Es un patrón estructural y consiste en crear una única clase de manejo más fácil, que permita acceder a un conjunto numeroso y complicado de clases. La fachada satisface a la mayoría de los clientes, sin ocultar las funciones de menor nivel a aquellos que necesiten acceder a ellas (27). En la solución se pone de manifiesto este patrón en las clases pertenecientes al paquete “services” del componente las cuales sirven de fachada a las funcionalidades brindadas por el mismo al resto de los componentes.

- Solitario

Es un patrón creacional que tiene como propósito garantizar una única instancia de una clase, proporcionando un punto de acceso global a la misma (27). 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.

## 2.4 Modelo de datos

Un modelo de datos es una colección de conceptos bien definidos matemáticamente que ayudan a expresar las propiedades estáticas y dinámicas de una aplicación con un uso de datos intensivo (28).

El modelo de datos de la solución cuenta con 4 clases persistentes para un total de 7 tablas. Para su confección se tuvieron en cuenta los procesos de normalización y la utilización de nomencladores para facilitar su diseño. A continuación se muestra el diagrama entidad-relación que lo describe.

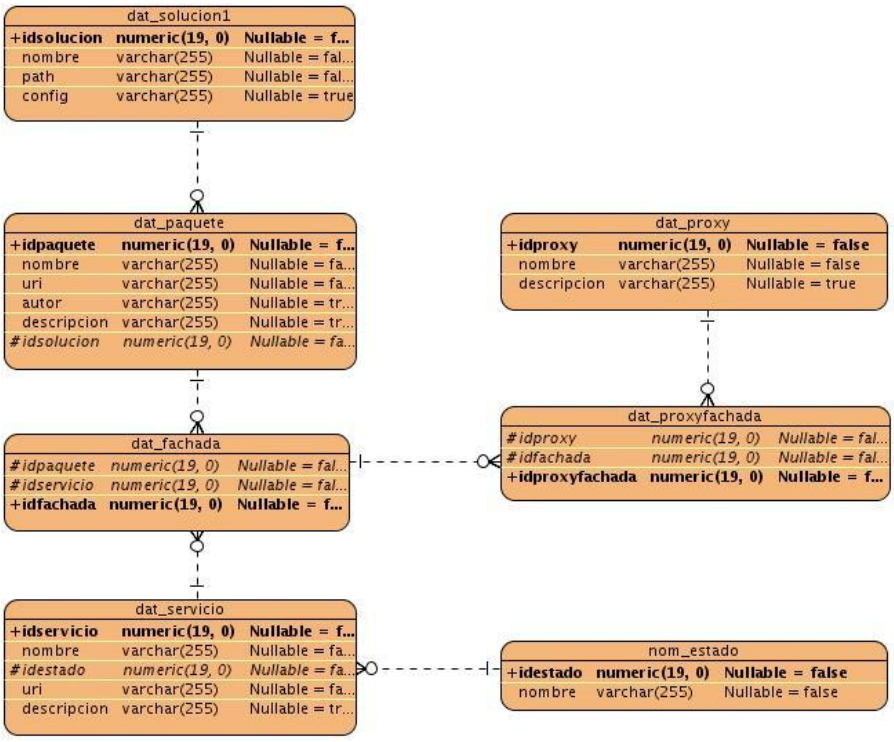



**Figura 6 Diagrama del Modelo de Datos.**

## 2.5 Conclusiones parciales

En este capítulo fueron expuestos los artefactos generados durante el diseño de la solución propuesta. En el mismo se mostró la arquitectura base que rige el diseño, el modelo de datos y el diagrama de clases del diseño propuesto por los autores. Una vez concluido el diseño de la solución puede darse paso a los flujos de implementación y prueba de la misma.

# 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 describe cómo los elementos del modelo de diseño, se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros, además de los recursos necesarios para poder ejecutar el sistema desarrollado.

### 3.1.1 Diagrama de componentes

La solución propuesta consta de un solo componente llamado WSB el cual interactúa con otros componentes del marco de trabajo como ZendExt_Service para la creación de los wsdl y con el IOC para la resolución de dependencias. A continuación se muestra el diagrama de componentes elaborado.

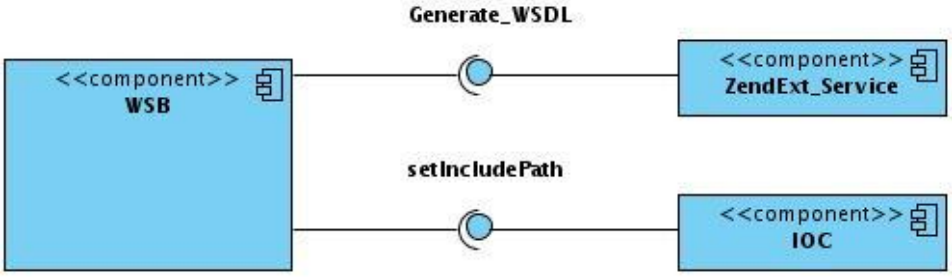



**Figura 7 Diagrama de Componentes.**

### 2.1.2 Diagrama de despliegue

El usuario, desde una estación de trabajo, podrá acceder al sistema el cual estará desplegado en el mismo servidor web donde se encuentre ubicada la aplicación a la cual se le desee generar servicios web. Dicho servidor estará conectado a un servidor de bases de datos en el cual se almacenará la información de interés para la solución. A continuación se muestra el diagrama de despliegue elaborado.


**Figura 8 Diagrama de Despliegue.**

## 3.2 Métricas de software

Las métricas de software son una medida cuantitativa que permite a los desarrolladores tener una visión profunda de la eficacia del proceso del software y de los proyectos que dirigen utilizando el proceso como un marco de trabajo. Se reúnen los datos básicos de calidad y productividad. Estos datos son entonces analizados, comparados con promedios anteriores, y evaluados para determinar las mejoras en la calidad y productividad. Las métricas son también utilizadas para señalar áreas con problemas de manera que se puedan desarrollar los remedios y mejorar el proceso del software (29).

Las métricas empleadas están diseñadas para evaluar los siguientes atributos de calidad:

- ***Responsabilidad.*** Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- ***Complejidad de implementación.*** Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- ***Reutilización.*** Consiste en el grado de reutilización presente en 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 del 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, modulo, clase, conjunto de clases, etc.) diseñado.

