Universidad de las Ciencias Informáticas
Facultad 4

Componente para la gestión de servicios web en el marco de trabajo Xalix

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

Autores:
Yasmin Cid González
Leonel Hernández López

Tutor:
Ing. Ernesto Vladimir Pereda Díaz

La Habana, junio 2014
"Año 56 de la Revolución"

Declaración de autoría

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

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2014.

_____ _____

Yasmin Cid González
Firma del autor

Leonel Hernández López
Firma del autor

_____

Ing. Ernesto Vladimir Pereda Díaz
Firma del Tutor

Agradecimientos

Este trabajo de diploma no hubiera sido posible sin el apoyo y la entrega de muchas personas que de manera desinteresada se brindaron con el mayor interés y dedicación.

A nuestro tutor Ernesto por orientarnos y ayudarnos a lo largo de todo el desarrollo.

A nuestro tribunal Nilber, Augusto, y en especial a Mailyn por corregirnos nuestras fallas y tener siempre extendida su mano para nosotros.

A Yoandy por jugar los roles de consultor, tutor y oponente al mismo tiempo, siempre con el objetivo de mejorar nuestro trabajo.

A Irina y Orlando por estar pendiente de nuestras dudas. En general, a todos los profesores que contribuyeron a nuestro desarrollo como profesional y a la realización de esta tesis.

A todos Muchas Gracias.

De Yasmin:

A mi mamá por ser mi ejemplo a seguir, porque con su trabajo, sus sacrificios y sus ganas de abrirme camino en la vida ha sido la principal protagonista de este logro.

A mi papá por ser mi mayor inspiración, porque aunque no pude disfrutar de él cuanto quise, está siempre conmigo guiando cada uno de mis pasos, a él va dedicada esta tesis.

A Yusely por ser la mejor hermana del mundo y por estar siempre pendiente de mí.

A mis tíos por ser como unos padres para mí, por ayudarme a cumplir cada uno de mis sueños.

A mi novio Frank por su dedicación y entrega para que las cosas en mi vida marchen bien, por su ayuda para realizar esta tesis y por su amor incondicional y confianza.

A mis suegros por brindarme su ayuda incondicional siempre que los he necesitado.

A toda mi familia, por ser parte importante en mi vida y sentirse siempre orgullosos de mí.
A mi compañero de tesis Leonel por enseñarme a ser mejor cada día y por ser un gran amigo.

A mis amigos Claudia, Edgar, Camilo y Armando por compartir conmigo estos años y por cada cosa que he podido aprender de ellos.

A mis compañeras de apartamento Yaneisi, Massiel y Yuliexa por todos los buenos momentos que compartimos juntas.

A todas las personas maravillosas que he conocido durante estos 5 años.

De Leonel:

Quiero agradecer a todas las personas que permanecen siempre en mi mente y fueron mi fuerza en todo momento. Este trabajo para mí lleva sus nombres.

A mi abuela Amparo, que su preocupación, entrega y amor me impulsaron a realizar un sueño que también es el suyo.

A mi madre querida Regla, que no hay minuto que no piense en ella, que su sacrificio, amor y su esfuerzo por verme cada día mejor, dieron sus frutos.

A mi padre, hermano y amigo Pedro "Mi POX", que siempre creyó en mí, que nunca ha dejado de guiar mis pasos, esto no hubiera sido posible sin tu apoyo y amor incondicional.

A Adrian, Ansley, Mabel, Yamilé, Migue y Erick a todos ustedes muchas gracias por estar siempre en mi corazón, por confiar y dar todo su apoyo y amor por la realización de este sueño.

A mi familia, que no hay lugar en el mundo donde me sienta mejor que con todos ustedes juntos, no hay nada que quiera proteger más que eso. Por ustedes es que intento ser mejor cada día.

A mi novia Gloria que me supo entender en los momentos más críticos, que me brindó su ayuda incondicional, y que a pesar de las dificultades estuvo siempre dispuesta a darme lo mejor de ella.

A mi compañera de tesis Yasmin, que con el desarrollo de este trabajo no sólo cumplí un sueño sino también gané una gran amiga, que me enseñó a mirar las cosas desde un mejor punto de vista, y confió en mí en todo momento.

A mis amistades y hermanos de campaña que sin mencionarlos ustedes saben quiénes son. Porque compartimos, noches sin dormir, nervios, estrés y muchas experiencias, las cuales siempre estarán en mi memoria.

A todas aquellas personas que han contribuido en mi vida a que lograra mi sueño y el sueño de mi familia. A todos Muchas Gracias.

Resumen

El auge de las Tecnologías de la Información y las Comunicaciones ha posibilitado el desarrollo de la industria de software. La Universidad de las Ciencias Informáticas tiene varios centros de desarrollo entre los que se encuentra el Centro de Tecnologías para la Formación (FORTES), en este centro se están realizando un conjunto de transformaciones aplicando un modelo de desarrollo basado en Líneas de Productos de Software (LPS), para ello se creó la línea Ambiente Integrado de Aprendizaje (AIA), entre otras. La línea cuenta con un marco de trabajo llamado Xalix, este adopta una arquitectura basada en componentes y se creó con la idea de que sea la base para implementar plataformas educativas, constituyendo el objetivo de esta investigación, desarrollar un componente para la gestión de servicios web con un diseño basado en contrato, para garantizar la interoperabilidad entre sistemas creados dentro y fuera de este marco de trabajo. Durante el desarrollo de la investigación se estudiaron los principales estándares, tecnologías y herramientas utilizadas para la creación de los servicios web, seleccionando las adecuadas para la solución. Haciendo uso de la metodología de desarrollo Proceso Unificado de Desarrollo, se realizaron los artefactos propuestos por los flujos de trabajo Modelamiento de negocio, Requisitos, Análisis y Diseño, Implementación y Prueba. Finalmente, aplicándose tecnologías y estándares para garantizar la interoperabilidad entre sistemas, se obtuvo el componente funcional estructurado en forma de bundle de Symfony 2, que ofrece dentro de sus principales funcionalidades la gestión de los servicios web basado en un diseño Top down.

Palabras clave: diseño basado en contrato, interoperabilidad, servicios web, top down.

Índice

Introducción

El surgimiento y auge de las Tecnologías de la Información y las Comunicaciones (TIC) ha posibilitado el desarrollo de la industria de software. Existen numerosos equipos de trabajo en el mercado, lo que trae como consecuencia, la presencia de niveles de calidad y competitividad muy elevados, de manera que se hace sumamente importante lograr la plena satisfacción del cliente en el tiempo establecido.

La Universidad de las Ciencias Informáticas tiene entre sus principales objetivos informatizar el país y desarrollar la industria del software, para contribuir así al desarrollo económico. Para lograr sus objetivos conformó varios centros de desarrollo, en los cuales existen disímiles proyectos productivos.

En el Centro de Tecnologías para la Formación (FORTES) de la Facultad 4, se realizan un conjunto de transformaciones con el objetivo de facilitar y agilizar el trabajo en los proyectos productivos de la facultad. Como parte de la definición de la arquitectura de referencia para el desarrollo de aplicaciones en dicho centro, se ha aplicado un modelo de desarrollo basado en Líneas de Productos de Software (LPS) con el propósito de disminuir la diversidad tecnológica de las soluciones. Entre las líneas propuestas se encuentra Ambiente Integrado de Aprendizaje (AIA).

La línea AIA involucra los principales procesos o funcionalidades de los sistemas para la gestión del aprendizaje, sistemas para la gestión de contenidos de aprendizaje y repositorio de recursos y desarrolla componentes asociados a: Croda, Rhoda y la Plataforma Educativa Zera. Además, cuenta con un marco de trabajo denominado Xalix, que desde el punto de vista del desarrollo de software, es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado (1).

El marco de trabajo Xalix, incluye una base de tecnologías para el desarrollo y define una estructura y un mecanismo de integración para los componentes creados. Adopta una arquitectura basada en componentes, donde cada elemento puede desacoplarse y evolucionar independiente uno del otro y son esas partes las que conforman los productos genéricos una vez ensambladas.

En proyectos asociados a la línea AIA se han creado componentes de manera independiente para garantizar la interoperabilidad. Esto se debe a la ausencia de una solución estándar que garantice su reutilización en otros proyectos de la línea, evitando problemas de compatibilidad y altos costos de mantenimiento.

Una de las soluciones que facilita la interoperabilidad es el uso de servicios web, definido como un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la web. Estas intercambian datos entre sí con el objetivo de ofrecer servicios (2). Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios los solicitan llamando a estos procedimientos a través de la web (3). Para establecer un acuerdo entre proveedores y usuarios se crea un contrato, el mismo describe un servicio a través de los métodos que ofrece.

Existen herramientas que permiten, al implementar un servicio web, generar automáticamente el contrato, esto provoca que sus detalles dependan de la herramienta que se ha usado para crearlo. Si es necesario que el contrato sea generado usando una herramienta diferente o una versión superior a la utilizada, este puede cambiar y por ende los clientes de ese servicio.

Un ejemplo de la utilización de los servicios web para garantizar la interoperabilidad en los proyectos asociados a la línea AIA, es el componente sfAOServicesPlugin, creado en La Plataforma Educativa Zera. El mismo está desarrollado con el framework Symfony 1.4.20 que no tiene soporte desde finales del año 2012, además el contrato que se establece entre clientes y proveedores se genera automáticamente lo que trae como consecuencia altos costos de mantenimiento, problemas de incompatibilidad y desacuerdo.

Partiendo de este análisis se formula el siguiente **problema de investigación:** ¿Cómo permitir la interoperabilidad entre sistemas creados dentro y fuera del marco de trabajo Xalix?

Para solucionar el problema planteado se propone como **objetivo general**, desarrollar un componente para la gestión de servicios web, con un diseño basado en contrato, para garantizar la interoperabilidad entre sistemas creados dentro y fuera del marco de trabajo Xalix.

El **objeto de estudio** se enmarca en la interoperabilidad basada en servicios web, teniendo como **campo de acción**, servicios web con un diseño basado en contrato.

A partir del objetivo general definido, se derivan los siguientes **objetivos específicos:**

- Construir el marco teórico conceptual que sustenta la investigación.
- Analizar los elementos necesarios para diseñar el componente para la gestión de servicios web del marco de trabajo Xalix.

- Diseñar el componente para la gestión de servicios web del marco de trabajo Xalix.
- Implementar y probar el componente diseñado.

Se propone como **idea a defender:**

El desarrollo de un componente para la gestión de servicios web con un diseño basado en contrato, permitirá la interoperabilidad entre las aplicaciones creadas dentro y fuera del marco de trabajo Xalix.

Para dar cumplimiento a los objetivos específicos se proponen **tareas de investigación**:

- Estudio de los principales conceptos y tecnologías utilizadas en el desarrollo de servicios web.
- Análisis y definición de la metodología de desarrollo a utilizar, así como las herramientas de modelado.
- Análisis y definición de los lenguajes de programación a utilizar, así como las herramientas adecuadas que cumplan los requisitos para ser utilizadas en el desarrollo.
- Identificación y especificación de casos de uso del sistema.
- Elaboración del modelo de casos de uso que posibilita el cumplimiento de los requisitos funcionales y no funcionales identificados.
- Elaboración de diagramas de clases de análisis y diseño para cada caso de uso del sistema.
- Diseño del modelo de datos correspondiente a la aplicación.
- Confección de diagramas de componentes del sistema.
- Implementación de un componente para la gestión de servicios web en el marco de trabajo Xalix.
- Validación de la solución.

Con el fin de resolver y dar cumplimiento a los objetivos y las tareas propuestas se emplearon **métodos de investigación**.

Métodos teóricos:

- **Inductivo-Deductivo:** se utilizó en el análisis de otras soluciones que garantizan la interoperabilidad utilizando servicios web para comprender su arquitectura.

- **Histórico-Lógico:** permitió identificar tecnologías, lenguajes y metodologías de desarrollo a utilizar en la solución propuesta.
- **Modelación:** se utilizó para la realización de los artefactos correspondientes al análisis, diseño e implementación de las funcionalidades propuestas.

Métodos empíricos:

- **Observación:** este método permitió estudiar cómo funciona el proceso de interoperabilidad utilizando servicios web entre aplicaciones y los principales problemas asociados a este. Además, posibilitó realizar las pruebas al producto desarrollado.

Estructura capitular

Capítulo I: Fundamentación teórica.

En este capítulo se expone la fundamentación teórica de esta investigación. Se describen las soluciones informáticas que serán utilizadas (metodologías, modelo de desarrollo de software, tecnologías y herramientas).

Capítulo II: Análisis y diseño del sistema.

En este capítulo son descritos los principales conceptos relacionados con la investigación a través de un modelo de dominio. Se identifican los requisitos funcionales y no funcionales los cuales son agrupados en casos de uso. Se obtienen los diagramas de clases del análisis, de colaboración, así como los diagramas de clases de diseño, además es construido el modelo de datos y son realizadas las descripciones correspondientes.

Capítulo III: Implementación y prueba.

Se describen los principales aspectos del desarrollo**,** definen los tipos de pruebas y los casos de prueba realizados al software.

Capítulo I: Fundamentación teórica

1.1 Introducción

En la web existen muchas aplicaciones desarrolladas en tecnologías diferentes y resulta complicado lograr la interoperabilidad entre estos sistemas. Una de las soluciones para este problema fue la creación de servicios web. Con el objetivo de desarrollar un componente para gestionar servicios web con un diseño basado en contrato para la línea AIA, que sea capaz de comunicar aplicaciones y pueda ser reutilizado, se hace necesario realizar un estudio de los servicio web y de las tecnologías idóneas para su implementación, fundamentando de este modo las bases de la investigación.

1.2 Interoperabilidad

Según publicó el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE, por sus siglas en inglés) la interoperabilidad es la habilidad de dos o más sistemas, redes de comunicación, aplicaciones o componentes para intercambiar información entre ellos y para usar la información que ha sido intercambiada (4).

También ha sido definida como la capacidad que tiene un producto o un sistema, cuyas interfaces son totalmente conocidas para funcionar con otros productos o sistemas existentes, sin restricción de acceso o de implementación (5).

Gisela Sú y Jesús Carrasco definen interoperabilidad como la habilidad para intercambiar y compartir información entre sistemas informáticos con características, funciones y tecnologías distintas a través de interfaces conocidas (6).

La oficina de administración de la información del gobierno australiano, define la interoperabilidad como la capacidad de transferir y utilizar información de una manera uniforme y eficiente a través de múltiples organizaciones y sistemas de tecnologías de la información (7).

Para el desarrollo de esta investigación será considerada la interoperabilidad como: la habilidad de intercambiar información entre sistemas informáticos, independientemente del sistema operativo o lenguaje de programación en que hayan sido desarrollados.

Los servicios web se utilizan para garantizar la interoperabilidad entre sistemas informáticos.

1.3 Servicios web

En la actualidad existen varias definiciones de servicios web:

La comunidad World Wide Web Consortium (W3C) lo define como un conjunto de aplicaciones o tecnologías con capacidad para interoperar en la web que intercambian datos entre sí con el objetivo de ofrecer servicios, basados en estándares de comunicación para el transporte, la codificación y el intercambio de datos entre sistemas heterogéneos (2).

Los servicios web permiten que las aplicaciones compartan información y que además invoquen funciones de otros sistemas independientemente de cómo se hayan creado estos, cuál sea el sistema operativo o la plataforma en que se ejecutan y cuáles los dispositivos utilizados para obtener acceso a ellos (8).

Un servicio web es un componente software que puede ser registrado, descubierto e invocado mediante protocolos estándares de Internet (9).

Por lo que se puede considerar que un servicio web es una tecnología que utiliza un conjunto de protocolos y estándares, en redes de ordenadores como Internet, que permiten intercambiar datos entre aplicaciones independientemente del lenguaje de programación o sistema operativo en que se desarrollaron.

Entre las principales ventajas que presenta la utilización de los servicios web se destacan las siguientes (8):

- Los servicios web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Permiten la interoperabilidad entre plataformas de distintos fabricantes por medio de protocolos estándar y abiertos. Las especificaciones son gestionadas por una organización abierta, la W3C, por tanto no hay secretismo por intereses particulares de fabricantes concretos y se garantiza la plena interoperabilidad entre aplicaciones.

- Se basan en HTTP[1] sobre TCP[2] en el puerto 80, dado que las organizaciones protegen sus redes mediante cortafuegos que filtran y bloquean gran parte del tráfico de internet.

1.3.1 Protocolos de comunicación entre aplicaciones

Los servicios web son publicados por un proveedor y solicitados por un cliente. El intercambio de información entre ambos está guiado por un protocolo que deben utilizar para que este proceso se realice correctamente.

Entre los protocolos creados para desarrollar servicios web se encuentran Remote Procedure Call sobre XML (XML-RPC), Simple Object Access Protocol (SOAP), REpresentational State Transfer (REST), Remote Procedure Call sobre JavaScript Object Notation (JSON-RPC) y Action Message Format 3 (AMF3). Cada uno es incompatible con otro, o sea, la interoperabilidad estará garantizada si clientes y proveedores utilizan el mismo protocolo de comunicación. A continuación se realiza una breve caracterización de cada uno de ellos.

1.3.1.1 XML-RPC

XML-RPC es un protocolo basado en XML que usa mensajes de llamadas a procedimientos remotos (RPC). Los requerimientos son codificados en XML y enviados vía HTTP POST, las respuestas vienen embebidas en mensajes RESPONSE de HTTP. XML-RPC es independiente de la plataforma (10).

XML-RPC define de manera sencilla cómo enviar el nombre de un método y una lista de argumentos de un sistema a otro. Su idea fundamental es que un documento XML sea utilizado para indicar el nombre de un método y la lista de argumentos que este necesita. Este documento XML es enviado a un servidor web utilizando HTTP POST. El servidor procesa el documento XML y ejecuta el método solicitado, retornando el resultado en otro

[1] Protocolo de Transferencia de Hipertexto: protocolo de comunicaciones que permite la transferencia de documentos de lenguaje de marcas de hipertexto (HTML) desde servidores web a navegadores web.

[2] Protocolo de Control de Transmisión: es un protocolo orientado a conexión, debe asegurar que los datos se transmiten y reciben correctamente por los computadores atravesando las correspondientes redes.

documento XML. Los parámetros del procedimiento pueden ser escalas, números, cadenas, fechas, entre otras (11).

Finalmente, XML-RPC es un protocolo fácil de usar, independiente del ambiente donde se ejecute y del sistema operativo utilizado, utiliza los métodos HTTP para pasar información entre los clientes y el servidor. Puede ser utilizado por desarrolladores sin mucha experiencia. Microsoft al considerar su simpleza decidió añadirle un conjunto de funcionalidades que contribuyeron a su evolución, convirtiéndolo posteriormente en SOAP.

1.3.1.2 SOAP

SOAP es un protocolo ligero diseñado para el intercambio de información estructurada en un entorno descentralizado y distribuido. Utiliza lenguaje de marcas extensible (XML) para la codificación de los mensajes que pueden ser intercambiados a través de varios protocolos de transporte como HTTP, SMTP[3], entre otros. Es independiente del lenguaje de programación y consta de tres partes: una envoltura que define un marco para describir lo que está en un mensaje y cómo procesarlo, un conjunto de reglas de codificación para expresar instancias a solicitudes de tipos de datos definidos y una convención para representar llamadas a procedimientos remotos y respuestas (12).

Un mensaje SOAP es una unidad básica de comunicación entre dos nodos, el emisor que es el nodo que transmite un mensaje y el receptor que constituye el nodo que lo recibe.

Es muy útil utilizar SOAP cuando es necesario crear un contrato formal para la descripción del servicio web, está basado en XML, y permite intercambiar información entre aplicaciones heterogéneas utilizando protocolos de transporte. Provee un sencillo marco para los mensajes, cuyo núcleo está pensado para ser extensible de forma que se puedan agregar nuevas funcionalidades. Además, contiene un elemento que se utiliza para llevar información de error dentro de los mensajes SOAP. No provee directamente ningún mecanismo para realizar transacciones con control de acceso, confidencialidad, integridad y no rechazo, aunque una forma de posibilitar estas medidas, es mediante extensiones SOAP. Existen en la actualidad muchas herramientas y componentes que se

[3] Protocolo para la Transferencia Simple de Correo Electrónico: Es el mecanismo de transmisión subyacente que utilizan la mayoría de los sistemas para enviar correo entre servidores en Internet.

encargan de gestionar servicios web utilizando este protocolo pues como se ha descrito hasta este momento brinda muchas ventajas, entre las que se destacan las siguientes:

- No está asociado a ningún lenguaje de programación.
- SOAP no especifica una interfaz de API[4], su implementación depende del lenguaje de programación que se elija.
- No se encuentra fuertemente asociado a ningún protocolo de transporte, depende de la capacidad que tenga el protocolo deseado para transmitir cadenas de texto.
- No está ligado a ninguna infraestructura de objetos distribuidos.
- Aprovecha los estándares existentes en la industria, ya que al crear SOAP se tomaron estos y se ampliaron.
- Permite la interoperabilidad entre múltiples entornos.

1.3.1.3 REST

REST es un estilo de arquitectura de software para sistemas hipermedias distribuidos tales como la web. Se refiere estrictamente a una colección de principios para el diseño de arquitecturas en red. Estos principios resumen cómo los recursos son definidos y diseccionados. El término frecuentemente es utilizado en el sentido de describir a cualquier interfaz que transmite datos específicos de un domino sobre HTTP sin una capa adicional (9).

Para esta investigación se considerará REST como una arquitectura de software que solicita y manipula recursos en la web utilizando los métodos de HTTP.

Para crear un servicio REST se necesita de una clase que preferiblemente contenga solo cuatro métodos públicos cada uno con una tarea específica:

1. GET: para obtener un valor, puede ser un listado de objetos.
2. POST: para guardar un nuevo objeto en la aplicación.
3. DELETE: para eliminar un objeto.
4. PUT: para actualizar un objeto.

[4] Interfaz de Programación de Aplicaciones**:** conjunto de convenios de llamada en programación que definen cómo se invoca un servicio a través de la aplicación.

Los principios de este estilo de arquitectura REST se listan a continuación (13):

- Escalabilidad de la interacción con los componentes. La web ha crecido exponencialmente sin degradar su rendimiento. Una prueba de ellos es la variedad de clientes que pueden acceder a través de la web: estaciones de trabajo, sistemas industriales y dispositivos móviles.
- Generalidad de interfaces. Cualquier cliente puede interactuar con cualquier servidor HTTP sin ninguna configuración especial.
- Puesta en funcionamiento independiente. Este hecho es una realidad que debe tratarse cuando se trabaja en Internet. Los clientes y servidores pueden ser puestos en funcionamiento durante años. Por tanto, los servidores antiguos deben ser capaces de entenderse con clientes actuales y viceversa. Diseñar un protocolo que permita este tipo de características resulta muy complicado. HTTP permite la extensibilidad mediante el uso de las cabeceras, a través de las URI[5], a través de la habilidad para crear nuevos métodos y tipos de contenido.
- Compatibilidad con componentes intermedios. Estos componentes son llamados proxys para web como las caches que se utilizan para mejorar el rendimiento y los cortafuegos que permiten reforzar las políticas de seguridad. Esto posibilita reducir la latencia de interacción y fortalecer la seguridad.

Para diseñar un servicio web usando REST es necesario identificar las funcionalidades que se expondrán como servicio, luego se crean las URL[6] teniendo en cuenta que no pueden contener verbos, seguidamente se categorizan los recursos, de manera que sea posible determinar si los clientes pueden obtener una representación del recurso o si pueden modificarla. Para el primero, se debe hacer los recursos accesibles utilizando un HTTP GET, para el último utilizando HTTP POST, PUT y DELETE. Es importante describir la forma de invocar el servicio usando un contrato, no dejar ninguna representación aislada y especificar el formato de los datos de respuesta mediante un esquema (9).

[5] Identificador Uniforme de Recursos: cadena de caracteres corta que identifica inequívocamente un recurso.

[6] Localizador de Recursos Uniforme: es una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se utiliza para nombrar recursos en Internet para su localización o identificación.

1.3.1.4 JSON-RPC

JSON-RPC es un protocolo de llamadas a procedimientos remotos codificadas en JSON (acrónimo de JavaScript Object Notation) similar a XML-RPC. Permite el envío de notificaciones y llamadas múltiples al servidor que pueden ser respondidas en orden o no, donde cada solicitud invoca un servicio remoto a través de HTTP (14).

JSON es un formato de texto para intercambio de datos, tal como XML, solo que mucho más liviano. Se utiliza para representar estructuras de datos simples llamados objetos y arreglos asociativos.

1.3.1.5 AMF 3

AMF 3 es un formato binario compacto que se utiliza para serializar gráficos de objetos ActionScript. Una vez serializado un gráfico de objeto codificado en AMF se puede utilizar para conservar y recuperar el estado de la aplicación a través de sesiones o permitir la comunicación de dos sistemas a través del intercambio de datos. Además, soporta el envío de objetos complejos por referencia, ayudando a evitar el envío de instancias redundantes (15).

Protocolos a utilizar en la solución propuesta

Las grandes empresas que lideran el mercado de la informática y son más populares en la red, basan sus servicios web en los protocolos SOAP y REST. Empresas tales como: Google (con su popular API Google Maps y Google SOAP Search API), Youtube, Amazon (con su API Amazon S3), Facebook, Twitter, PayPal, OpenSocial, Flickr, entre otros. También estos protocolos son usados en el mundo del marketing a través de las APIs: Api de Zoho Reports, Google BigQuery, Google Chart Tools, entre otras. En esta investigación serán empleados SOAP y REST para el intercambio de información, ya que son muy usados a nivel mundial y los más utilizados en la línea AIA, donde los productos relacionados a ésta trabajan con alguno de estos protocolos o ambos. Es decir, la implementación de SOAP y REST en el componente para el marco de trabajo Xalix es objetivo principal de la línea.

1.3.2 Tecnologías de servicios web

Los servicios web utilizan XML para codificar la información a intercambiar y para generar contratos, que varían dependiendo del protocolo utilizado, para describir servicios SOAP se creó el Lenguaje de Descripción de servicios Web (WSDL, por sus siglas en inglés) y

para REST existe el Lenguaje de Descripción de Aplicación Web (WADL, por sus siglas en inglés). Estos contratos son publicados en Catálogos de Negocios de Internet (por sus siglas en inglés UDDI) a través de los cuales pueden ser accedidos por clientes que necesiten consumir el servicio. A continuación se describen las tecnologías mencionadas.

1.3.2.1 XML

Según la W3C, XML es un Lenguaje de Etiquetado Extensible muy simple pero estricto, que juega un papel fundamental en el intercambio de una gran variedad de datos. Es un lenguaje muy similar a HTML pero su función principal es describir datos y no mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones (16).

En resumen XML constituye un lenguaje simple que utiliza etiquetas para estructurar el conocimiento, facilitando su validación e interpretación. Es ampliamente utilizado para desarrollar servicios web.

1.3.2.2 WSDL

WSDL está basado en XML y permite la descripción de los servicios web desplegados, además se utiliza para la localización de estos servicios en Internet. Un documento WSDL no es más que un documento XML que describe ciertas características propias de un servicio web, así como su localización y aquellos parámetros y métodos que soporta (17).

Los elementos principales de WSDL para la definición de los servicios son (18):

- Tipos: permite definir tipos empleando algún sistema de tipos, como XML Schema[7].
- Mensaje: una descripción de los datos intercambiados cuando se hace uso de los servicios.
- Operación: una funcionalidad ofrecida por el servicio.
- Tipo de puerto: un conjunto de operaciones abstractas ofrecidas por uno o más puntos finales.
- Vinculaciones: un protocolo de comunicaciones y un formato de mensajes concretos utilizados por un tipo de puerto.

[7] XML Schema: estándar con el propósito de definir la estructura de los documentos XML que estén asignados a tal esquema y los tipos de datos válidos para cada elemento y atributo.

Existen varias versiones de WSDL las cuales brindan un conjunto de opciones para crear contratos. Entre ellas se encuentran WSDL v1.0, v1.1 y v2.0. La versión 1.0 fue desarrollada por IBM y Microsoft para describir sus servicios web SOAP. La versión 1.1 es la formalización de WSDL 1.0 a la cual no fueron introducidos cambios significativos con respecto a la versión 1.0. Las versiones 1.0/1.1 definen enlaces que describen cómo utilizar WSDL junto con SOAP 1.1, HTTP GET y POST, y MIME[8]. WSDL 1.1 tiene algunos inconvenientes, entre ellos una estructura excesivamente compleja que hace que sea algo ilegible y además no cuenta con una definición formal de autoridad. Aunque imperfecta esta versión es lo suficientemente buena para la mayoría de los propósitos y constituye la forma más utilizada para la descripción de servicios web.

WSDL 2.0 fue liberado primeramente como WSDL 1.2 y renombrado a 2.0 porque existe una diferencia substancial con respecto a su versión anterior como por ejemplo: la etiqueta “PortType” fue cambiado a “Interface”, “Port” sustituido por “Endpoint”, además, acepta todos los métodos de petición HTTP (no solamente GET y POST como en la versión 1.1) y ofrece mejor soporte para los servicios web RESTful, también es mucho más simple de implementar. Además, tiene soporte para patrones de mensajes adicionales, soporta SOAP 1.2, proporciona una mejor y más potente notación para el manejo de errores, entre otras características. En la actualidad WSDL 2.0 cuenta con muy poco apoyo y al no ser implementado por muchas aplicaciones se convierte en poco interoperable.

Se puede concluir que el WSDL es un contrato que permite describir servicios web para que estos puedan ser utilizados en la red. Contiene información sobre un conjunto de elementos que resulta esencial conocer para poder comunicar sistemas heterogéneos. Teniendo en cuenta que WSDL 1.1 es más interoperable, porque resulta más utilizado en el mundo que WSDL 2.0, para describir servicios SOAP en esta investigación se utilizará WSDL 1.1.

[8] Multipurpose Internet Mail Extensions o MIME: serie de convenciones o especificaciones dirigidas al intercambio a través de Internet de todo tipo de archivos de forma transparente para el usuario.

1.3.2.3 WADL

WADL es un XML que sirve para describir servicios REST, modela los recursos proporcionados por el servicio y las relaciones entre ellos. WADL tiene la intención de simplificar la reutilización de servicios web que se basan en la arquitectura HTTP de la web. Es independiente de la plataforma y lenguaje de programación y tiene como objetivo promover la reutilización de aplicaciones más allá del uso básico en un navegador web. WADL es el equivalente REST de WSDL para SOAP. WADL es ligero, fácil de entender y de escribir. En ocasiones no resulta tan flexible como WSDL pero es muy útil para generar el contrato asociado a cualquier servicio REST.

El servicio se describe mediante un conjunto de elementos "resource". Cada recurso contiene elementos "param" para describir las entradas y elementos "method" que describen la petición y la respuesta de un recurso. El elemento "request" especifica la forma de representar la entrada, qué tipos son necesarios y las cabeceras HTTP específicas que se requieren. La respuesta, elemento "response", describe la representación de la respuesta del servicio, así como cualquier información de fallo, para hacer frente a los errores.

WADL es el lenguaje de descripción de servicios creado para servicios web RESTful, estos también pueden ser descritos por WSDL 2.0, pero es un lenguaje orientado principalmente a SOAP. NetBeans IDE utiliza para registrar los servicios REST el documento WADL, y en general el lenguaje de programación Java lo implementa, creando para esto herramientas como wadl2java que permite hacer clientes y servidores a partir del WADL, entre otras funciones. A otros lenguajes de programación les falta camino para llegar a utilizarlo como estándar, pero se cree que este será el lenguaje para describir servicios REST por excelencia. En esta investigación se utilizará este lenguaje para la descripción de los servicios REST, siguiendo las prácticas de Java, lenguaje avanzado en lo que respecta a servicios web.

1.3.2.4 UDDI

UDDI es uno de estándares básicos de los servicios web. Está diseñado para ser interrogado por mensajes SOAP y proveer acceso a documentos WSDL, en los que se describen los requisitos del protocolo y los formatos del mensaje solicitado para interactuar con los servicios web del catálogo de registros (19).

UDDI es un único registro conceptual distribuido a lo largo de multitud de nodos que replican la información unos de otros. UDDI pretende ser la solución al descubrimiento y la integración de servicios web, que de otra forma, sería una labor inabordable (20).

UDDI es similar a un motor de búsqueda de Internet para procesos de negocio. Permite buscar uno o más registros UDDI para ver los diferentes servicios web que se exponen y las especificaciones de estos.

1.3.3 Análisis de las soluciones similares

Con el objetivo de obtener conclusiones que ayuden en el desarrollo de la investigación, se hace referencia a un conjunto de soluciones que permiten gestionar servicios web.

1.3.3.1 Componente para la gestión de los servicios web en Eclipse

Eclipse es un Entorno de Desarrollo Integrado (IDE[9]) de código abierto multiplataforma para desarrollar aplicaciones. Brinda la posibilidad de crear servicios web de manera sencilla, usando una interfaz visual. Para ello cuenta con dos opciones: Top down Java bean Web Service o **Top down**, que permite crear el código de la clase Java a partir del WSDL y Bottom up Java bean Web Service o **Bottom up**, para crear el WSDL a partir del código de la clase en Java.

1.3.3.2 Apache Axis

Apache Axis es una implementación a código abierto de SOAP que proporciona un entorno de ejecución para servicios web. Es una implementación sólida, madura y extendida para ejecutar, testear y administrar servicios web implementados en Java (21).

Axis trae una herramienta para crear esqueletos de servicios a partir del WSDL, tanto para los clientes como para los servicios, llamada WSDL2Java. Esta herramienta puede presentar problemas en ocasiones, ya que el código que genera a veces no compila, y cuando compila puede no cumplir con la interfaz definida por el WSDL. Pero en muchos casos sí funciona bien, y cuando no lo hace da una aproximación al resultado que puede ser completado luego.