Las métricas escogidas como instrumento para evaluar la calidad del diseño descrito en el capítulo anterior y su relación con los atributos de calidad 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 14 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 15 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 16 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 17 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 |

### 3.2.1 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 75% 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 18 Instrumento de evaluación de la métrica TOC.**

| Clase | Cantidad de Procedimientos | Responsabilidad | Complejidad | Reutilización |
|---|---|---|---|---|
| WSBController | 21 | Alta | Alta | Baja |
| WSBModel | 14 | Alta | Alta | Baja |
| DocumentatorModel | 2 | Baja | Baja | Alta |
| IntrospectorModel | 10 | Media | Media | Media |
| DatFachadaModel | 4 | Baja | Baja | Alta |
| DatPaqueteModel | 7 | Media | Media | Media |
| DatProxyFachadaModel | 4 | Baja | Baja | Alta |
| DatProxyModel | 5 | Baja | Baja | Alta |
| DatServicioModel | 4 | Baja | Baja | Alta |
| DatSolucionModel | 5 | Baja | Baja | Alta |
| DatFachada | 5 | Baja | Baja | Alta |
| DatPaquete | 2 | Baja | Baja | Alta |
| DatProxy | 3 | Baja | Baja | Alta |
| DatProxyfachada | 4 | Baja | Baja | Alta |
| DatServicio | 2 | Baja | Baja | Alta |
| DatSolucion | 2 | Baja | Baja | Alta |

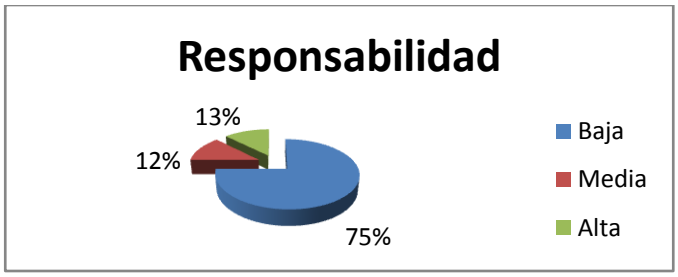



**Figura 9 Resultados de la evaluación de la métrica TOC para el atributo Responsabilidad.**

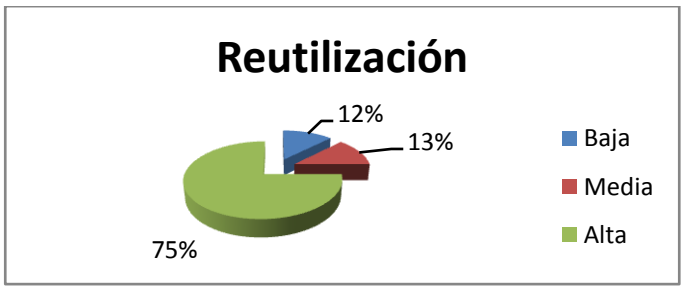



**Figura 10 Resultados de la evaluación de la métrica TOC para el atributo Reutilización.**

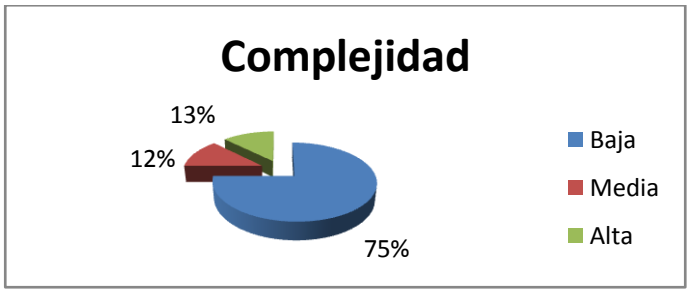



**Figura 11 Resultados de la evaluación de la métrica TOC para el atributo Complejidad.**

### 3.2.2 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 75% 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 19 Instrumento de evaluación de la métrica RC.**

| Clase | Cantidad de Relaciones de Uso | Acoplamiento | Complejidad Mant. | Reutilización | Cantidad de Pruebas |
|---|---|---|---|---|---|
| WSBController | 27 | Alto | Alta | Baja | Alta |
| WSBModel | 5 | Alto | Media | Media | Media |
| DocumentatorModel | 0 | Ninguno | Baja | Alta | Baja |
| IntrospectorModel | 0 | Ninguno | Baja | Alta | Baja |
| DatFachadaModel | 2 | Bajo | Baja | Alta | Baja |
| DatPaqueteModel | 6 | Alto | Media | Media | Media |
| DatProxyFachadaModel | 2 | Bajo | Baja | Alta | Baja |
| DatProxyModel | 5 | Alto | Media | Media | Media |
| DatServicioModel | 2 | Bajo | Baja | Alta | Baja |
| DatSolucionModel | 2 | Bajo | Baja | Alta | Baja |
| DatFachada | 0 | Ninguno | Baja | Alta | Baja |
| DatPaquete | 0 | Ninguno | Baja | Alta | Baja |
| DatProxy | 0 | Ninguno | Baja | Alta | Baja |
| DatProxyfachada | 0 | Ninguno | Baja | Alta | Baja |
| DatServicio | 0 | Ninguno | Baja | Alta | Baja |
| DatSolucion | 0 | Ninguno | Baja | Alta | Baja |

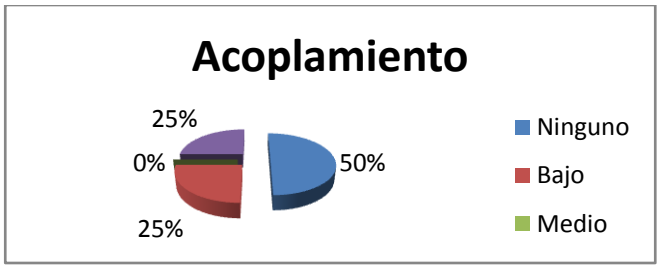



**Figura 12 Resultados de la evaluación de la métrica RC para el atributo Acoplamiento.**

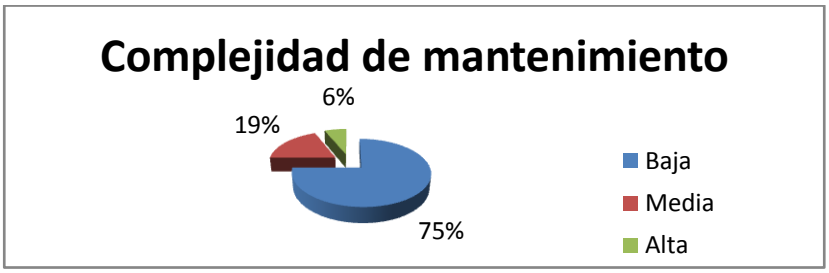



**Figura 13 Resultados de la evaluación de la métrica RC para el atributo Complejidad de Mantenimiento.**

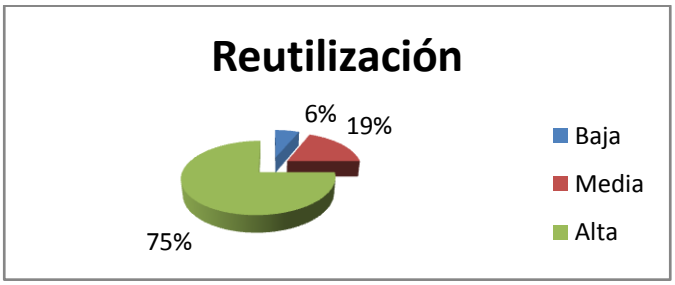



**Figura 14 Resultados de la evaluación de la métrica RC para el atributo Reutilización.**

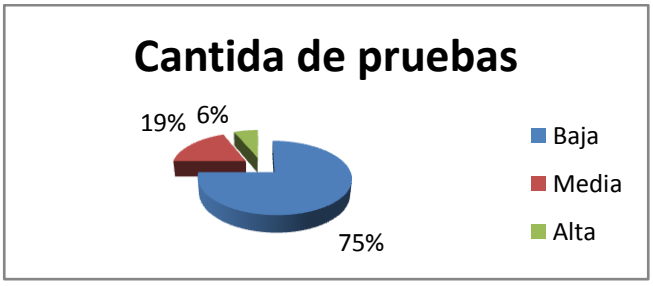



**Figura 15 Resultados de la evaluación de la métrica RC para el atributo Cantidad de Pruebas.**

### 3.2.3 Matriz de inferencia de indicadores de calidad

La matriz inferencia de indicadores de calidad, también llamada matriz de cubrimiento, es una representación estructurada de los atributos de calidad y métricas utilizadas para evaluar la calidad del diseño propuesto. Dicha matriz 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 general para cada atributo promediando todas sus relaciones no nulas.