[9] IDE: Es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

Entre otras funcionalidades Axis proporciona un entorno de ejecución para servicio web Java. Herramientas para crear WSDL desde clases java, para crear clientes Java desde un WSDL, para desplegar, probar y monitorizar servicios web.

1.3.3.3 Herramientas de Altova MissionKit para gestionar servicios web

Contiene un conjunto de herramientas privativas que puede crear servicios web de forma visual en una interfaz gráfica intuitiva, lo cual facilita enormemente el diseño y la implementación de servicios web. Una vez terminado de definir el servicio web, puede generar automáticamente el código Java o C# libre de derechos de autor necesario para implementar el servicio en un servidor. La función de generación automática de código a partir del diseño garantiza que el código escrito sea coherente en todo el proyecto, porque el código se produce de acuerdo con estándares sectoriales, parámetros y opciones definidos globalmente.

La mayoría de las herramientas de Altova MissionKit pueden integrarse en Visual Studio y Eclipse y permiten editar archivos de configuración de Java EE[10].

Entre las herramientas Altova MissionKit se encuentran:

- XMLSpy: editor WSDL gráfico para diseñar, editar y validar archivos WSDL, así como su documentación, además constituye un cliente SOAP para generar, validar y probar sus mensajes.
- MapForce: permite la creación visual de servicios web. Genera código Java y C# y los scripts necesarios para implementación de los mismos a partir de archivos WSDL.

1.3.3.4 Gestión de servicios web de ASP.NET

ASP.NET es un framework para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores y diseñadores para construir sitios web dinámicos, aplicaciones web y servicios web XML (22).

Para implementar un servicio web usando ASP.NET se puede generar automáticamente el WSDL de manera predeterminada, pero también puede ser deshabilitada esa opción para controlar su generación. Una vez creado el WSDL se utiliza la herramienta

[10] Java EE: plataforma de programación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java.

(Wsdl.exe) que genera código para servicios web XML y clientes de servicios web XML de ASP.NET a partir de archivos de contrato WSDL.

Los documentos de descubrimiento de un servicio web XML se pueden obtener utilizando la herramienta (Disco.exe). Los archivos que crea esta herramienta se pueden usar como entrada para (Wsdl.exe) (23).

Cuando se utiliza Wsdl.exe para crear una clase de proxy, se crea un único archivo de código fuente en el lenguaje de programación especificado. Durante el proceso de generación del código fuente para la clase de proxy, la herramienta determina el tipo más adecuado para utilizarlo con los objetos especificados en la descripción de servicio. En algunos casos, la herramienta usa un enfoque de denominador menos común para convertir los objetos a un tipo determinado (23).

1.3.3.5 Componentes para la gestión de servicios web en la Plataforma Educativa Zera

La Plataforma Educativa Zera cuenta con componentes funcionales agrupados en forma de plugins, que fueron creados aplicando tecnologías y estándares de interoperabilidad. Entre sus principales funcionalidades se encuentra la gestión de los servicios web.
Estos componentes fueron desarrollados utilizando protocolos de seguridad para garantizar la autenticación, confidencialidad e integridad de los servicios web. Utilizan para el intercambio de datos varios protocolos tales como: SOAP, REST, XML-RPC, JSON-RPC, AMF 3, aprovechando las ventajas que cada uno de ellos brinda.

1.3.3.6 Conclusiones del análisis

En las soluciones analizadas anteriormente se detectaron un conjunto de deficiencias y características entre las cuales se puede señalar:

- El componente para la gestión de servicios web de Eclipse es una herramienta útil, pero no posee la opción de generar código en varios lenguajes de programación, está creado solo para Java. Permite desarrollar servicios web de las formas Top down y Bottom up.
- Apache Axis posee un entorno de desarrollo solo para servicios SOAP con su respectiva descripción WSDL, no permite ejecutar y administrar servicios basados en otros protocolos. Posibilita crear servicios de la forma Top down. Además es una herramienta orientada al lenguaje de programación Java.

- Las herramientas de Altova MissionKit para gestionar servicios web, además de ser privativas se basan solamente en SOAP con WSDL y generan código solo para Java y C#.
- El componente para la gestión de servicios web de ASP.NET es una herramienta privativa de Microsoft que promueve solamente el uso de SOAP y WSDL.
- Los componentes para la gestión de servicios web en la Plataforma Educativa Zera, tienen soporte para varios protocolos de comunicación y el código que genera está en PHP, pero está desarrollado en una tecnología sin soporte y no permite crear servicios de la forma Top down.

El aporte fundamental del estudio de las soluciones similares fue encontrar la existencia de dos formas para crear los servicios web, la manera Top down y la Bottom up, propuestas por el componente para la gestión de servicios web de Eclipse. Además los componentes para gestionar los servicios web de la Plataforma Educativa Zera proponen una visión en la que es posible gestionar varios protocolos en un mismo plugin. Algunas de las herramientas estudiadas permiten crear un servicio SOAP a partir del contrato, aspecto que se tendrá en cuenta para el desarrollo de la solución.

1.3.4 Diseño basado en contrato

El propósito fundamental de los servicios web es garantizar la interoperabilidad entre sistemas y para desarrollarlos existen dos formas:

- Bottom up (de lo detallado a lo general): consiste en exponer un componente o recurso de la aplicación existente como servicio web. Es decir, a partir de la clase con las funcionalidades ya implementadas, utilizando una herramienta, se genera automáticamente la descripción del servicio. Esta variante permite que el contrato dependa de la herramienta utilizada para crearlo, lo que implica, que al utilizar una tecnología diferente o una versión superior de la misma puedan ocurrir variaciones en el acuerdo y esto podría no ser muy apropiado para los clientes que están utilizando el servicio, pues tendrían que ajustarse al nuevo contrato. Este método no garantiza la interoperabilidad entre las aplicaciones en cualquier circunstancia, además va en contra de la lógica del mundo real, es decir, primero se realiza un acuerdo entre clientes y proveedores y no se debería violar ese acuerdo, lo apropiado es que la herramienta se adapte al contrato y no al revés. Se considera que la forma bottom up en un entorno real provoca problemas de interoperabilidad.

- Top down (de lo general a lo detallado): permite tomar la definición abstracta de un servicio web contenida en un contrato y generar una implementación concreta para el mismo. Es decir, de esta manera se crea la clase donde se implementarán las funcionalidades que serán expuestas como servicio a partir del acuerdo establecido entre clientes y proveedores. Esta forma garantiza que la herramienta utilizada para crear el servicio dependa del contrato generado y los clientes pueden consumir de estos independientemente de la tecnología con que hayan sido desarrollados. A la manera Top down se le conoce como diseño basado en contrato porque tiene semejanza con la vida real ya que se respeta el acuerdo por las partes implicadas, posibilitando la interoperabilidad entre las aplicaciones en cualquier circunstancia.

En esta investigación será utilizada la forma Top down para desarrollar servicios web, pues son creados a partir del contrato que lo describe y permite cumplir con el acuerdo establecido entre clientes y proveedores, garantizando de esta manera la interoperabilidad.

1.3.5 Herramientas para el desarrollo de servicios web

Con el objetivo de hallar herramientas para el trabajo con los servicios web en el lenguaje PHP y utilizarlas en la solución final, se realizó una investigación sobre las mismas. Seguidamente se muestra un resumen de los resultados obtenidos.

1.3.4.1 Zend Framework

Zend Framework es un framework de código abierto para desarrollar aplicaciones web y servicios web con PHP 5. Usa código orientado a objetos, ofrece una arquitectura Modelo-Vista-Controlador. Sus componentes están construidos con baja dependencia entre ellos, lo que permite a los desarrolladores utilizarlos por separado, pero cuando se juntan conforman un framework de aplicaciones (24).

Zend Framework es muy útil para desarrollar servicios web utilizando el protocolo de comunicación SOAP, pues proporciona una forma sencilla de generar el archivo WSDL a partir de la clase AutoDiscover. Para ello utiliza anotaciones PHPDoc, que permiten describir las funcionalidades. AutoDiscover se encarga de leer las anotaciones PHPDoc en una clase y a partir de ahí genera el WSDL. Además tiene una forma fácil de generar un servidor-cliente SOAP y permite crear servicios REST, XML-RPC, JSON-RPC y AMF3.

1.3.4.2 NuSOAP

NuSOAP es un kit de herramientas para desarrollar servicios web SOAP con el lenguaje PHP. Provee soporte para el desarrollo de clientes y de servidores. NuSOAP está basado en SOAP 1.1, WSDL 1.1 y HTTP 1.0/1.1 (25).

NuSOAP no necesita ninguna extensión adicional de PHP para ser utilizada, además permite crear servicios SOAP de manera sencilla, genera fácilmente el WSDL utilizando el método configureWSDL() y para registrar la función con sus parámetros de entrada y retorno el método register(). Además puede generar tipos de datos complejos y arreglos en el WSDL.

1.3.4.3 PHP-WSDL-2.3

PHP-WSDL-2.3 es un paquete que se puede utilizar para generar un WSDL a partir del código de una clase PHP con bloques de anotaciones o sin ellos. El WSDL creado es óptimo y puede ser modificado posteriormente. Genera la documentación necesaria en línea usando formato HTML, además brinda la posibilidad de descargarla en formato PDF. Trabaja con la clase SoapServer de PHP o cualquier tipo de servidor SOAP. Permite crear un cliente SOAP en lenguaje PHP e incluye código fuente totalmente documentado. Almacena en la memoria caché el WSDL para obtener un mejor rendimiento y brinda soporte para añadir tipos de datos simples, complejos y arreglos. Además permite levantar SOAP, XML-RPC, JSON, HTTP y servicios web REST con el mismo código fuente. Resulta altamente extensible (26).

1.3.4.4 Clase SoapServer de PHP

El lenguaje de programación PHP provee la clase SoapServer, la cual proporciona un servidor para los protocolos SOAP 1.1 y SOAP 1.2. Se puede usar con o sin lenguaje de descripción WSDL. Brinda un conjunto de funcionalidades que permiten: agregar funcionalidades al servicio, listar las funciones existentes, crear un servidor SOAP, notificar cuando han ocurrido errores, controlar peticiones SOAP, entre otras. Para su utilización se necesita tener instalada y habilitada la extensión de PHP para SOAP.

1.3.4.5 BeSimpleSoapBundle de Symfony2

Facilita la creación de servicios web SOAP y WSDL en las aplicaciones Symfony2, crea el WSDL de manera automática a partir de anotaciones. Los servicios se definen

directamente en un archivo YAML[11]. Requiere tener habilitada la extensión SOAP de PHP. La única diferencia significativa de este bundle con respecto a la clase SoapServer de PHP es que permite generar el WSDL de la manera “buttom-up” (27).

1.3.4.6 Guzzle, framework para desarrollar API RESTful

Este framework tiene soporte para prácticamente todos los tipos de solicitudes HTTP: GET, HEAD, POST, DELETE, PUT, PATCH, y OPTIONS. También permite manejar conexiones persistentes y tiene soporte para plantillas en lo que se refiere al esquema de direcciones URI. Al igual que la mayoría de los frameworks de este tipo tanto la autenticación como la autorización se realiza mediante el protocolo HTTP (28).

1.3.4.7 FOSRestBundle de Symfony2

Este bundle para Symfony2 provee varias herramientas para facilitar la creación de aplicaciones basadas en el protocolo de comunicación REST a partir de controladores definidos, entre ellas (29):

- Una nueva capa entre el controlador y la vista, que permite crear controladores compatibles con los diferentes formatos.
- Un cargador de rutas que se encarga de generar automáticamente las rutas que tradicionalmente utilizan las aplicaciones REST.
- Negociación del formato de la respuesta mediante la cabecera Accept, incluso para tipos MIME propios.
- Controlador especial para manejar correctamente las excepciones enviando el código de estado HTTP adecuado para cada error.

Herramientas seleccionadas

La clase SoapServer al ser una clase incluida en el lenguaje, permite crear servicios de manera sencilla. También muchas de las herramientas encontradas utilizan esta clase para crear sus servicios, como Zend Framework, PHP-WSDL-2.3 y BeSimpleSoapBundle. En la documentación oficial de Symfony2 emplean en sus controladores SoapServer para crear un servidor SOAP. Su utilización evita la dependencia de una librería externa pues

[11] YAML: Formato de serialización de datos legible por humanos donde la identación del archivo juega un papel fundamental, no se admiten tabulaciones.

puede no tener soporte, además PHP es un lenguaje que se encuentra en pleno desarrollo, lo que garantiza el soporte para esta extensión. Los servicios web SOAP gestionados por el componente se desarrollarán utilizando esta clase como proveedora del servicio, basada en el WSDL generado por la aplicación para cada servicio web.

FOSRestBundle es una solución probada para desarrollar APIs REST, este bundle es propuesto en tutoriales y documentación por las ventajas que brinda para crear API REST utilizando Symfony2, pues provee muchas funcionalidades útiles para el desarrollador de la API, además de gestionar los formatos a devolver que espera el cliente y de la creación de rutas de manera automática del tipo REST tradicional. También tiene soporte para todos los tipos de solicitudes HTTP. Para garantizar la accesibilidad por REST de los servicios web gestionados por el componente se utilizará este bundle.

1.4 Proceso de desarrollo del software

Un proceso de desarrollo de software define quien está haciendo qué, cuándo y cómo alcanzar un determinado objetivo. Es un proceso efectivo que proporciona normas para el desarrollo eficiente de un software con calidad, reduce riesgos y hace el proyecto más predecible (30).

A continuación se describen un conjunto de elementos fundamentales para guiar el desarrollo de esta investigación.

1.4.1 Metodología de desarrollo de software

Una metodología es un conjunto de procedimientos, técnicas, herramientas y documentos que posibilitan el desarrollo de un software (6).

Con el objetivo de seleccionar una metodología que permita guiar el proceso de desarrollo de software se analizaron un conjunto de características de Programación Extrema y Proceso Unificado de Desarrollo.

Programación Extrema (XP)

Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. No recomendada para proyectos complejos. Se basa en retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y decisión para enfrentar los cambios. Se

define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, donde existe un alto riesgo técnico (31).

Proceso Unificado de Desarrollo (RUP)

RUP es una metodología de desarrollo de software orientado a objeto que establece las bases, plantillas y ejemplos para todos los aspectos y fases de desarrollo del software, cada uno de estos artefactos se realiza dependiendo de las necesidades del desarrollador. RUP unifica al equipo de desarrollo de software, lo que favorece la comunicación, contribuye a la asignación de recursos en forma eficiente, y a la entrega de los artefactos correctos en el tiempo establecido. Permite el desarrollo de software a gran escala, mediante un proceso continuo de pruebas y retroalimentación, facilitando el cumplimiento de los objetivos propuestos con grandes resultados (30).

En la presente investigación se ha decidido utilizar como metodología Proceso Unificado de Desarrollo (en inglés Rational Unified Process) porque:

- Teniendo en cuenta la complejidad de la solución el equipo de trabajo considera que resulta necesario el desarrollo de un análisis profundo, generando artefactos que brinden los elementos necesarios para proceder al diseño y posteriormente a la implementación del componente de acuerdo con las necesidades del cliente.
- RUP ayuda a orientar el desarrollo de software, comenzando por la obtención de requisitos, que posteriormente son agrupados en casos de uso, a partir de los cuales se realiza el modelo de análisis, el diseño, la implementación y finalmente las pruebas.
- El uso de esta metodología permite realizar iteraciones completas a cada caso de uso, garantizando que se prioricen las funcionalidades más complejas.
- Permite seleccionar e implantar los componentes específicos del proceso dependiendo de las necesidades del proyecto.

1.4.2 Lenguaje de modelado

Un lenguaje de modelado constituye una guía estándar que provee un conjunto de diagramas, símbolos y mecanismos para diseñar sistemas informáticos (6).

Lenguaje Unificado de Modelado

Lenguaje Unificado de Modelado (UML) es un lenguaje gráfico de modelado de sistemas de software. Permite visualizar, especificar, construir y documentar un sistema. Se puede

utilizar para dar soporte a una metodología de desarrollo de software (tal como RUP), pero no especifica una en particular. No tiene propietario (32).

Para el desarrollo de la investigación este lenguaje será utilizado para generar:

- Diagrama de clases: tipo de diagrama que describe la estructura de un sistema a través de sus clases.
- Modelo de dominio: permite identificar los principales conceptos del sistema y las relaciones que se establecen entre ellos.
- Modelo de datos: describe la estructura de la información que gestiona el sistema.
- Diagrama de componentes: divide el sistema de software en componentes y representa las relaciones entre ellos.
- Diagrama de caso de uso: representa la relación que se establece entre los casos de uso y los actores del sistema.

1.4.3 Herramienta para el modelado

Las herramientas para el modelado son un conjunto de programas y ayudas que brindan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software (Investigación preliminar, Análisis, Diseño, Implementación e Instalación).

Existen varias herramientas para el modelado entre las que se encuentran Rational Rose, ArgoUML y Visual Paradigm. En el desarrollo de esta investigación será utilizada Visual Paradigm para crear los diagramas que serán elaborados en cada flujo de trabajo propuesto por la metodología.

Visual Paradigm v8.0

Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Puede ser utilizada en diferentes plataformas. Permite realizar ingeniería directa e inversa, y generar bases de datos, convirtiendo los diagramas de entidad relación en tablas.

1.5 Entornos de desarrollo

Los entornos de desarrollo son herramientas para crear software sobre ambientes amigables. A continuación, se hace énfasis en las principales herramientas y tecnologías que serán utilizadas en el transcurso de esta investigación, priorizándose las definidas como parte de la arquitectura del marco de trabajo Xalix.

1.5.1 Lenguajes de programación

Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos respectivamente (33). En el mundo del desarrollo web, existen dos tipos de lenguajes, los del lado del cliente y los del lado del servidor pues internet se basa en una arquitectura cliente-servidor. El navegador web o cliente hace una petición al servidor, una URL en el caso de una petición por el protocolo HTTP o HTTPs y el servidor procesa la información y devuelve una respuesta, en este caso una página web.

Lenguajes de programación del lado del cliente

Los lenguajes del lado del cliente son aquellos que basan su procesamiento en un cliente web, es decir, son los lenguajes que son interpretados por el navegador.

Lenguaje de Marcado de Hipertexto v5 (HTML5)

HTML5 es la quinta revisión de HTML, el lenguaje en el que está creada la web. Esta nueva versión pretende remplazar al actual (X)HTML, corrigiendo problemas con los que los desarrolladores web se encuentran, así como rediseñar el código actualizándolo a nuevas necesidades que demanda la web de hoy en día (34).

A diferencia de otras versiones de HTML, los cambios en HTML5 comienzan añadiendo semántica y accesibilidad implícitas, especificando cada detalle y borrando cualquier ambigüedad (34).

HTML5 ayuda a los desarrolladores a mejorar la estructura de las páginas a través de nuevas etiquetas como: “<header>” para las cabeceras, <article>, <section> y <aside> para ajustar el cuerpo, <footer> para el pie de página. Además permite validar la información insertada por el usuario sin necesidad de JavaScript, para ello incorpora nuevas opciones en los elementos de los formularios como: “required” si un elemento es obligatorio, “pattern” definiendo una expresión regular para validar un campo, también nuevos tipos de campos como “email”, “date”, “birthday”, entre otros.

Hojas de Estilo en Cascada v3 (CSS3)

Es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML. Es la mejor forma de separar los contenidos y su presentación.

La separación de los contenidos y su presentación presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados *"*documentos semánticos*"*). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes (35).

CSS3 permite a los desarrolladores mantener un control mucho más preciso sobre la apariencia de las páginas, a través de la incorporación de nuevos mecanismos para mostrar los elementos de las páginas, sin tener que recurrir a JavaScript. Para lograr esto incorpora los selectores por atributos, nuevas propiedades para los textos como: "text-shadow", "text-overflow", también para bordes define: "border-color", "border-image", "border-radius", entre otras.

JavaScript

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

Permite la verificación y procesamiento de los datos introducidos por el usuario antes de ser enviados al servidor y es soportado por los navegadores disponibles actualmente (36).

Este lenguaje ayuda a los desarrolladores a mejorar el dinamismo de las páginas elaboradas. En la creación de formularios dinámicos es muy útil, pues con HTML5 y CSS3 no es posible eliminar o crear elementos a partir de las interacciones del usuario. Es decir, JavaScript posibilita manipular completamente el DOM[12] de una página web.

Lenguaje de programación del lado del servidor

Los lenguajes del lado del servidor son reconocidos, ejecutados e interpretados por el propio servidor y luego son enviados al cliente en un formato comprensible para él. Se

[12] Modelo de Objetos del Documento (DOM): es una API para acceder, añadir y cambiar dinámicamente contenido estructurado en documentos.

pueden mencionar algunos como: Perl, Python, Ruby, .NET y PHP. Entre las tecnologías para el desarrollo definidas en el marco de trabajo Xalix se encuentra PHP, el cual será utilizado para la implementación de la solución.

PHP v5.4.13

PHP, es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. Tiene soporte para cualquiera de los principales sistemas operativos del mercado y para una gran variedad de base de datos dentro de las que se incluye PostgreSQL. Además de ser soportado por la mayoría de los servidores web de hoy en día, incluyendo Apache (37).

Entre sus características principales se incluyen la autogeneración, almacenamiento y edición de archivos, procesamiento de información en formularios, manipulación de archivos en varios formatos como XML y YAML, entre otros. Posee librerías que facilitan el trabajo con servicios web, además los integrantes del equipo de desarrollo poseen conocimientos del mismo, lo que permite un ágil desarrollo de la solución.

1.5.2 Frameworks para el desarrollo

Un framework o marco de trabajo es un producto que sirve como base para la programación avanzada de aplicaciones, que aporta una serie de funciones o códigos para realizar tareas habituales. Por decirlo de otra manera, los frameworks son librerías de código que contienen procesos o rutinas ya listos para usar. Los programadores utilizan los frameworks para no tener que desarrollar ellos mismos las tareas más básicas, puesto que en el propio framework ya hay implementaciones que están probadas, funcionan y no se necesitan volver a programar (38).

1.5.2.1 Frameworks del lado del cliente

JQuery v1.10.2

JQuery es un marco de trabajo rápido, poderoso y fácil de utilizar que permite a los desarrolladores y diseñadores web agregar elementos dinámicos e interactivos a sus sitios. Simplifica la manera de interactuar con los documentos HTML, así como la manipulación del árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con el servidor de manera asincrónica (38).

El uso de JQuery es de gran ayuda para los programadores que deseen realizar tareas avanzadas en sus aplicaciones web sin preocuparse del navegador que está haciendo la

visita. Este framework posee muchas funcionalidades para tratar con los elementos de las páginas, además de muchos selectores para poder obtener estos elementos. Además cuenta con un gran número de plugins que permiten extender sus funcionalidades como: prettify.js para colorear código mostrado en pantalla en un lenguaje determinado, scrollTo.js para desplazarse hacia una sección determinada de la página, entre otros.

Bootstrap v3.0

Bootstrap, es un framework para el desarrollo de front-end utilizando CSS y HTML, con una altísima compatibilidad con navegadores y mínimo peso. Se trata de un conjunto de recursos (estilos tipográficos, elementos de formulario, botones, tablas, barras de navegación, y más) que, con su utilización se puede lograr una impronta muy profesional economizando tiempo de diseño y maquetado (39).

Bootstrap facilita mucho el trabajo con el maquetado de la página, permite que la plantilla diseñada se pueda observar de la misma manera en todos los navegadores, además brinda la funcionalidad de que el diseño sea optimizado para dispositivos móviles. También posee una alta gama de elementos y código implementado para ser utilizados en las funciones básicas y avanzadas del desarrollo de una aplicación web.

1.5.2.1 Frameworks del lado del servidor

Symfony v2.3.7

Symfony 2 es un popular framework para desarrollar aplicaciones PHP, el cual supone un cambio radical tanto en arquitectura interna como en filosofía de trabajo respecto a sus versiones anteriores. Symfony 2 ha sido ideado para exprimir al límite todas las nuevas características de PHP 5.3 y por eso es uno de los frameworks PHP con mejor rendimiento (40).

Symfony es un completo framework diseñado para optimizar el desarrollo de las aplicaciones web basado en el patrón Modelo-Vista-Controlador. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web (40).

Symfony 2 está desarrollado completamente en PHP 5.3. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony 2 es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server. Se puede ejecutar tanto en plataformas (Unix, Linux, etc.) como en plataformas Windows. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en el proyecto.

1.5.3 Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado (IDE) es un entorno de programación que ha sido empaquetado como un programa de aplicación; es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes, facilita el trabajo de los desarrolladores, puede estar orientado a uno o varios lenguajes de programación y brinda facilidades como: resaltado de sintaxis y completamiento de código.

Net Beans v7.3.1

NetBeans es un reconocido entorno de desarrollo integrado de código abierto y de distribución gratuita apoyado por una amplia comunidad de desarrolladores. Entre sus funciones incluye resaltado de sintaxis para los lenguajes de programación PHP, HTML, JavaScript, CSS, entre otros. Brinda depuración y un potente completamiento de código. Integra la gestión de configuración y las herramientas de control de versiones, lo que contribuye a mejorar el trabajo en equipo.

Ofrece soporte para los frameworks Symfony y jQuery agilizando el desarrollo de la solución. La integración con Symfony posibilita la ejecución de tareas tales como crear módulos, limpiar caché, entre otras. Además, este IDE utiliza asistentes para crear servicios REST y generar código para invocar servicios web (tanto en REST como en SOAP).

1.5.4 Sistema Gestor de Base de Datos

PostgreSQL v9.1

PostgreSQL, es un Sistema Gestor de Base de Datos (SGBD) multiplataforma y altamente escalable. Permite acomodar enorme cantidad de datos y usuarios

concurrentes. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Presenta sofisticadas funciones de replicación, backup en línea y consultas de gran complejidad. Se adapta para trabajar a alta velocidad en escenarios con gran carga de trabajo, además, el equipo de desarrollo está más familiarizado con este sistema gestor de base de datos (41).

1.5.5 Servidor web

Un servidor web es un programa informático para procesar aplicaciones del lado del servidor realizando conexiones bidireccionales o unidireccionales y síncronas o asíncronas con el cliente generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web (42).

Apache v2.4.4

Apache es un servidor web multiplataforma de código abierto totalmente gratuito. Cuenta con un elaborado índice de directorios y un directorio de alias. Permite generar informes de errores HTTP y gestionar recursos para procesos hijos. Se caracteriza por ser configurable, además, es fácil de instalar y goza de gran popularidad y aceptación (42).

1.6 Conclusiones del capítulo

El estudio de los conceptos fundamentales sobre interoperabilidad y servicios web, además de la selección de la metodología y de las principales herramientas que guiarán el desarrollo del componente para la gestión de servicios web en el marco de trabajo Xalix, formaron las bases del marco teórico conceptual de la investigación. A través del estudio de las soluciones similares se llegaron a conclusiones más claras sobre cómo desarrollar los servicios web, determinándose así que existen dos maneras Bottom up y Top down, esta última será la implementada por la solución debido a que no presenta problemas de interoperabilidad. Se determinó a partir del estudio de los protocolos existentes que SOAP y REST son la tendencia de las grandes empresas para desarrollar sus servicios, además, son los utilizados por la línea AIA, por tanto la solución tendrá soporte para estos protocolos en su primera versión. RUP será la metodología que guiará el proceso de desarrollo de software apoyada en UML junto con Visual Paradigm para generar los artefactos seleccionados. Los lenguajes de programación y las herramientas seleccionadas fueron definidos por el marco de trabajo Xalix.

Capítulo II: Análisis y diseño del sistema

2.1 Introducción

En las fases de análisis y diseño se crean un conjunto de artefactos que sirven de entrada a la implementación del sistema. El análisis ayuda a obtener una comprensión más precisa de los requisitos funcionales y una descripción de los mismos. El diseño permite modelar el sistema de manera que soporte todos los requerimientos. En ambas fases se generan un conjunto de artefactos que permiten el cumplimiento de sus objetivos.

2.2 Modelo de dominio

Un modelo del dominio es una representación de las clases conceptuales del mundo real, no de componentes de software. Constituye el artefacto más importante que se crea durante el análisis orientado a objetos. Mediante la notación UML, se representa con un conjunto de diagramas de clases en los que no se define ninguna operación. En el mismo se muestran (objetos del dominio o clases conceptuales, asociaciones entre las clases conceptuales y atributos de las clases conceptuales) (43).

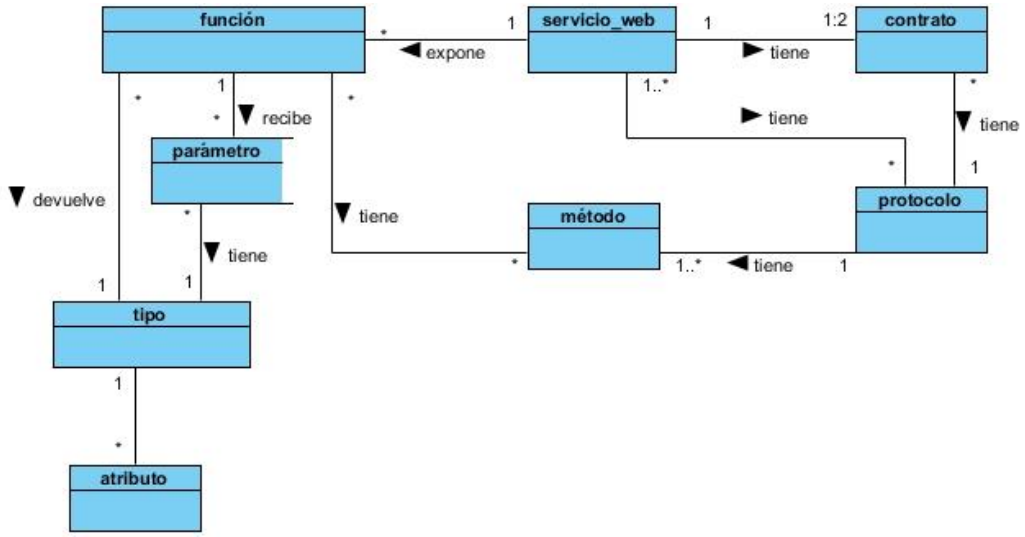


Figura 2.1 Diagrama de modelo de dominio.

A continuación se muestra una breve descripción de los conceptos asociados al modelo de dominio.

- **servicio_web**: tecnología para intercambiar información entre sistemas.
- **contrato**: acuerdo que se establece entre clientes y proveedores para crear y consumir un servicio web.

- **protocolo**: tecnología que permite que dos o más sistemas se comuniquen.
- **método**: método que posee un protocolo para garantizar la comunicación.
- **función**: procedimiento a ejecutar por un servicio web.
- **parámetro**: dato que necesita una determinada función para ser ejecutada.
- **tipo**: tipo de valor que puede tomar o devolver una variable o función en el sistema.
- **atributo**: característica que posee un tipo de dato específico.

2.3 Descripción del sistema propuesto

El sistema propuesto tiene como objetivo, crear un componente para la gestión de servicios web en el marco de trabajo Xalix, que permita la interoperabilidad entre los sistemas creados. A continuación se describen sus principales funcionalidades.

El sistema cuenta con dos roles, un usuario anónimo y un administrador. Esto implica que existan dos niveles de acceso.

El usuario anónimo tiene acceso al directorio de servicios web o UDDI, con el propósito de encontrar un servicio web disponible y utilizarlo. Para ello se muestran todos los datos relevantes del servicio como: la dirección del contrato asociado a este (pueden ser dos contratos, dependiendo de la disponibilidad del servicio), las funciones que provee además de los parámetros necesarios para su ejecución, los tipos de dato que maneja y retorna, entre otros. En la UDDI se publican servicios SOAP y REST a partir de sus respectivas descripciones WSDL y WADL.

El administrador puede acceder a la sección administrativa de la aplicación. En esta sección se proveen un grupo de funcionalidades como: la gestión de los servicios web mediante los protocolos SOAP y REST, la configuración de los protocolos existentes (pueden ser activados o desactivados), además de gestionar los tipos de datos y acceder a la UDDI.

Al crear o editar un servicio web se puede percibir el diseño basado en contrato, puesto que para realizar cualquiera de estas dos acciones después de introducir o modificar los datos, es necesario confirmar el contrato que se muestra en pantalla. De este contrato se espera que se consuma el servicio web por parte del cliente y se implemente basado en esas especificaciones, por parte del proveedor. El sistema, después de ser confirmado el WSDL o WADL mostrado, crea una clase en el lenguaje PHP con toda la información plasmada en el acuerdo lista para ser implementada. Si la acción es editar no se pierde el

código que pueda estar implementado, pues el sistema renombra la clase anterior a nombreAnterior.php[old].

También la aplicación posibilita realizar las tareas, activar o desactivar un protocolo o servicio web a partir de comandos, los cuales pueden ser útiles para el proveedor en caso de que no tenga la posibilidad o no desee por algún motivo utilizar la interfaz visual para estas acciones.

2.4 Requisitos de software

Un requisito de software constituye una restricción que se determina con alta precisión, y debe ser satisfecha por el software final. Existen dos tipos de requisitos de software, los funcionales y los no funcionales.

Los requisitos funcionales (RF) permiten expresar una especificación detallada de las responsabilidades del sistema que se propone, además determinan de una manera clara, el comportamiento del mismo.

Los requisitos no funcionales (RNF) permiten definir las cualidades que el software debe tener con el objetivo de brindar a los usuarios una mayor usabilidad, disponibilidad y rendimiento.

2.4.1 Requisitos funcionales

A continuación se describen los requisitos funcionales del sistema.

RF 1 Gestionar servicio web.