A continuación se muestran los resultados obtenidos.

**Tabla 20 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 21 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 |

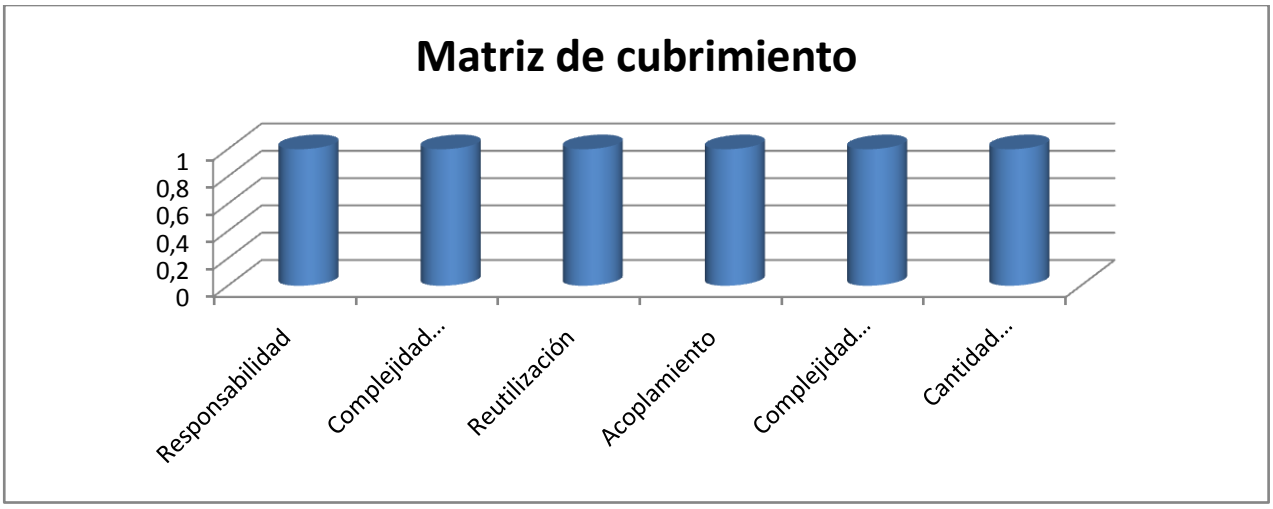



**Figura 16 Resultados obtenidos de la evaluación de los atributos de calidad.**

## 3.3 Pruebas de software

Las pruebas del software son un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación. Dichas pruebas son realizadas con el objetivo de detectar errores en el sistema, por lo que se llevan a cabo durante todo el ciclo de vida del producto.

Los casos de prueba especifican una forma de probar el sistema, incluyendo las entradas con las que se ha de probar, las condiciones bajo las que ha de probarse, así como los resultados esperados.

### 3.3.1 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.

El empleo de este tipo de pruebas permitirá diseñar casos de prueba que comprueben que todas las sentencias del sistema se ejecuten al menos una vez, además de todas las condiciones, tanto verdaderas como falsas. Para esto primero se procede a enumerar las sentencias del código y a partir del mismo se construye el grafo de flujo asociado.

```
function EliminarMetodo($str_clase, $str_nombreMetodo)
{
    $obj_class = Zend_CodeGenerator_Php_Class::fromReflection(new Zend_Reflection_Class($str_clase));1
    $obj_nueva = new Zend_CodeGenerator_Php_Class();1
    if($obj_class->getDocBlock() != null)2
        $obj_nueva->setDocBlock($obj_class->getDocBlock());3
    $obj_nueva->setName($obj_class->getName());4
    if($obj_class->isAbstract())5
        $obj_nueva->setAbstract(true);6
    $obj_nueva->setExtendedClass($obj_class->getExtendedClass());7
    $obj_nueva->setImplementedInterfaces($obj_class->getImplementedInterfaces());7
    $arr_metodos = $obj_class->getMethods();7
    $arr_nuevos = array();7
    foreach ($arr_metodos as $obj_metodo)8
    {
        if($obj_metodo->getName() != $str_nombreMetodo)9
            $arr_nuevos[] = $obj_metodo;10
    }11
    $obj_nueva->setMethods($arr_nuevos);12
    return $obj_nueva->generate();12
}
```

**Figura 17 Código fuente de la funcionalidad Eliminar Método.**

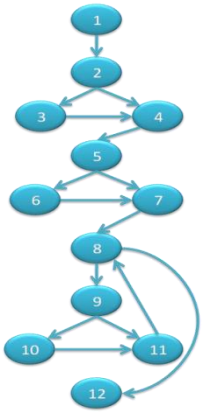



**Figura 18 Grafo de flujo asociado a la funcionalidad Eliminar Método.**

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) = (15 - 12) + 2

V (G) = 5

**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) = 4 + 1

V (G) = 5

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

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

V (G) = 5

El cálculo efectuado mediante las tres fórmulas ha dado el mismo valor, dando como resultado 5, lo que indica que existen 5 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: 1-2-4-5-7-8-12
- Camino básico #2: 1-2-3-4-5-7-8-12
- Camino básico #3: 1-2-4-5-6-7-8-12
- Camino básico #4: 1-2-4-5-7-8-9-11-8-12
- Camino básico #5: 1-2-4-5-7-8-9-10-11-8-12

Para cada camino se realiza un caso de prueba.

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

  **Descripción:** Los datos de entrada cumplirán con los siguientes requisitos: La clase y el nombre del método serán cadenas.

  **Condición de ejecución:** La clase se llamará Clase1 y el nombre del método será nulo. Clase1 debe ser una clase que no debe tener bloque de comentarios, no debe ser abstracta ni poseer métodos.

**Entrada:** $str_clase = ¨Clase1¨, $str_nombreMetodo = null

**Resultados esperados:** Una clase nueva exactamente igual a Clase1.

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

  **Descripción:** Los datos de entrada cumplirán con los siguientes requisitos: La clase y el nombre del método serán cadenas.

  **Condición de ejecución:** La clase se llamará Clase2 y el nombre del método será nulo. Clase2 debe ser una clase que debe tener bloque de comentarios, no debe ser abstracta ni poseer métodos.