RF 1.1 Adicionar servicio web: este requisito permite crear un servicio web, para ello es necesario introducir los datos generales (nombre, estado, protocolo, descripción) y generar un contrato. A un servicio se le pueden añadir funciones pero no es obligatorio, aunque sí es recomendado pues existe con el objetivo de exponer sus funciones. Cuando se introducen los datos es necesario generar el contrato del servicio y si el cliente está de acuerdo entonces el servicio quedará creado. Cuando se adiciona un nuevo servicio se muestra un mensaje de confirmación.

RF 1.2 Mostrar servicio web: permite mostrar toda la información de un servicio web determinado.

RF 1.3 Editar servicio web: permite editar los datos de un servicio web determinado. Puede ser modificada la información general del servicio y sus funciones asociadas, luego

se debe generar nuevamente el contrato y si el cliente está de acuerdo guarda los cambios realizados aprobando el contrato.

RF 1.4 Eliminar servicio web: permite eliminar un servicio web del sistema. Al eliminar el servicio también se elimina el contrato y las funciones que contenga. Antes de eliminar el sistema solicita confirmación, y después muestra un mensaje que indica que la operación se realizó correctamente.

RF 2 Gestionar función.

RF 2.1 Adicionar función: permite adicionar una función a un servicio web. Para ello es necesario introducir los datos de la función a crear (nombre, descripción, método, retorno). Una función puede contener parámetros, estos deben tener un tipo de dato que puede ser creado, si no se encuentra entre los tipos de datos existentes en el sistema.

RF 2.2 Mostrar función: permite mostrar los valores de entrada, salida, retorno y la descripción de cada función en un servicio web determinado.

RF 2.3 Editar función: permite editar los datos (nombre, descripción, método, retorno) de una función en un servicio web, así como sus parámetros asociados y el tipo de dato de estos. Una vez editada la función, se hace necesario generar nuevamente el contrato del servicio, pues este cambia.

RF 2.4 Eliminar función: este requisito permite eliminar las funciones asociadas a un servicio web, lo que trae como consecuencia la necesidad de volver a generar el contrato del servicio, para realizar una actualización del mismo. Cuando se elimina una función también son eliminados sus parámetros, no siendo así con los tipos de datos ya que permanecen en el sistema para que puedan ser utilizados en otra ocasión.

RF 3 Gestionar parámetro.

RF 3.1 Adicionar parámetro: permite adicionar un parámetro a una función. Cada parámetro tiene un tipo de dato asociado que puede no existir en el sistema, en ese caso es necesario crearlo.

RF 3.2 Mostrar parámetro: cuando se muestra una función son mostrados también sus parámetros.

RF 3.3 Editar parámetro: cuando se edita una función pueden ser editados sus parámetros.

RF 3.4 Eliminar parámetro: permite eliminar un parámetro de una función.

RF 4 Generar contrato: permite crear un contrato para cada protocolo basado en los datos introducidos por el usuario para adicionar o editar un servicio web y teniendo en cuenta las funciones asociadas. Un servicio web no puede ser creado sin haberse generado antes el contrato. Si se realiza alguna modificación en los datos del servicio o sus funciones después de haber generado el contrato, se hace necesario repetir este proceso, pues estos cambios se reflejan en el contrato.

RF 5 Gestionar tipo de dato.

RF 5.1 Adicionar tipo de dato: permite adicionar un tipo de dato al sistema que posteriormente podrá ser utilizado en las funciones creadas. Estos tipos de datos contendrán al menos un atributo.

RF 5.2 Mostrar tipo de dato: permite mostrar la información de un tipo de dato seleccionado por el usuario.

RF 5.3 Editar tipo de dato: permite editar un tipo de dato que haya sido creado por el usuario. Si el tipo de dato a editar está asociado a un contrato, no es posible editarlo.

RF 5.4 Eliminar tipo de dato: permite eliminar los tipos de datos añadidos por el usuario y que no estén asociados a ningún contrato, pues si están siendo usados en el sistema, es necesario eliminar la entidad que los utilice primeramente, de lo contrario no podrá ser eliminado. Cuando se intenta eliminar un tipo de dato aparece un mensaje de confirmación.

RF 5.5 Listar tipo de dato: muestra un listado con todos los tipos de datos que contiene el sistema creados por defecto y otro con los tipos de datos creados por el usuario.

RF 6 Gestionar atributo.

RF 6.1 Adicionar atributo: permite adicionar atributos a un tipo de dato creado por el usuario. Para ello es necesario especificar el nombre del atributo y el tipo de dato del mismo.

RF 6.2 Mostrar atributo: permite mostrar los atributos de un tipo de dato adicionado por el usuario.

RF 6.3 Editar atributo: permite editar los atributos de un tipo de dato adicionado por el usuario.

RF 6.4 Eliminar atributo: permite eliminar los atributos de un tipo de dato adicionado por el usuario.

RF 7 Listar servicio web: se muestra un listado de todos los servicios web del sistema, así como los datos generales de cada uno (nombre, descripción, protocolo, estado). Posee las opciones editar, eliminar y adicionar un servicio.

RF 8 Consultar UDDI: muestra el listado con todos los servicios web publicados del sistema. Permite filtrar en el listado los servicios utilizando su nombre, descripción o el protocolo a través del cual fueron creados.

RF 8.1 Buscar servicio web: permite buscar un servicio web publicado, apoyándose en criterios de búsqueda.

RF 8.2 Consultar servicio web: permite mostrar toda la información de un servicio web publicado.

RF 11 Configurar protocolo.

RF 11.1 Ver datos de un protocolo: se muestran los datos de los protocolos.

RF 11.2 Activar protocolo: permite activar los protocolos de acceso a los servicios web disponibles en el sistema.

RF 11.3 Desactivar protocolo: permite desactivar los protocolos de acceso a los servicios web disponibles en el sistema.

RF 11.4 Mostrar cantidad de servicios web por protocolo: muestra la cantidad de servicios web que utiliza cada protocolo.

2.4.2 Requisitos no funcionales

- **Software:**

RNF 1 Tener cualquier navegador instalado que sea compatible con CSS3 y HTML5.

RNF 2 Tener un entorno de desarrollo web configurado para Symfony 2.3.7.

- **Hardware:**

RNF 3 El servidor deberá cubrir las siguientes características o contar con una variante equivalente a servidores web con procesador: Intel Core 2 Duo. Memoria RAM: 2 GB o más, con posibilidades de expansión en caso de ser necesario. Capacidad de almacenamiento: 160 GB.

- **Diseño e implementación:**

RNF 4 Framework de desarrollo: Symfony v2.3.7 LTS.

RNF 5 Lenguaje de programación para el servidor: PHP 5.4.13.

RNF 6 Lenguaje para el cliente: HTML5 (debe hacer uso de las nuevas etiquetas).

RNF 7 Gestor de base de datos: PostgreSQL v9.1.

RNF 8 Librería de CSS: Bootstrap v3.0.0 (compilado para el marco de referencia).

RNF 9 Librería de Javascript: jQuery v1.10.2 con jQuery UI v1.10.3.

- **Apariencia o interfaz externa:**

RNF 10 El diseño de interfaz debe ser sencillo, fácil de usar y con una navegación sugerente.

RNF 11 El sistema proporcionará una correcta organización de la información, permitiendo la interpretación correcta e inequívoca de ésta.

- **Usabilidad**:

RNF 12 El sistema podrá ser utilizado por cualquier persona que posea conocimientos básicos de informática y de programación.

RNF 13 El sistema debe permitir acceso al menú general desde cualquiera de sus páginas.

RNF 14 Se debe mantener informado al usuario del resultado de las acciones realizadas.

- **Rendimiento:**

RNF 15 Se debe garantizar que la respuesta a peticiones de los usuarios del sistema sea en un período de tiempo breve (de segundos), al igual que la velocidad de procesamiento de la información.

2.5 Modelo de casos de uso

Un modelo de casos de uso del sistema es una representación gráfica de procesos y su interacción con actores. Este modelo está formado por los casos de uso del sistema, los actores y las relaciones que entre ellos se establece. Un actor es un individuo o grupo que interactúa con el sistema, por otra parte, un caso de uso no es más que una secuencia de acciones que producen un valor observable para un actor concreto (30).

2.5.1 Descripción de los actores del sistema

Actor	Descripción
Administrador	Rol que posee privilegios para gestionar todas las funcionalidades del sistema. Encargado de gestionar los servicios, las funciones, los parámetros, los tipos de datos y de configurar los protocolos de comunicación.
Anónimo	El usuario anónimo solo puede acceder a la UDDI con el propósito de encontrar servicios web disponibles y consumir de ellos.

Tabla 2.1 Descripción de los actores del sistema.

2.5.2 Patrones y relaciones de casos de uso

Patrones de casos de uso

- CRUD Completo: se utiliza para administrar la información ya que unifica varios casos de uso simples en una unidad, permite modelar las diferentes operaciones, tales como adicionar, editar, mostrar y eliminar.

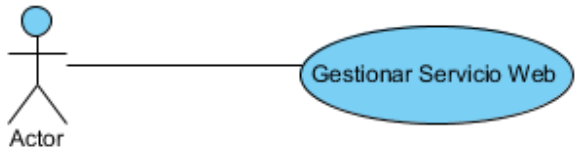


Figura 2.2 CRUD Gestionar servicio web.

- Reuso: cuando se determina que hay pasos iguales en dos o más casos de uso es recomendable utilizar este patrón, pues permite economizar, y ser más eficiente en el desarrollo.

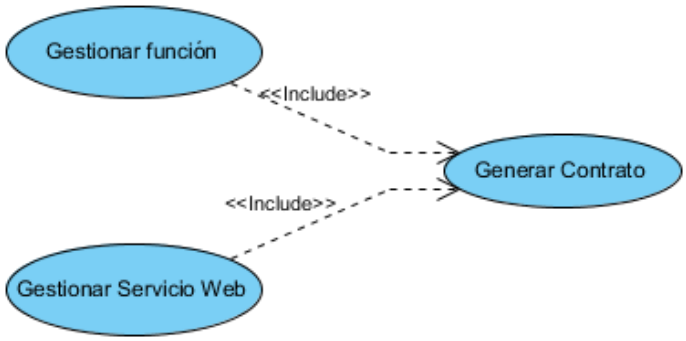


Figura 2.3 Reuso del caso de uso Generar Contrato.

Relaciones entre casos de uso

- Extensión: consiste en dos casos de uso y una relación extendida entre ellos. Esto significa que no resulta necesario que ocurra el caso de uso de extensión, aunque cuando ocurre ofrece un valor extra. Por ejemplo: una función puede contener o no parámetros por lo que ocurre una extensión entre los casos de uso Gestionar función y Gestionar parámetros.

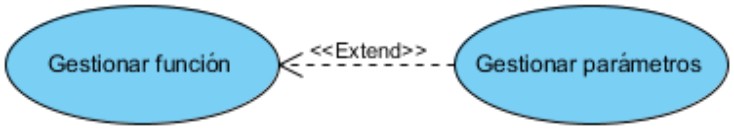


Figura 2.4 Extensión entre Gestionar función y Gestionar parámetros.

- Inclusión: Consiste en relacionar dos casos de uso con una fuerte dependencia de manera que uno no puede funcionar sin el otro. Por ejemplo: los tipos de datos que pueden ser creados en el sistema tienen que tener al menos un atributo, por lo que existe una inclusión entre ellos.

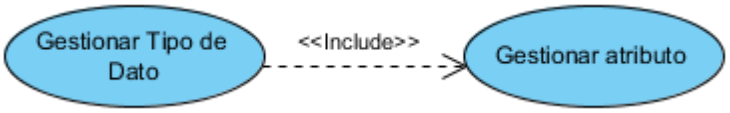


Figura 2.5 Inclusión entre Gestionar Tipo de Dato y Gestionar Atributo.

2.5.3 Diagrama de casos de uso del sistema

El diagrama de casos de uso del sistema especifica las funcionalidades y el comportamiento del sistema a través de la interacción con los actores, representados gráficamente. Permite que los clientes y usuarios validen que el sistema se convierta en lo que esperaban y los desarrolladores del sistema construyan lo que se espera (30).

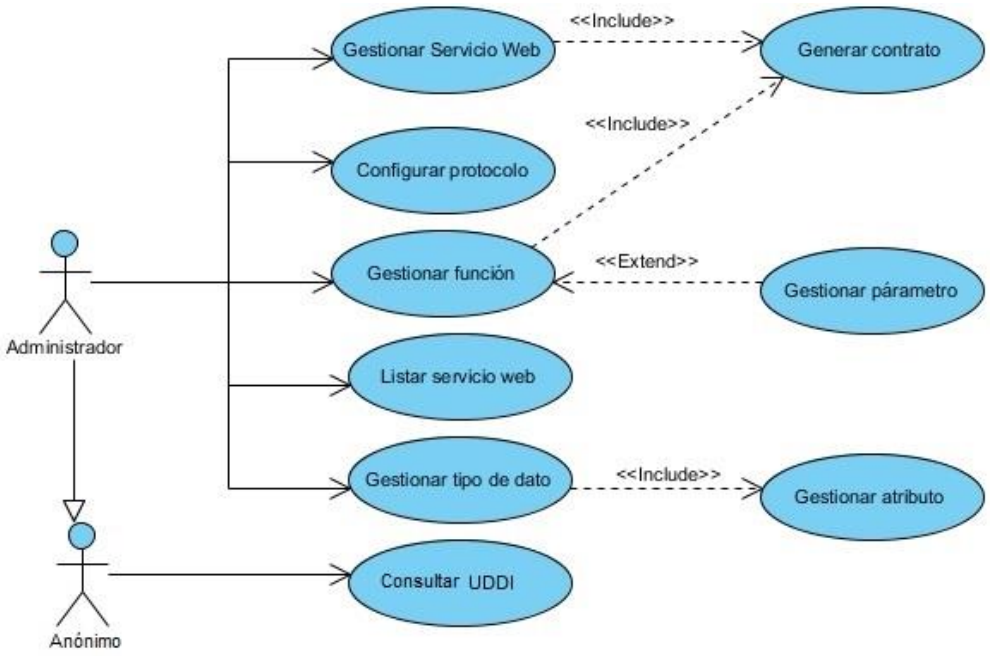


Figura 2.6 Diagrama de casos de uso del sistema.

2.5.4 Descripción de los casos de uso del sistema

A continuación se muestra la descripción del caso de uso Gestionar servicio web, para ver las restantes descripciones consultar el Anexo #1 Descripción de los casos de uso del sistema.

Objetivo	Permitir adicionar, editar, mostrar y eliminar servicios web.
Actores	Administrador.
Resumen	El CU inicia cuando el administrador selecciona la opción gestionar servicio web, se muestra un listado con todos los servicios web existentes en el sistema y permite adicionar, editar, mostrar y eliminar servicios. Si el actor decide crear un servicio, el sistema brinda la posibilidad de introducir los datos del servicio y generar un contrato, que puede ser cancelado por el actor si no está de acuerdo. En caso de que decida editar un servicio, debe tener previamente seleccionado el elemento a modificar, el sistema brindará la posibilidad de realizarle modificaciones a los campos existentes permitiéndole guardar los cambios realizados. Si decide mostrar el servicio lo selecciona y se muestra. Si selecciona eliminar

	un servicio, pide confirmación, brindando la posibilidad de eliminar o cancelar la opción. Termina el caso de uso.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El actor ha sido identificado.
Postcondiciones	El servicio ha sido creado, editado, modificado o eliminado correctamente.

Flujo de eventos		
Flujo básico <Gestionar servicio web>		
	Actor	Sistema
	1. Selecciona Gestionar Servicios Web.	
		2. Muestra un listado de todos los servicios web existentes en el sistema. Permite realizar las siguientes opciones: • Adicionar un servicio web. • Editar servicio web. Ver sección 1: Editar servicio web. • Mostrar servicio web. Ver sección 2: Mostrar servicio web. • Eliminar servicio web. Ver sección 3: Eliminar servicio web.
	3. Selecciona Adicionar Servicio Web.	
		4. Muestra un formulario para introducir o seleccionar los

		siguientes datos: • Nombre • Descripción • Estado • Protocolo y permite Generar contrato.
	5. Introduce los datos generales del servicio.	
	6. Selecciona Generar contrato.	
		7. Valida los datos.
		8. Muestra el contrato generado a partir de los datos introducidos anteriormente (Ver CU Generar contrato) y permite: • Aceptar. • Cancelar.
	9. Selecciona Aceptar.	
		10. Se crea el servicio.
		11. Muestra el servicio creado.
		12. Termina el caso de uso.
Flujos alternos		
7.a Introduce datos incompletos.		
	Actor	Sistema
		7.a.1. El sistema resalta en color rojo el campo obligatorio vacío y muestra el mensaje: "Rellene este campo".

		7.a.2. Regresa al paso 4 del flujo básico.
7.b Introduce datos incorrectos.		
	Actor	Sistema
		7.b.1. El sistema resalta en color rojo el dato y campo incorrecto y muestra el mensaje: "Ajústese al formato solicitado".
		7.b.2. Regresa al paso 4 del flujo básico.
9.a Selecciona Cancelar.		
	Actor	Sistema
		9.a.1. Regresa al paso 4 del flujo básico.
Sección 1		
Flujo básico < Editar servicio web **>**		
	Actor	Sistema
	1. Selecciona Editar Servicio Web.	
		2. Muestra el servicio permitiendo modificar los siguientes datos: • Nombre • Estado • Protocolo • Descripción y permite: Generar contrato.
	3. Modifica los datos.	
	4. Selecciona Generar contrato.	
		5. Valida los datos.