  **Entrada:** $str_clase = ¨Clase2¨, $str_nombreMetodo = null

  **Resultados esperados:** Una clase nueva exactamente igual a Clase2.

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

  **Descripción:** Los datos de entrada cumplirán con los siguientes requisitos: La clase y el nombre del método serán cadenas.

  **Condición de ejecución:** La clase se llamará Clase3 y el nombre del método será nulo. Clase3 debe ser una clase que no debe tener bloque de comentarios, debe ser abstracta y no poseer métodos.

  **Entrada:** $str_clase = ¨Clase3¨, $str_nombreMetodo = null

  **Resultados esperados:** Una clase nueva exactamente igual a Clase3.

- **Caso de prueba para el Camino básico #4:**

  **Descripción:** Los datos de entrada cumplirán con los siguientes requisitos: La clase y el nombre del método serán cadenas.

  **Condición de ejecución:** La clase se llamará Clase4 y el nombre del método será MetodoBuscado. Clase4 debe ser una clase que no debe tener bloque de comentarios, no debe ser abstracta y poseer solo un método llamado MetodoBuscado.

  **Entrada:** $str_clase = ¨Clase4¨, $str_nombreMetodo = ¨MetodoBuscado¨

**Resultados esperados:** Una clase nueva exactamente igual a Clase4, difiriendo solamente en que el método buscado no existirá en esta nueva clase.

- **Caso de prueba para el Camino básico #5:**

  **Descripción:** Los datos de entrada cumplirán con los siguientes requisitos: La clase y el nombre del método serán cadenas.

  **Condición de ejecución:** La clase se llamará Clase5 y el nombre del método será MetodoBuscado. Clase5 debe ser una clase que no debe tener bloque de comentarios, no debe ser abstracta y poseer solo un método que no se llame MetodoBuscado.

  **Entrada:** $str_clase = ¨Clase5¨, $str_nombreMetodo = ¨MetodoBuscado¨

  **Resultados esperados:** Una clase nueva exactamente igual a Clase5.

Luego de aplicar los distintos casos de pruebas, se pudo comprobar que el flujo de trabajo de la función es correcto ya que cumple con las condiciones necesarias que se habían planteado.

### 3.3.2 Pruebas funcionales o de caja negra

Las llamadas pruebas de caja negra se basan en la especificación del programa o componente a ser probado para elaborar los casos de prueba. El componente es visto como una “caja negra” cuyo comportamiento es desconocido y sólo puede ser evaluado estudiando sus entradas y las salidas obtenidas. También son conocidas como pruebas de comportamiento o pruebas inducidas por los datos.

Algunas técnicas utilizadas en la prueba de caja negra son:

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

- ***Grafos de causa-efecto:*** Es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Para la aplicación de este tipo de pruebas se diseñaron un conjunto de escenarios de pruebas con el objetivo de evaluar el cumplimiento de los requisitos del sistema. A continuación se muestran los escenarios empleados.

**Tabla 22 Escenarios de prueba.**

| Nombre del requisito | Descripción general | Escenarios de pruebas | Flujo del escenario |
|---|---|---|---|
| **1: Gestionar Solución.** | Permite la gestión de las soluciones. | EP 1.1: Adicionar Solución. | Seleccionar la opción Crear una Nueva Solución y aceptar, va completando los datos y selcciona Siguiente, al final se muestra una interfaz de confirmación. |
| | | EP 1.2: Cargar Solución. | Seleccionar la opción Solución Existente escoge la solución que desea cargar en el menú desplegable y presiona el botón Aceptar. |
| | | EP 1.3: Eliminar Solución. | Seleccionar la opción Solución |

| | | | |
|---|---|---|---|
| | | | Existente escoge la solución que desea eliminar en el menú desplegable y presiona el botón Eliminar. |
| **2: Gestionar Comentarios.** | Permite la gestión de los comentarios de los métodos de las soluciones. | EP 2.1: Adicionar Comentario | Selecciona el método de la solución que desea comentar, presiona clic secundario y selecciona la opción Comentar. Llena los datos del formulario y presiona el botón Aceptar. |
| | | EP 2.2: Modificar Comentario | Selecciona el método de la solución al cual desea modificar el comentario, presiona clic secundario y selecciona la opción Comentar. Modifica los datos del formulario y presiona el botón |

| | | | |
|---|---|---|---|
| | | | Aceptar. |
| **3: Gestionar servicios.** | Permite la gestión de los servicios de las soluciones. | EP 3.1: Generar Paquete de servicios. | Escoge los métodos de la solución que se desean exportar como servicios web y presiona el botón Generar Paquetes. Llena los datos del formulario y presiona el botón Aceptar. |
| **4: Gestionar Proxy.** | Permite la gestión de los proxies que consumirán los servicios de las soluciones. | EP 4.1: Generar Proxy. | Selecciona el conjunto de servicios que desea que el proxy consuma y presiona el botón Generar Proxy. . Llena los datos del formulario y presiona el botón Aceptar. |
| | | EP 4.2: Descargar Proxy. | Selecciona los proxies que desea descargar y presiona el botón Descargar. |

La aplicación fue probada por el Departamento de Calidad del CEIGE, donde se comprobó el correcto funcionamiento de la misma, lo cual queda avalado por el acta de liberación de la herramienta expuesta en el anexo 2.

## 3.4 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 la culminación de la presente investigación se arriba las siguientes conclusiones:

- Las métricas orientadas a clases empleadas, validaron que se realizó un diseño correcto de la solución.
- Las pruebas estructurales y funcionales realizadas a la aplicación, validaron el correcto funcionamiento de la misma.
- La herramienta obtenida permitirá al CEIGE, así como a otros equipos de desarrollo que utilizan PHP como lenguaje de programación, agilizar los procesos de integración de sus aplicaciones con otras, lo cual acorta el tiempo de desarrollo de las mismas, reduciendo de esta manera los costos de producción.

La herramienta obtenida permitirá que productos claves para el desarrollo del país, como CEDRUX, puedan tener una cartera de servicios que les permita interoperar con otras aplicaciones en las entidades en las cuales son desplegados.

# RECOMENDACIONES

Una vez concluido el proceso de desarrollo se recomienda llevar a cabo la construcción de una segunda versión donde se incorporen las siguientes funcionalidades.

- Creación de proxies para otros lenguajes de programación altamente utilizados como c#, java y c++.
- Creación de proxies para el consumo de servicios web externos.

# REFERENCIAS

1. **Sandoe, K, Corbitt, G y Boykin, R.** *Enterprise Integration.* s.l. : John Wiley & Sons, 2001. 0471359939.

2. **Pérez, María, Mendoza, Luis E y Carvajal, Yorka.** *Orientaciones para la selección de tecnologías de integración de sistemas de software.* Universidad Simón Bolívar. Caracas, Venezuela : s.n.

3. **García, Ignacio, y otros.** *Servicios Web.* Toledo, España : Universidad de Castilla-La Mancha, 2005.

4. **Short, Scott.** *Creación de servicios web XML para la plataforma Microsoft .NET.* Madrid, España : McGraw-Hill, 2002. 978-84-481-3702-1.

5. **Alonso, Gustavo, y otros.** *Web Services and their Approach to Distributed Computing.* Berlin : Springer, 2004. 3-540-44008-9.

6. *XML Web Services Automation: A Software Engineering Approach.* **Nicoloudis, Nicholas y Mingins, Christine.** Gold Coast, Australia : IEEE Computer Society, 2002. Proceedings of the Ninth Asia-Pacific Software Engineering Conference (APSEC'02). 0-7695-1850-8.

7. *Web Services Computing: Advancing Software Interoperability.* **Chung, Jen-Yao, Lin, Kwei-Jay y Mathieu, Richard G.** 10, s.l. : IEEE Computer Society, 2003, Computer, Vol. 36. 0018-9162.

8. NetBeans Docs & Support. [En línea] [Citado el: 23 de Enero de 2010.] http://netbeans.org/kb/index.html.

9. ASP.NET Web Services. [En línea] [Citado el: 23 de Enero de 2010.] http://msdn.microsoft.com/en-us/library/t745kdsh.aspx.

10. Zend Studio - The Professional PHP IDE - Zend.com. [En línea] [Citado el: 23 de Enero de 2010.] http://www.zend.com/en/products/studio/.

11. *Spatial world chants the EAI mantra.* **Chutkay, Sumathi.** Hyderabad, India : s.n., 2003.

12. *New Developments in Practice IV: Managing the Technology Portfolio.* **McKeen, James D y Smith, Heather A.** 5, s.l. : Communications of the Association for Information Systems, 2002, Vol. 9.

13. **Slee, Mark, Agarwal, Aditya y Kwiatkowski, Marc.** *Thrift: Scalable crosslanguage services implementation.* Palo Alto, California : s.n., 2007.

14. **Francois, Romain y Eddelbuettel, Dirk.** *RProtoBuf: An R API for Protocol Bu_ers.* 2010.

15. **Diaz Peña, Omar Antonio.** *Diseño arquitectónico de una plataforma para la arquitectura distribuida en PHP basada en Servicios Web.* Universidad de las Ciencias Informáticas. La Habana : s.n., 2010.

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

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

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

19. *Secure Programming with the Zend-Framework.* **Esser, Stefan.** Amsterdam : s.n., 2009. Dutch PHP Conference.

20. **Aquino, Aylienn y Linares, Yasser.** *Implementación del módulo de Contabilidad General del Sistema Integral de Gestión Cedrux.* Universidad de las Ciencias Informáticas. La Habana : s.n., 2009.

21. Área temática de Linux. Portal de Linux - Ciberaula. [En línea] [Citado el: 15 de Diciembre de 2009.] http://linux.ciberaula.com.

22. **Prada Nicot, Héctor y Sánchez González, Kenner.** *Desarrollo de los componentes Puesto de Trabajo y Pagos Adicionales del subsistema Capital Humano integrado al sistema integral de gestión CEDRUX.* La Habana : s.n., 2009.

23. **Nuseibeh, Bashar y Easterbrook, Steve.** Requirements engineering: a roadmap. *Proceedings of the Conference on The Future of Software Engineering.* Limerick : ACM, 2000.

24. **IEEE.** *Compilation of IEEE Standard Computer Glossaries, 610-1990.* 1991. 1559370793 .

25. **Kotonya, Gerald y Sommerville, Ian.** *Requirements Engineering: Processes and Techniques.* s.l. : John Wiley & Sons, 1998. 978-0-471-97208-2.

26. **Sommerville, Ian y Sawyer, Pete.** *Requirements Engineering: A good practice guide.* Lancaster University. 1997.

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

28. *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.

29. **Pressman, Roger S.** *Ingenieria del Software - Un Enfoque Practico.* s.l. : McGraw-Hill Companies, 2002. 8448132149.

30. **IEEE Computer Society.** *Guide to the Software Engineering Body of knowledge.* 2004. 0-7695-2330-7.

31. **Bosh, Jan.** *Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach.* s.l. : Addison-Wesley Professional, 2000. 0201674947.