		6. Muestra el contrato generado utilizando los nuevos datos (Ver CU Generar contrato) y permite: • Aceptar. • Cancelar.
	7. Selecciona Aceptar.	
		8. Actualiza los datos del servicio.
		9. Muestra el servicio con las modificaciones que le fueron realizadas.
		10. Termina el caso de uso.
Flujos alternos		
3.a Introduce datos incompletos.		
	Actor	Sistema
		3.a.1. El sistema resalta en color rojo el campo obligatorio vacío y muestra el mensaje: "Rellene este campo".
		3.a.2. Regresa al paso 5 del flujo básico.
3.b Introduce datos incorrectos.		
	Actor	Sistema
		3.b.1. El sistema resalta en color rojo el dato y campo incorrecto y muestra el mensaje: "Ajústese al formato solicitado".
		3.b.2. Regresa al paso 5 del flujo básico.
6.a Selecciona Cancelar.		
	Actor	Sistema

		6.a.1. Regresa al paso 4 del flujo básico.
Sección 2		
Flujo básico < Mostrar servicio web **>**		
	Actor	Sistema
	1. Selecciona un servicio web para ser mostrado.	
		Muestra los datos del servicio seleccionado: • Nombre • Descripción • Protocolo • Estado • URI • Trasporte Listado con sus funciones y permite: Volver a la lista.
	2. Selecciona Volver a la lista.	
		3. Muestra un listado con todos los servicio web existentes en el sistema.
		4. Termina el caso de uso.
Flujos alternos		
2.a Muestra servicio web que no tiene funciones.		
	Actor	Sistema
		2.a.1 Muestra los datos del servicio seleccionado: • Nombre

		• Descripción • Protocolo • Estado • URI • Trasporte • El mensaje: ”Este Servicio Web no tiene funciones creadas” y permite: Volver a la lista.
	2.a.2. Selecciona Volver a la lista.	.
		2.a.3. Muestra un listado con todos los servicio web existentes en el sistema.
		2.a.4. Termina el caso de uso.
Sección 3		
Flujo básico < Eliminar servicio web **>**		
	Actor	Sistema
	1. Selecciona Eliminar Servicio Web.	
		2. Muestra el mensaje: ”¿Desea eliminar el Servicio Web <nombre>?” y permite: • Eliminar • Cancelar
	3. Selecciona Eliminar.	
		4. Elimina el servicio web.
		5. Muestra mensaje de confirmación: “El Servicio Web

		<nombre del servicio> ha sido eliminado correctamente".
		6. Termina el caso de uso.
Flujos alternos		
2.a Selecciona Cancelar.		
	Actor	Sistema
		2.a.1. Muestra un listado con todos los servicio web existentes en el sistema.
		2.a.2. Termina el caso de uso.
Relaciones	**CU incluidos**	Generar contrato.

Tabla 2.2 Descripción del caso de uso Gestionar servicio web.

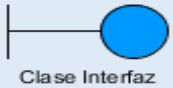
2.6 Modelo de análisis

El modelo de análisis ofrece una especificación más detallada de los requisitos funcionales, se describe utilizando el lenguaje de los desarrolladores y puede por tanto introducir mayor formalismo y ser utilizado para razonar sobre el funcionamiento interno del sistema. Permite estructurar los requisitos de modo que se facilita su comprensión, su preparación y su modificación. Este modelo puede considerarse como una primera aproximación del modelo de diseño (30).

En la presente investigación se crean los diagramas de clases de análisis y diagramas de colaboración correspondientes a cada caso de uso como parte del modelo de análisis.

Diagramas de clases de análisis

El diagrama de clases de análisis (DCA) representa una abstracción de los requisitos funcionales. Las operaciones y los atributos son poco detallados y siempre tienen tres estereotipos básicos: interfaz, control y entidad, los cuales serán descritos a continuación.

Estereotipos	Característica
	Modelan la interacción entre el sistema y sus actores.

	Modelan los aspectos dinámicos del sistema, coordinando las acciones y los flujos de control principales de los objetos que implementan la funcionalidad del CU.
	Modelan información que posee larga vida y que es a menudo persistente.

Tabla 2.3 Descripción de los estereotipos.

A continuación se muestra el DCA del caso de uso Gestionar servicio web, el resto de los diagramas pueden ser consultados en el Anexo # 2.

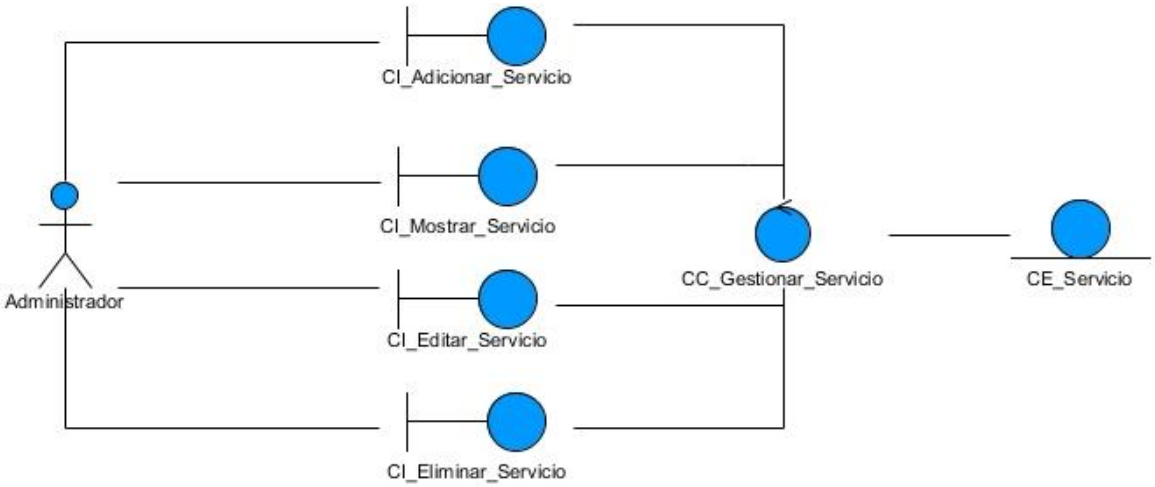


Figura 2.7 Diagrama de clase de análisis del CU Gestionar servicio web.

Diagramas de colaboración

Un diagrama de colaboración muestra las interacciones entre los objetos del sistema, a través de mensajes enumerados que indican un orden secuencial, permitiendo que se brinde una visión clara del flujo de control para cada caso de uso (30). A continuación se muestra el diagrama de colaboración perteneciente a la sección Adicionar servicio web correspondiente al caso de uso Gestionar servicio web.

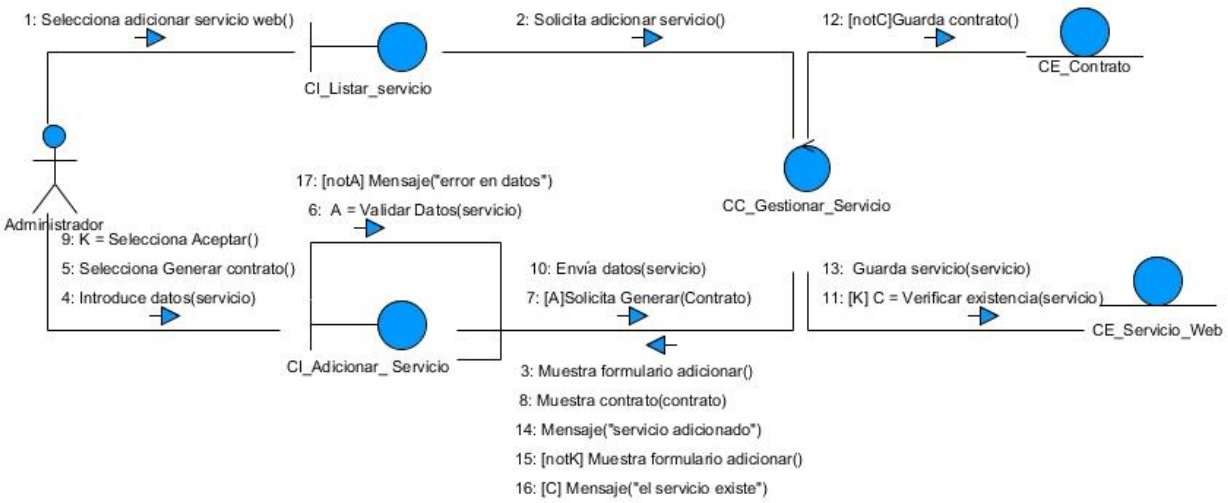


Figura 2.8 Diagrama de colaboración de la sección Adicionar servicio web.

2.7 Arquitectura del sistema

Symfony2 utiliza en su implementación una serie de patrones arquitectónicos que describen formas de solucionar problemas en el desarrollo de aplicaciones utilizando este framework. Symfony2 basa su funcionamiento interno en la arquitectura Modelo - Vista - Controlador (MVC). En lo adelante se explicarán algunos de los patrones utilizados directamente en la solución.

Patrón arquitectónico Modelo – Vista – Controlador

La arquitectura MVC permite dividir las aplicaciones en tres grandes capas, la capa del Modelo, la Vista y el Controlador. Symfony2 internamente lo emplea de la siguiente manera (40):

- El sistema de enrutamiento determina qué Controlador está asociado con la página solicitada.
- Symfony2 ejecuta el Controlador asociado a la página. Un controlador se encarga de procesar y mostrar los datos obtenidos en el modelo. Trabaja de intermediario entre la vista y el modelo, encargándose también de la lógica de negocio, es decir, no es más que una clase PHP en la que se puede ejecutar cualquier código deseado.
- El Controlador solicita al Modelo los datos necesarios. El modelo no es más que una clase PHP especializada en obtener información, normalmente de una base de datos (en este caso, el modelo está formado por las entidades de Doctrine).

- Con los datos devueltos por el Modelo, el Controlador solicita a la Vista que cree una página mediante una plantilla y que inserte los datos del Modelo.
- El Controlador entrega al servidor la página creada por la Vista.

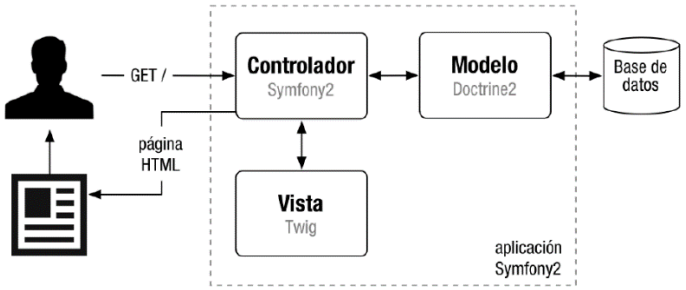


Figura 2.9 Arquitectura Modelo – Vista – Controlador (40)

Con Symfony2, el funcionamiento interno siempre es el mismo:

- El Modelo busca la información que se le pide.
- La Vista crea páginas con plantillas y datos.
- El Controlador manda y ordena.

En la solución, cuando un administrador selecciona listar servicios web, el routing determina que se debe ejecutar la acción indexAction() en la clase controladora WebServiceController. El controlador se encarga de solicitar al modelo, específicamente a la clase entidad WebService, el listado de todos los servicios. Con los datos devueltos por el modelo el controlador solicita a la plantilla index.html.twig que construya una página con los datos obtenidos, la cual será mostrada como respuesta a la solicitud del actor.

2.7.1 Patrones de diseño

Los patrones de diseño brindan solución a problemas comunes que pueden ser encontrados durante el diseño, perfeccionando los componentes de un sistema de software y sus relaciones. El framework Symfony2 para estructurar el diseño del sistema incluye un conjunto de patrones que formarán parte de la solución, como:

- **Experto:** es muy utilizado, mediante el cual se asignan responsabilidades a la clase que cuenta con la información necesaria. Se evidencia a partir de todas las clases entidades por ejemplo: WsFunction, WebService, Type y Contrate.

- **Creador:** permite crear objetos de una clase determinada. Es utilizado en la mayoría de las clases controladoras para crear instancias de formularios y entidades.
- **Controlador:** se basa en asignar la responsabilidad de todos los eventos realizados a una clase específica que constituye el único punto de entrada para cada evento. Este patrón se evidencia en las clases app_dev.php, app.php y las clases controladoras "Controller" de cada bundle.
- **Decorador**: el objetivo de este patrón es añadir dinámicamente funcionalidad a un objeto. El layout (base.html.twig) guarda el código HTML que se repite en todas las páginas del sistema y luego es utilizado para decorar cada plantilla. A continuación se muestra el uso de este patrón en la solución propuesta.

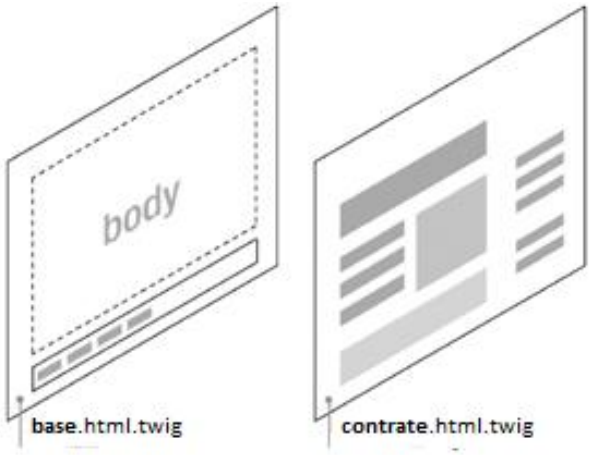


Figura 2.10 Patrón de diseño decorador.

- **Alta cohesión:** Symfony2 permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Las clases "Controller" están formadas por varias funcionalidades que están ampliamente relacionadas, siendo la misma responsable de definir las acciones para las plantillas y colaborar con otras para realizar diferentes operaciones, instanciar objetos y acceder a sus propiedades.

2.8 Modelo de datos

El modelo de datos se utiliza para describir la estructura lógica y física de la información persistente que puede ser gestionada por el sistema. A continuación se muestra el diseño de las tablas necesarias para el desarrollo de las funcionalidades que formarán parte de la base de datos, después de identificar las clases persistentes y realizar las

transformaciones correspondientes. Consultar el anexo # 5 Descripción del modelo de datos del sistema.

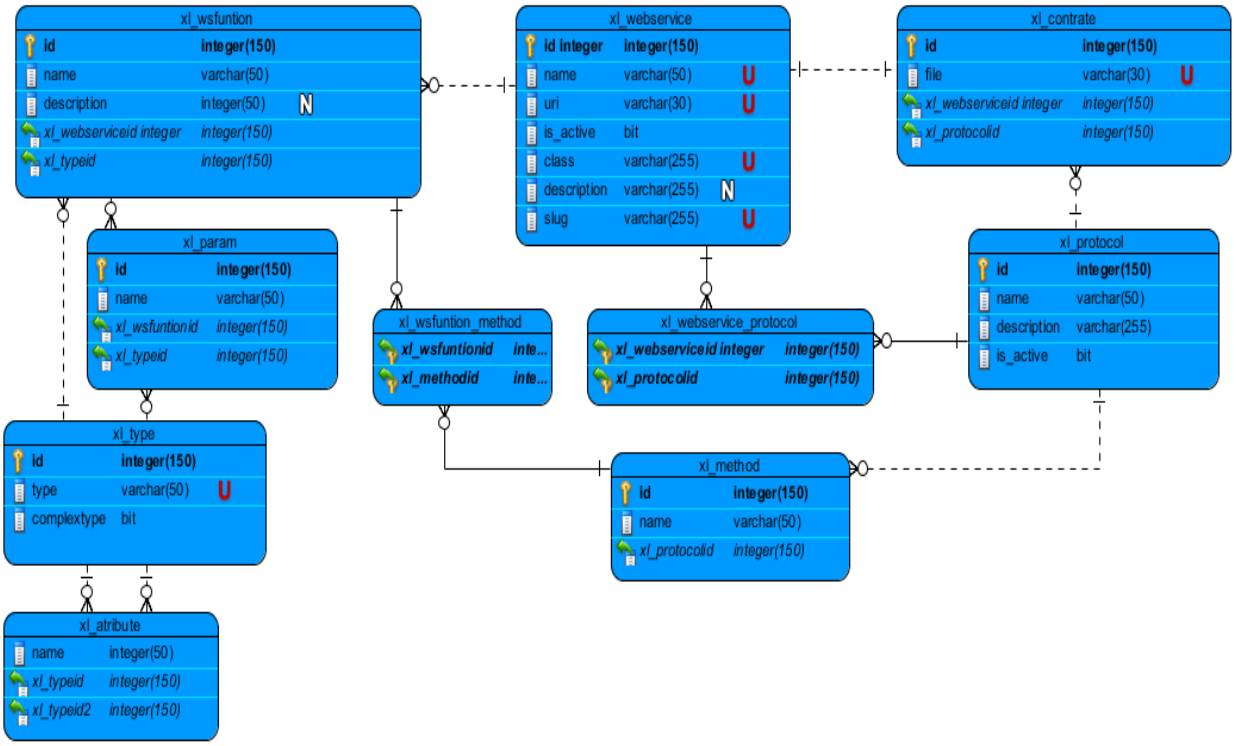


Figura 2.11 Modelo de datos del sistema.