# ANEXOS

## Anexo 1: Prototipos de interfaz visual

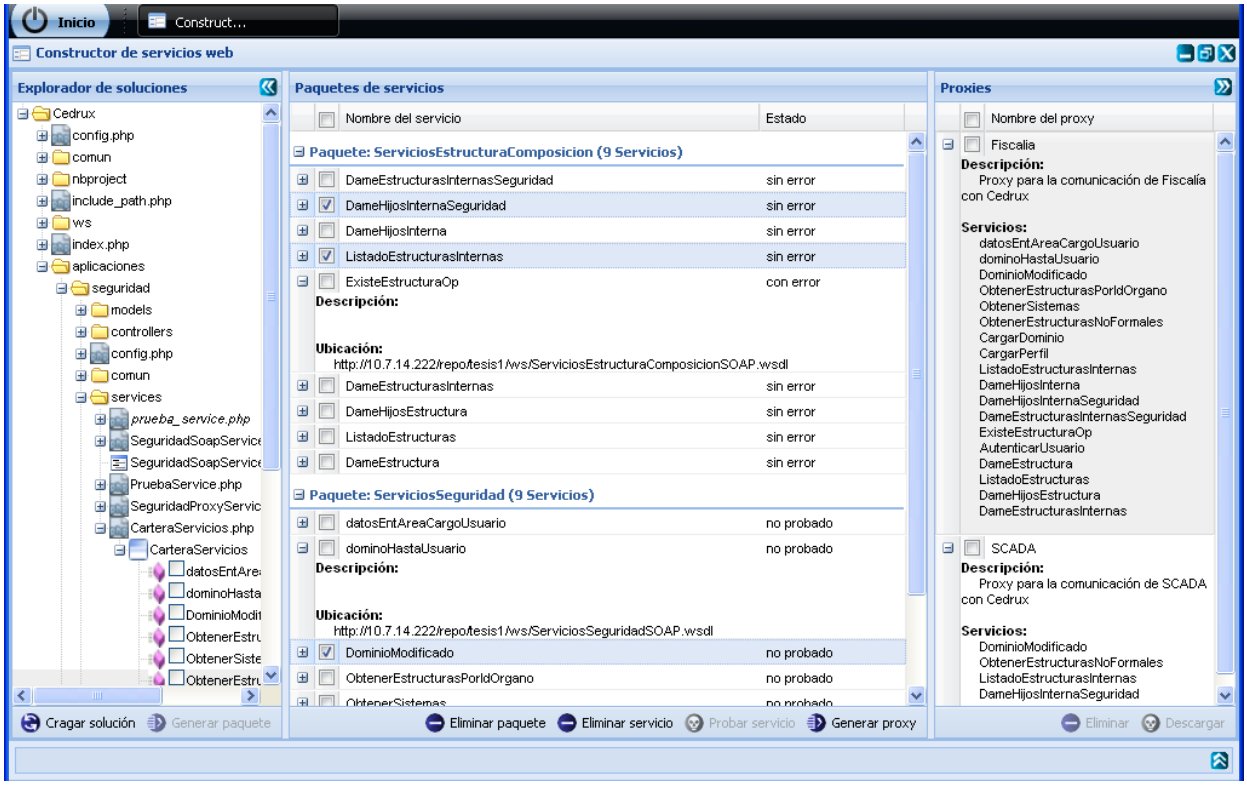



**Figura 19 Prototipo de interfaz principal.**

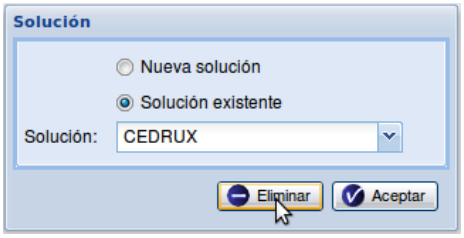



**Figura 20 Prototipo de interfaz Gestionar Solución.**

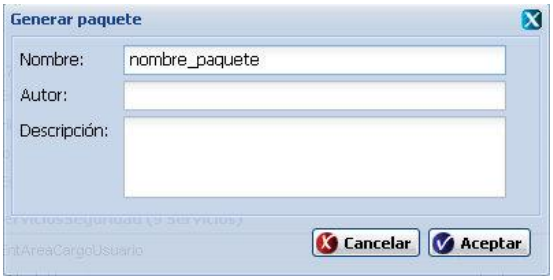



**Figura 21 Prototipo de interfaz Crear Paquete de Servicios.**

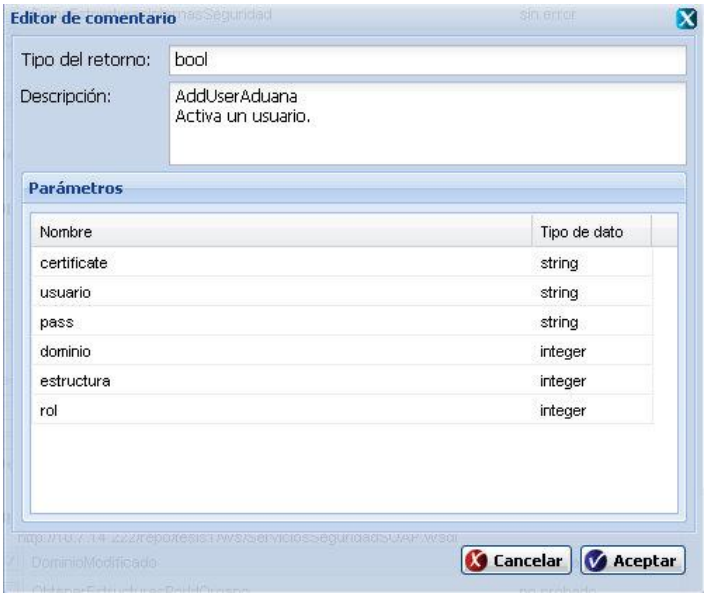



**Figura 22 Prototipo de interfaz Gestionar Comentario.**


**Figura 23 Prototipo de interfaz Crear Proxy.**

## Anexo 2: Acta de liberación del producto

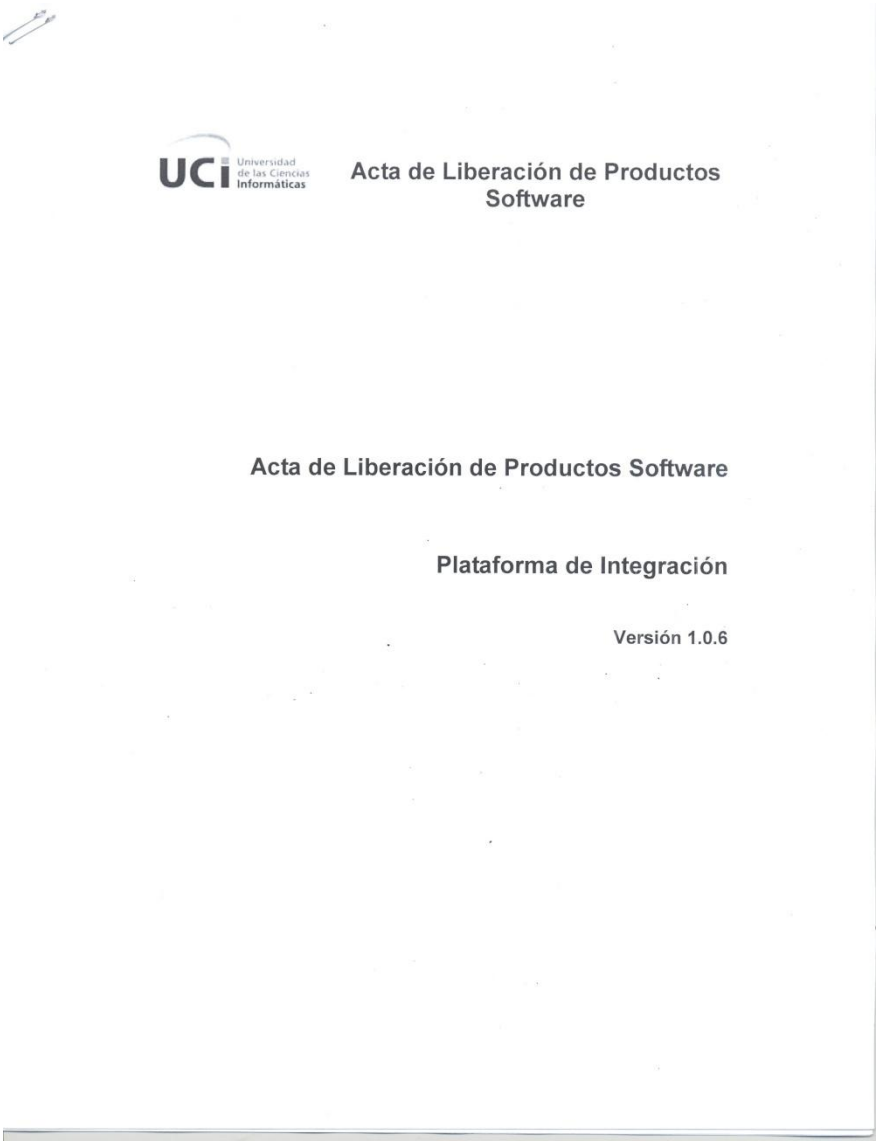
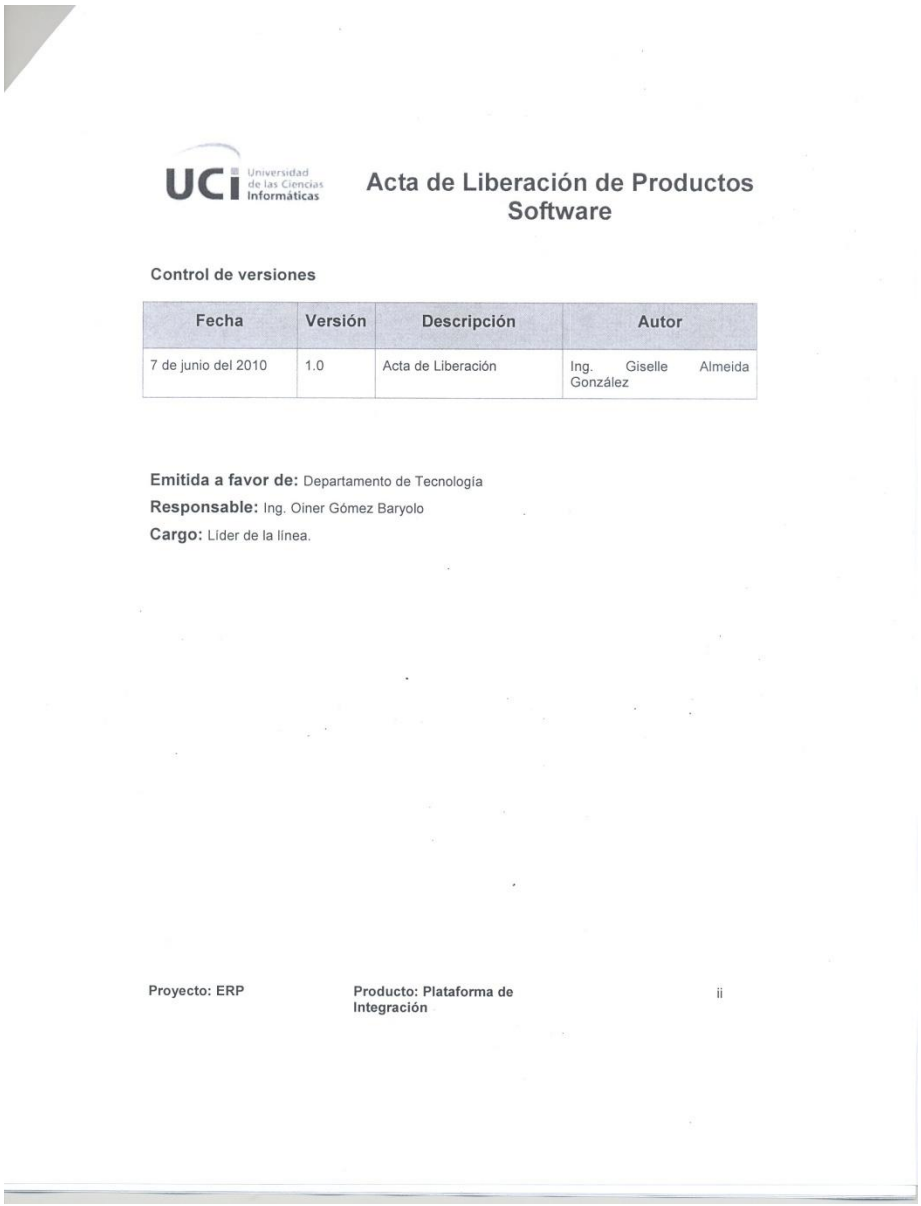



**Figura 24 Acta de Liberación, página 1 de 5.**



UCi Universidad de las Ciencias Informáticas

## Acta de Liberación de Productos Software

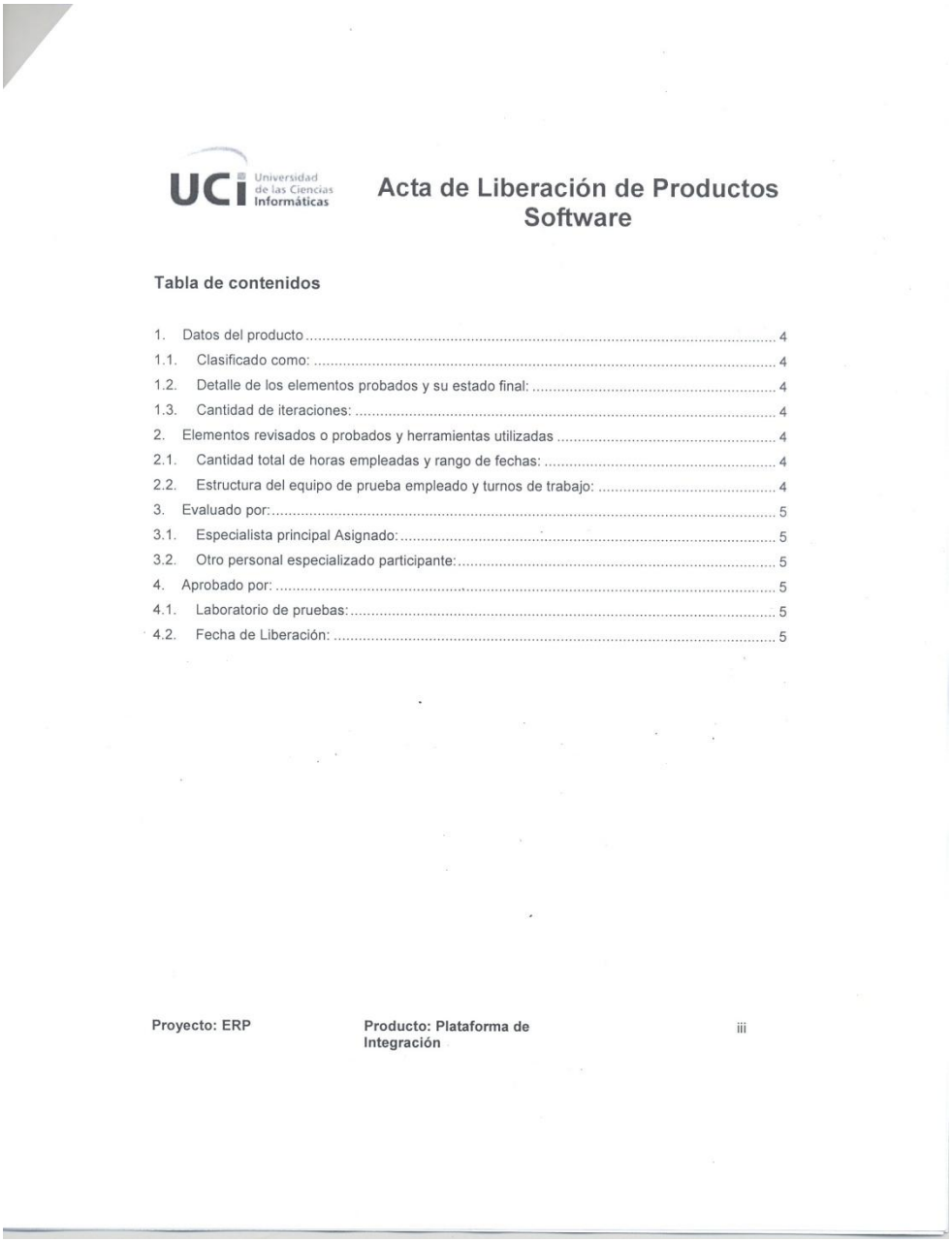
**Control de versiones**

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 7 de junio del 2010 | 1.0 | Acta de Liberación | Ing. Giselle Almeida González |

**Emitida a favor de:** Departamento de Tecnología
**Responsable:** Ing. Oiner Gómez Baryolo
**Cargo:** Líder de la línea.