2.9 Modelo de diseño

Un modelo de diseño es un modelo de objetos que representa las colaboraciones que ocurren entre las páginas, facilitando una abstracción de la implementación del sistema donde cada página lógica puede ser representada como una clase. Constituye el artefacto fundamental de entrada a la implementación.

Los propósitos del diseño son (30):

- Adquirir una comprensión en profundidad de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, sistemas operativos, componentes reutilizables, tecnologías de distribución y concurrencia, tecnologías de interfaz de usuario y tecnologías de gestión de transacciones.
- Crear una entrada apropiada y un punto de partida para actividades de implementación.

- Ser capaz de descomponer los trabajos de implementación en partes más manejables, que puedan ser llevadas a cabo por diferentes equipos de desarrollo, teniendo en cuenta la posible concurrencia.

2.9.1 Diagrama de clases del diseño

Es una representación visual del interior del módulo. Representa las páginas, sus enlaces y todo el contenido dinámico de crear las páginas del lado del servidor y el contenido dinámico en el lado del cliente. Además de la relación entre las capas del MVC con el componente de Symfony2. A continuación se muestra diagrama de clases de diseño (DCD) correspondiente al caso de uso Gestionar servicio web.

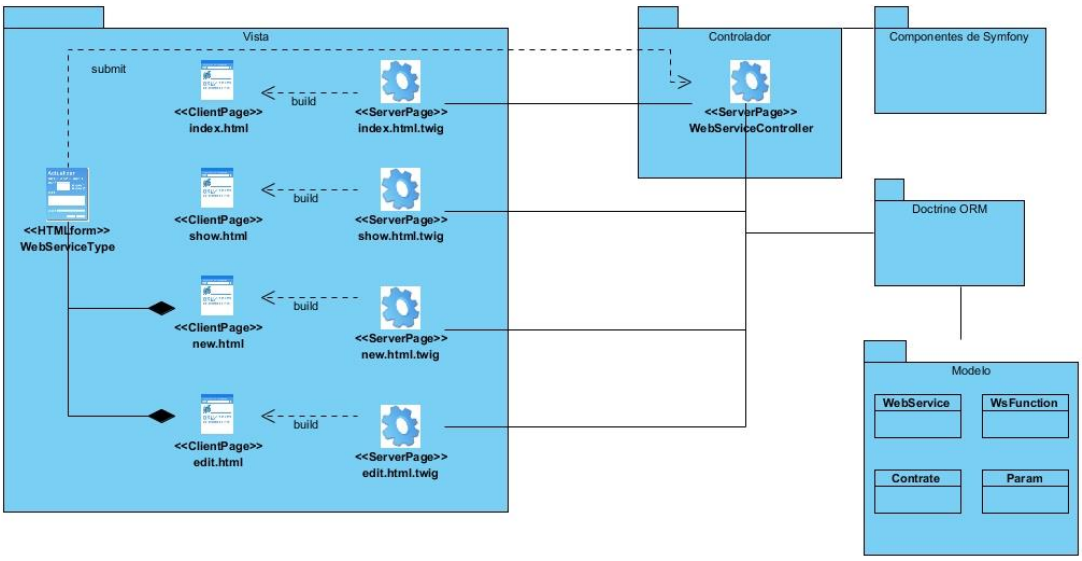


Figura 2.12 DCD del caso de uso Gestionar servicio web.

2.10 Conclusiones del capítulo

El análisis y diseño de la aplicación constituyen elementos fundamentales para dar paso a la implementación del sistema. El modelo de dominio permitió ofrecer una visión de los tipos más importantes de objetos que existen en el dominio del negocio y las relaciones que se establecen entre ellos y facilitó la identificación de los requisitos funcionales. Fue posible desarrollar iteraciones completas de las funcionalidades más complejas con ayuda del diagrama de casos de uso, el cual posibilitó agrupar y describir los principales procesos del sistema. La utilización de Symfony2 facilitó el uso de la arquitectura MVC y permitió eliminar problemas existentes en el diseño a través de los patrones que implementa. Concluidas las primeras fases de la metodología RUP se encuentran creadas las condiciones para proceder a implementar la solución propuesta.

Capítulo III: Implementación y prueba

3.1 Introducción

Durante el flujo de trabajo Análisis y Diseño son generados los artefactos que se utilizan como entrada fundamental al flujo Implementación. En este flujo se describen los principales elementos relacionados con la implementación del sistema, además de las pruebas que le fueron realizadas a la solución, para garantizar su calidad.

3.2 Modelo de implementación

El modelo de implementación describe cómo las clases y los elementos del modelo de diseño se implementan en términos de componentes. Describe también cómo estos se organizan de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados (30).

3.2.1 Estándares de codificación

Un estándar de codificación comprende todos los aspectos de la generación de código. Un código fuente completo debe quedar como si un único programador hubiera escrito todo el código de una sola vez. Al inicio de un desarrollo de software, es necesario establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. Para un programador comprender bien un sistema de software, influye directamente la legibilidad del código fuente. La mantenibilidad del código es la facilidad con que el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento (43).

Para el desarrollo del componente para la gestión de servicios web, Xalix propone a los desarrolladores que deben escribir el código estandarizado según las normas que se exigen en el documento sobre estándares de codificación. Para el caso de las tablas en las base de datos su nombre debe tener el prefijo xl_, por ejemplo si una tabla se llama webservice en la base de datos debería llamarse xl_webservice.

En resumen, el documento propone las siguientes normas para la estructura del código (43):

- Añadir un solo espacio después de cada delimitador coma.
- Añadir un solo espacio alrededor de los operadores (==, &&,…).

- Añadir una coma después de cada elemento del arreglo en un arreglo multilínea, incluso después del último.
- Añadir una línea en blanco antes de las declaraciones return, a menos que el valor devuelto solo sea dentro de un grupo de declaraciones (tal como una declaración if).
- Usar llaves para indicar la estructura del cuerpo de control, independientemente del número de declaraciones que contenga.
- Declarar las propiedades de clase antes que los métodos.
- Declarar los métodos públicos (public) primero, luego los protegidos (protected), y finalmente los privados (private).
- Utilizar paréntesis cuando se instancien clases, independientemente del número de argumentos que tenga el constructor.
- Utilizar mayúsculas intercaladas —sin guiones bajos— en nombres de variable, función, método o argumentos.
- Utilizar espacios de nombres para todas las clases.

Un ejemplo del uso de este estándar de codificación es la imagen que se muestra a continuación:

```
public function updateAction(Request $request, $id) {

    if ($editForm->isValid()) {
        foreach ($originalTags as $tag) {
            if (false === $entity->getAtribute()->contains($tag)) {
                $em->remove($tag);
            }
        }
        $em->flush();
        return $this->redirect($this->generateUrl('type_show', array('id' => $id)));
    }

    return $this->render('WebServiceBundle:Type:edit.html.twig', array(
                'entity' => $entity,
                'edit_form' => $editForm->createView(),
    ));
}
```

Figura 3.1 Ejemplo de código siguiendo el estándar definido por Xalix.

3.2.2 Diagrama de componentes

El diagrama de componentes (DC) se utiliza para modelar la implementación estática del sistema mostrando cómo está dividido en componentes y las dependencias que existen entre ellos.

A continuación se muestra el diagrama de componentes del caso de uso Gestionar servicio web y la descripción de los principales elementos que lo conforman.

- **Paquete Vista**: concentra los componentes pertenecientes a la vista del caso de uso.
- **Paquete Controlador**: agrupa los componentes que representan las acciones del caso de uso.
- **Paquete Modelo**: agrupa las clases para acceder a los datos de la base de datos.
- **Componente WebServiceType.php**: es el formulario del caso de uso encargado de estructurar cada elemento de la vista para editar o crear servicios.
- **Componente WebService**: contiene todas las plantillas con extensión html.twig relacionadas con el caso de uso.
- **Componente WebServiceController.php**: es el componente que contiene la clase controladora del caso de uso.
- **Componente Symfony**: contiene todos los elementos del framework que actúan o intervienen en la realización del caso de uso.
- **Componente WebService**.php: es la clase entidad que contiene todos los datos de los servicios.
- **Componente Contrate**.php: es la clase entidad que almacena los datos del contrato perteneciente a un servicio web.
- **Componente Protocol**.php: es la clase entidad que almacena los datos de los protocolos.
- **Componente WsFunction.php**: es la clase entidad donde se originan todas las relaciones que va a tener esa clase entidad con las asociadas a ella.
- **Componente Param.php**: es la clase entidad que contiene la información relacionada con los parámetros que reciben algunas funciones.
- **Componente Base de Datos**: encapsula todos los datos del sistema.
- **Componente css**: contiene los archivos que garantizan la correcta presentación de la información en las páginas web.
- **Componente jquery**: almacena los archivos con código javascript que aportan dinamismo necesario para las páginas web.

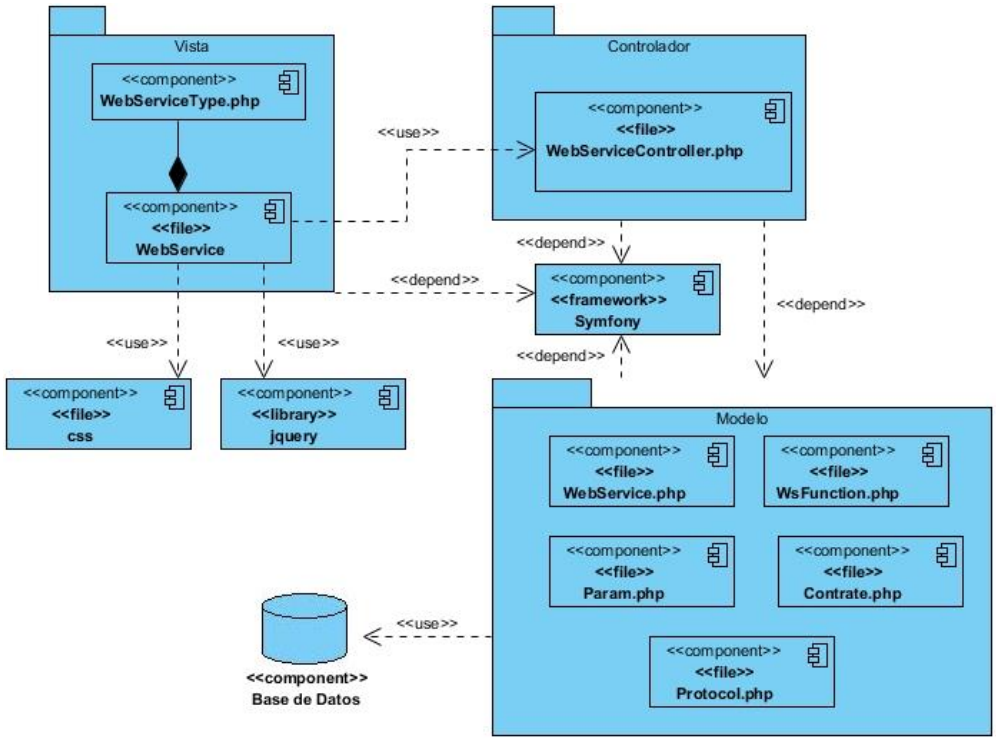


Figura 3.2 DC del caso de uso Gestionar servicio web.

3.3 Pruebas de software

Las pruebas se centran principalmente en la evaluación de un producto de software, para ello se buscan y se documentan los defectos en la calidad del software, lo que ayuda a validar que el producto funcione según lo diseñado.

Las pruebas de software se realizan teniendo en cuenta una serie de niveles que se mencionan a continuación:

- **Pruebas unitarias**: estas pruebas se ocupan de comprobar la lógica del procesamiento interno y las estructuras de datos. Son aplicadas a cada módulo de un software de manera independiente, con el objetivo de verificar que el módulo está correctamente codificado, de manera que permite comprobar las partes del software de manera independiente.
- **Pruebas de integración**: estas pruebas se aplican para descubrir errores asociados con la interfaz. El objetivo es tomar componentes a los que se le fueron aplicadas pruebas unitarias y evaluarlas en conjunto de una sola vez, permitiendo verificar que un gran conjunto de partes del software funcionen unidos.

- **Pruebas del sistema**: este tipo de pruebas estudia el producto completo para analizar defectos globales o para estudiar aspectos específicos de su comportamiento, tales como seguridad o rendimiento.
- **Pruebas de aceptación**: son como versiones de prueba con un conjunto de documentación aplicadas a clientes avanzados, el objetivo de estas pruebas no es modificar funcionalidades del sistema, si no detectar deficiencias que no hayan sido identificadas con pruebas anteriores.

En la medida en que las funcionalidades del componente fueron desarrolladas, su correcta implementación fue comprobada a través de pruebas unitarias. Estas fueron realizadas por los desarrolladores, no se registraron sus resultados ya que se llevaron a cabo durante todo el proceso de implementación. También fueron realizadas pruebas de sistema.

3.3.1 Métodos de prueba

Con el objetivo de comprobar la calidad del componente desarrollado, se realizó una revisión de las especificaciones del diseño y la codificación en correspondencia con los requisitos establecidos, para ello fueron aplicados los métodos de prueba Caja blanca y Caja negra.

Pruebas de Caja blanca: se realizan sobre las funciones internas de un módulo. Comprueban los caminos lógicos del software. Permiten examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado (44).

Pruebas de Caja negra: también denominadas pruebas de comportamiento, se llevan a cabo sobre la interfaz del software. Su objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto (44).

De las técnicas existentes para realizar pruebas de Caja negra se ha decidido utilizar Partición de equivalencia por su efectividad.

Una partición de equivalencia es una técnica de prueba de Caja negra que divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones

de entrada. Regularmente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica (44).

Para realizar las pruebas Unitarias se utilizó el método de prueba de Caja blanca y para las pruebas de sistema el método de Caja negra a través de la técnica Partición de equivalencia. A continuación se presenta el CP correspondiente al caso de uso Gestionar servicios web.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Gestionar servicio web.	Muestra un listado con todos los servicios web existentes en el sistema y permite: 1. Adicionar servicio web. 2. Editar servicio web. 3. Eliminar servicio web. 4. Mostrar servicio web.	Si selecciona la opción 1 ver sección Adicionar servicio web. Si selecciona la opción 2 ver sección Editar servicio web. Si selecciona la opción 3 ver sección Eliminar servicio web. Si selecciona la opción 4 ver sección Mostrar servicio web.	Inicio/Gestionar /Servicio Web.

Tabla 3.1 Sección Gestionar servicio web.

Escenario	Descripción	Nombre del servicio	Descripción	Protocolo	Estado	Respuesta del sistema	Flujo central
EC 1.1 Adicionar servicio web.	Muestra un formulario para introducir los siguientes	V	V	V	V	Muestra contrato.	Inicio/Gestionar Servicio Web/Adicionar
		V	NA	V	NA		

	datos: Nombre Descripción Protocolo Estado y permite Generar contrato.						Servicio Web.
EC 1.2 Introduce datos incorrectos.	Existen datos incorrectos.	I V	V I	V V	V V	El sistema resalta en color rojo el dato y el campo incorrecto, y muestra el mensaje: "Ajústese al formato solicitado".	Inicio/Gestionar Servicio Web/Adicionar Servicio Web.
EC 1.3 Campos obligatorios vacíos.	Existen campos obligatorios vacíos.	I V	V V	V I	V V	El sistema resalta en color rojo el campo obligatorio vacío y muestra el mensaje: "Rellene este campo".	Inicio/Gestionar Servicio Web/Adicionar Servicio Web.
EC 1.4 Aceptar el contrato.	Se acepta el contrato creado a partir de los datos del servicio.	NA	NA	NA	NA	Se guarda el servicio web y se muestra. Ver sección Mostrar servicio web.	Inicio/Gestionar Servicio Web/Adicionar Servicio Web/Generar

							Contrato/ Aceptar Contrato.
EC 1.6 Cancelar el contrato.	Se cancela el contrato creado a partir de los datos del servicio.	NA	NA	NA	NA	Muestra un formulario para introducir los siguientes datos: Nombre Descripción Protocolo Estado	Inicio/Gestionar Servicio Web/Adicionar Servicio Web/Generar Contrato/ Cancelar Contrato.

Tabla 3.2 Sección Adicionar servicio web.