**Proyecto: ERP** **Producto: Plataforma de Integración** ii

**Figura 25 Acta de Liberación, página 2 de 5.**



UCI Universidad de las Ciencias Informáticas

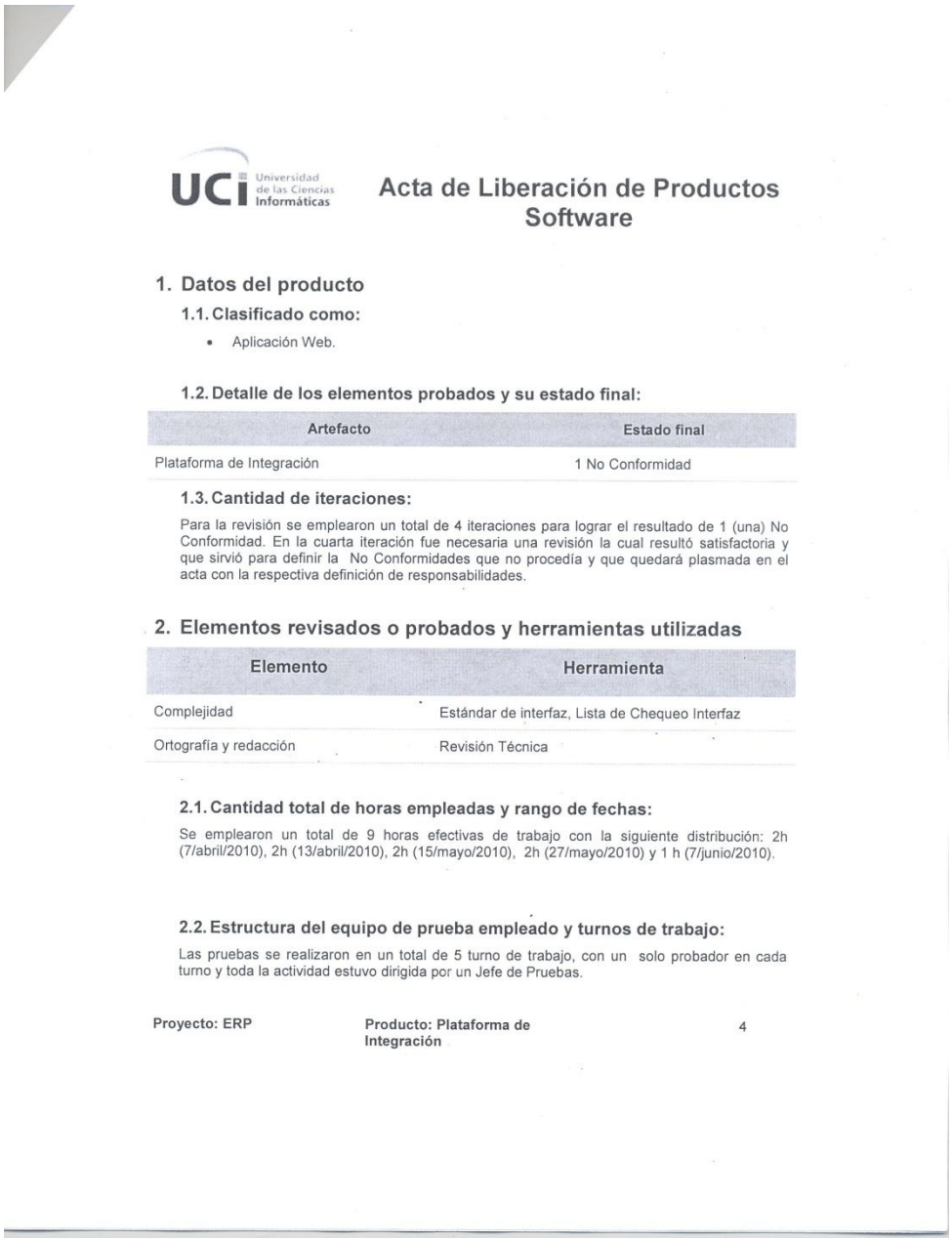
Acta de Liberación de Productos Software

Tabla de contenidos





**Figura 26 Acta de Liberación, página 3 de 5.**



UCI Universidad de las Ciencias Informáticas

## Acta de Liberación de Productos Software

### 1. Datos del producto

#### 1.1. Clasificado como:

- Aplicación Web.

#### 1.2. Detalle de los elementos probados y su estado final:

| Artefacto | Estado final |
|---|---|
| Plataforma de Integración | 1 No Conformidad |

#### 1.3. Cantidad de iteraciones:

Para la revisión se emplearon un total de 4 iteraciones para lograr el resultado de 1 (una) No Conformidad. En la cuarta iteración fue necesaria una revisión la cual resultó satisfactoria y que sirvió para definir la No Conformidades que no procedía y que quedará plasmada en el acta con la respectiva definición de responsabilidades.

### 2. Elementos revisados o probados y herramientas utilizadas

| Elemento | Herramienta |
|---|---|
| Complejidad | Estándar de interfaz, Lista de Chequeo Interfaz |
| Ortografía y redacción | Revisión Técnica |

#### 2.1. Cantidad total de horas empleadas y rango de fechas:

Se emplearon un total de 9 horas efectivas de trabajo con la siguiente distribución: 2h (7/abril/2010), 2h (13/abril/2010), 2h (15/mayo/2010), 2h (27/mayo/2010) y 1 h (7/junio/2010).

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

Las pruebas se realizaron en un total de 5 turno de trabajo, con un solo probador en cada turno y toda la actividad estuvo dirigida por un Jefe de Pruebas.

**Proyecto: ERP** **Producto: Plataforma de Integración** 4

**Figura 27 Acta de Liberación, página 4 de 5.**

UCI Universidad de las Ciencias Informáticas

## Acta de Liberación de Productos Software

**3. Evaluado por:**

**3.1. Especialista principal Asignado:**

Ing. Giselle Almeida González

**3.2. Otro personal especializado participante:**

| Nombre y apellidos | Rol ocupado o tarea |
|---|---|
| Ing. Iliannis Pupo Leyva | Especialista de Calidad |
| Mirta Juana del Río rivas | Probador |
| Javier Ruiz Durán | Desarrollador |

**4. Aprobado por:**

**4.1. Laboratorio de pruebas:**

Ing. Adieren Acosta Zamora

**4.2. Fecha de Liberación:**

Lunes, 7 de junio del 2010.

**Notas:**

- Se decide posponer la liberación del componente Plataforma de Integración teniendo todavía una NC no significativa por resolver teniendo en cuenta que no depende del desarrollador.

Ing. Giselle Almeida González
Especialista del Laboratorio de Calidad

Ing. Oiner Gómez Baryolo
Responsable por el Equipo de Desarrollo

**Proyecto: ERP** **Producto: Plataforma de Integración** 5

**Figura 28 Acta de Liberación, página 5 de 5.**