Escenario	Descripción	Nombre del servicio	Descripción	Protocolo	Estado	Respuesta del sistema	Flujo central
EC 1.1 Editar servicio web.	Muestra un formulario para editar los siguientes datos del servicio seleccionado: Nombre Descripción Protocolo Estado y permite Generar contrato.	V	V	V	V	Muestra contrato.	Inicio/Gestionar Servicio Web/<nombre del servicio>/ Editar Servicio Web.
			.				
EC 1.3 Datos	Existen datos incorrectos.	I	V	V	V	El sistema resalta en	Inicio/Gestionar Servicio
		V	I	V	V		

incorrectos.						color rojo el dato y el campo incorrecto, y muestra el mensaje: Ajústese al formato solicitado".	Web/<nombre del servicio>/Editar Servicio Web.
EC 1.4 Campos obligatorios vacíos.	Existen campos obligatorios vacíos.	I V	V V	V I	V V	El sistema resalta en color rojo el campo obligatorio vacío y muestra el mensaje: "Rellene este campo".	Inicio/Gestionar Servicio Web/<nombre del servicio>/Editar Servicio Web.
EC 1.5 Acepta el contrato.	Se acepta el contrato creado a partir de los datos del servicio.	NA	NA	NA	NA	Se guarda el servicio web y se muestra. Ver sección Mostrar servicio web.	Inicio/Gestionar Servicio Web/Editar Servicio Web/Generar Contrato/Aceptar Contrato.
EC 1.6 Cancelar el contrato	Se cancela el contrato creado a partir de los datos del servicio.	NA	NA	NA	NA	Muestra un formulario para introducir los siguientes datos: Nombre	Inicio/Gestionar Servicio Web/Editar Servicio Web/Generar

						Descripción Protocolo Estado.	Contrato/Cancelar Contrato.

Tabla 3.3 Sección Editar servicio web.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Mostrar servicio web.	Muestra los datos del servicio seleccionado: Nombre Descripción Protocolo Estado URI Trasporte Un listado sus las funciones. y permite: Volver a la lista.	Muestra un listado con todos los servicios web existentes en el sistema.	Inicio/Gestionar Servicio Web/<nombre del servicio>.
EC 1.2 Mostrar servicio web sin funciones.	Muestra los datos del servicio seleccionado: Nombre Descripción Protocolo Estado URI Trasporte El mensaje: “Este Servicio Web no tiene funciones creadas ” y permite: Volver a la lista.	Muestra un listado con todos los servicios web existentes en el sistema.	Inicio/Gestionar Servicio Web/<nombre del servicio>.

Tabla 3.4 Sección Mostrar servicio web.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Eliminar servicio web.	Muestra el mensaje: "¿Desea eliminar el Servicio Web <nombre del servicio>?" y permite: Eliminar Cancelar	Muestra el mensaje.	Inicio/Gestionar Servicio Web/<nombre del servicio>/Eliminar Servicio.
EC 1.2 Eliminar.	Elimina el servicio web.	Elimina el servicio y muestra el mensaje: "El Servicio Web **<**nombre del servicio**>** ha sido eliminado correctamente".	Inicio/Gestionar Servicio Web/<nombre del servicio>/Eliminar Servicio/Eliminar.
EC 1.3 Cancelar.	Se detiene el proceso de eliminar un servicio web.	Muestra un listado con todos los servicios web existentes en el sistema.	Inicio/Gestionar Servicio Web/<nombre del servicio>/Eliminar Servicio/Cancelar.

Tabla 3.5 Sección Eliminar servicio web.

3.3.2 Resultados de las pruebas

Al componente desarrollado se le han realizado tres iteraciones de pruebas de Caja negra guiadas por casos de pruebas, las cuales arrojaron las siguientes no conformidades (NC) que fueron resueltas favoreciendo la calidad del componente.

Tipo de No Conformidad	Iteración 1	Iteración 2	Iteración 3
Errores de Interfaz	2	1	0
Correspondencia con otros artefactos	4	1	1

Validación	1	0	0
Ortografía	4	2	0
Opciones que no funcionan	1	0	0
Excepciones	1	0	0

Tabla 3.6 Resultado de las NC.

Tipo de No Conformidad	Iteración 1
Errores de Interfaz	La interfaz utilizada para adicionar tipos de dato tiene nombre Adicionar tipo. La interfaz utilizada para configurar los protocolos se llama ConfigurarProtocolo.
Correspondencia con otro artefactos	Faltas de correspondencia entre los casos de prueba y la aplicación como por ejemplo: Diferencias en el mensaje que se muestra cuando el usuario solicita eliminar un servicio web.
Validación	El inicio del nombre de una función no puede contener números.
Ortografía	Se encontraron palabras sin acentuar como por ejemplo: descripción y acción.
Opciones que no funcionan	Al solicitar crear un contrato cuando se han introducido datos con una longitud menor a la establecida el sistema no genera el contrato y no emite ninguna notificación.
Excepciones	Al solicitar editar un tipo de dato el sistema muestra un mensaje de error.

Tabla 3.7 Ejemplos de las NC encontradas.

3.4 Conclusiones del capítulo

En este capítulo se expusieron los elementos principales de la implementación del componente para la gestión de servicios web, el cual podrá ser reutilizado por todos los proyectos de la línea AIA. Al componente se le realizaron pruebas Unitarias y de sistema utilizando los métodos de Caja blanca y de Caja negra, los que permitieron encontrar y solucionar no conformidades, elevando de esta manera la calidad en el sistema.

Conclusiones generales

- El estudio del estado del arte sobre la gestión de servicios web indicó el punto de partida de la solución, definiendo protocolos, herramientas, metodología y tecnologías a utilizar.
- El análisis y diseño de la solución propuesta utilizando la metodología RUP generó documentación detallada para futuros cambios y centró al equipo de desarrollo en los requerimientos de esta.
- Las pruebas realizadas a las funcionalidades permitieron corregir un grupo de errores en un margen corto de tiempo, lo cual validó el correcto funcionamiento de la solución propuesta.
- El componente para la gestión de servicios web, con un diseño basado en contrato, garantiza la interoperabilidad entre sistemas creados dentro y fuera del marco de trabajo Xalix.

Recomendaciones

- Crear un componente visual gráfico para la gestión de los servicios web.
- Incluir un editor visual de XML para editar los contratos generados.
- Extender el uso del componente para la gestión de servicios en otras líneas de productos de software.
- Extender el componente para brindar soporte a otros protocolos.

Glosario de términos

A continuación se muestran en orden alfabético algunos términos usados en este trabajo:

ActionScript: es el lenguaje de programación de la plataforma Adobe Flash.

Caché: búfer especial de memoria que poseen los ordenadores.

Clase proxy: se encarga de ofrecer una interfaz igual a la del servicio web al que se intenta acceder, con el objetivo de redirigir las llamadas que se hagan a los métodos del verdadero servicio. Permite acceder a servicios web remotos y se encarga de intercambiar mensajes SOAP necesarios para la comunicación remota.

Componentes: objetos físicos que hay en tiempos de ejecución, de compilación o de desarrollo.

E-learning: procesos de enseñanza-aprendizaje que se llevan a cabo a través de internet, caracterizados por una separación física entre profesorado y estudiantes, pero con el predominio de una comunicación tanto síncrona como asíncrona, a través de la cual se lleva a cabo una interacción didáctica continuada. Además, el alumno pasa a ser el centro de la formación, al tener que autogestionar su aprendizaje, con ayuda de tutores y compañeros.

Cortafuegos: es una parte de un sistema o una red que está diseñada para bloquear el acceso no autorizado, permitiendo al mismo tiempo comunicaciones autorizadas.

Hipermedia: término con el que se designa al conjunto de métodos o procedimientos para escribir, diseñar o componer contenidos que integren soportes tales como: texto, imagen, video, audio, mapas y otros soportes de información emergentes, de tal modo que el resultado obtenido, además tenga la posibilidad de interactuar con los usuarios.

Líneas de Productos de Software: se refiere a un conjunto de sistemas de software que comparten características y que son desarrollados a partir de un conjunto común de bienes.

Llamadas a procedimientos remotos: permiten a los programas llamar procedimientos localizados en otras máquinas, o sea un proceso X en una máquina A, puede llamar un procedimiento localizado en una máquina B.

Sistemas para la gestión del aprendizaje: un sistema de gestión de aprendizaje es un software instalado en un servidor web que se emplea para administrar, distribuir y controlar las actividades de formación no presencial de una institución u organización.

Sistemas para la gestión de contenidos de aprendizaje: un sistema de gestión de contenidos que se utiliza para crear y manejar el contenido de una parte de un programa de educación.

Mensaje response de HTTP: respuesta HTTP que da el servidor una vez que ha recibido y procesado una solicitud.

Plugins: es aquella aplicación que, en un programa informático, proporciona una funcionalidad adicional o una nueva característica al software.

Proxy: un programa o dispositivo que realiza una acción en representación de otro.

Repositorio de recursos: son archivos donde se almacenan recursos digitales de manera que estos pueden ser accesibles a través de Internet.

Servlets: una clase en el lenguaje de programación Java, utilizada para ampliar las capacidades de un servidor.

Servidor web: es un programa informático que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales y/o unidireccionales, las cuales pueden ser sincrónicas o asincrónicas con el cliente, generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente.

Sistemas informáticos heterogéneos: sistemas informáticos desarrollados con diferentes lenguajes de programación y que contengan diferentes sistemas operativos.

XML Schema: lenguaje de esquema utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML.

Referencias bibliográficas

1. Definición de Framework ¿qué es Framework? [En linea]. [Citado 13 de enero de 2014]. Disponible en: http://www.alegsa.com.ar/Dic/framework.php
2. **WORLD WIDE WEB COSORTIUM**. Guía brebe de servicios web w3c. [En linea]. Disponible en: http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb
3. **ADUANET**. Web Services. [En linea]. Disponible en: http://www.aduanet.gob.pe/aduanas/operatividad/WebServices.html
4. **GEMA BUENO DE LA FUENTE**. Análisis de la interoperabilidad entre los sistemas de apoyo a la formación de tecminho. Universidade do Minho Departamento de Sistemas de Informaçao, 2008.
5. **VILLOTA, ÁNGELA MARÍA ECHEVERRI y MONSALVE GALEANO, ELIZABETH**. Análisis, diseño e implementación de un prototipo de repositorio de objetos virtuales de aprendizaje para la UCP. Pereira: Repositorio Institucional Biblioteca Universidad Católica, 2011.
6. **GISELA SÚ RODRÍGUEZ, Argel Jesús Carrasco Oliva**. Desarrollo de componentes para la interoperabilidad de la plataforma educativa Zera. Universidad de las Ciencias Informáticas, 2012.
7. **AUSTRALIAN GOVERNMENT INFORMATION MANAGEMENT OFFICE**. Interoperability Technical Framework for the Australian Government. [En linea]. Junio 2005. Disponible en: http://www.agimo.gov.au/publications/2005/04/agtifv2
8. **SCS -Sistemas Cliente/Servidor**. [En linea]. October 2008. Diponible en: http://ccia.ei.uvigo.es/docencia/SCS
9. **IGNACIO GARCÍA, MACARIO POLO y FRANCISCO RUIZ, MARIO PIATTINI.** Servicios web. Universidad de Castilla-La Mancha, España.
10. Llamadas a procedimientos remotos con XMLRPC. [En linea]. Disponible en: www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r91064.PPT
11. XML-RPC. What is xmlrpc? [En linea]. 2011. Disponible en: http://xmlrpc.com/
12. **WORLD WIDE WEB**. SOAP Version 1.2 W3C Recommendation. [En linea]. Disponible en: http://www.w3.org/TR/soap12-part1/.
13. **RAFAEL NAVARRO MARSET**. Modelado, Diseño e Implementación de Servicios Web 2006-07. ELP-DSIC-UPV, [2006-2007].
14. JSONRPC. JSONRPC Specifications. [En linea]. Disponible en: http://jsonrpc.org/wiki/specification.
15. **ADOBE SYSTEMS INC**. Action Message Format -- AMF 3. 2006.
16. **WORLD WIDE WEB**. Guía Breve de Tecnologías XML. [En linea]. Disponible en: http://www.w3c.es/Divulgacion/GuiasBreves/TecnologiasXML
17. **CIBERNETIA**. WSDL. [En linea]. Disponible en: http://www.cibernetia.com/manuales/servicios_web/4_wsdl.php
18. **FELIPE JOSÉ GALLEGO RIVERA**. Lenguajes de Especificación de Servicios Web Semánticos [En linea]. Departamento de Lenguajes y Sistemas Informáticos: Universidad de Sevilla. Disponible en: http://www.lsi.us.es/docs/doctorado/memorias/Mem_Inv_DEA_FelipeGallego.pdf
19. Definicion de UDDI- ¿qué es UDDI? [En linea]. [Citado 18 de febrero de 2014].

Disponible en: http://www.alegsa.com.ar/Dic/uddi.php
20. Servicios Web. [En linea]. Julio 2008. Disponible en: http://kalistog.wordpress.com/servicios-web/
21. **JAVIER CÁMARA**. Creando un servicio web a partir de su interfaz WSDL. [En línea]. Disponible en: http://www.adictosaltrabajo.com
22. **MICROSOFT**. Servicios web de ASP.NET. [En linea]. Disponible en: http://www.msdn.microsoft.com/es-es/library
23. Herramienta Lenguaje de descripción de servicios web (Wsdl.exe). [En linea]. [Citado 3 de marzo de 2014]. Disponible en: http://msdn.microsoft.com/es-es/library/7h3ystb6(v=vs.90).aspx
24. **VIKRAM VASWANI**. Implement SOAP services with the Zend Framework. [En linea]. Disponible en: http://www.ibm.com/developerworks
25. Comenzamos a utilizar NuSOAP. [En linea]. [Citado 5 de marzo de 2014]. Disponible en: http://www.desarrolloweb.com/articulos/1884.php
26. **GOOGLE PROJECT HOSTING**. PhpWsdl. En linea]. Disponible en: http://code.google.com/p/php-wsdl-creator/
27. **SYMFONY.ES**. BeSimpleSoapBundle. [En linea]. Disponible en: http://symfony.es/bundles/besimple/besimplesoapbundle
28. Guzzle: framework para construir clientes de servicios REST | YoSymfony. [En linea]. [Citado 30 de mayo 2014]. Disponible en: http://yosymfony.com/guzzle-framework-para-construir-clientes-de-servicios-rest/
29. FOSRestBundle (bundle de Symfony2). [En linea]. [Citado 30de mayo de 2014]. Disponible en: http://symfony.es/bundles/friendsofsymfony/fosrestbundle
30. **JACOBSON IVAR, RUMBAUGH, JAMES y BOOCH GRADDY**. El Proceso Unificado de Desarrollo de Software. Madrid, 2000.
31. Ingenieria De Software: Metodologia XP. [En linea]. [Citado 5 de marzo 2014]. Disponible en: http://pnfiingenieriadesoftwaregrupocuatro.blogspot.com/2012/07/bienvenidos-al-blog.html
32. **LAURA MARTHÉ HINOJOSA CASTILLO**. Ingeniería del Software. [En Linea]. Disponible en: http://www.monografias.com/trabajos34/ingenieria-software/ingenieria-software.shtml
33. **LENGUAJES DE PROGRAMACION** | El Mundo Informático. [En linea]. [Citado 2 de junio de 2014]. Disponible en: http://jorgesaavedra.wordpress.com/2007/05/05/lenguajes-de-programacion/
34. **ALEJANDRO CASTILLO CANTÓN**. Manual de HTML5 en español [En linea]. Disponible en: http://es.scribd.com/doc/48124241/Manual-HTML-5-por-Alejandro-Castillo/
35. **JAVIER EGUILUZ**. Introducción a CSS [En línea]. 2009. Disponible en: http://librosweb.es/css/
36. **JAVIER EGUILUZ**. Introducción a JavaScript [En linea]. 2009. Disponible en: http://librosweb.es/javascript/
37. **PHP.NET**. ¿Qué puede hacer PHP? [En linea]. Disponible en: http://php.net/manual/es/intro-whatcando.php
38. **MIGUEL ANGEL ALVAREZ**. Introducción a jQuery. [En linea]. 2009. [Citado 13 de

enero de 2014]. Disponible en: http://www.desarrolloweb.com/articulos/introduccion-jquery.html
39. Developers Corner: A Quick Look At Twitter Bootstrap v3. [En linea].Disponible en: http://morganlinton.com/developers-corner-a-quick-look-at-twitter-bootstrap-v3/
40. **JAVIER EGUILUZ**. Desarrollo web ágil con Symfony 2.3. 2013. 7 de noviembre de 2013.
41. PostgreSQL: Documentation: 9.1: PostgreSQL 9.1.13 Documentation. [En linea]. [Citado 22 de mayo de 2014]. Disponible en: http://www.postgresql.org/docs/9.1/static/index.html
42. **ABCDATOS**. Servidor Apache v2.4.7. [En linea]. Disponible en: http://www.abcdatos.com/webmasters/programa/servidor-apache.html
43. **Larman, Craig.** UML y Patrones. Introducción al análisis y diseño orientado a objetos. 1999.
44. **ERNESTO VLADIMIR PEREDA DÍAZ y YAISMEL MIRANDA PONS**. Pautas de Codificación Centro FORTES Versión 1.0.
45. **ROGER PRESSMAN**. Ingeniería de software. Un enfoque práctico. 2002.