Universidad de las Ciencias Informáticas

Facultad 10

Implementación de servicios para consulta, seguridad y notificación de cambios realizados, en UDDI.

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

Autores:

Yanisleidis Pérez Guerrero

Javier Piloto Rodríguez

Alexander Pérez Lemes

Tutor:

Ing. Manuel Alejandro Gil Martín

Junio de 2008

Ciudad de La Habana

"No hacen falta alas para hacer un sueño, basta con las manos, basta con el pecho, basta con las piernas y con el empeño."

Silvio Rodríguez

DECLARACIÓN DE AUTORÍA

Como únicos autores de este trabajo autorizamos a la Facultad 10 y la Dirección de la Infraestructura Productiva de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Dejando constancia de ello a los __ días del mes de junio del año 2008.

_____	_____	_____
Yanisleidis Pérez Guerrero	Javier Piloto Rodríguez	Alexander Pérez Lemes
(Autor)	(Autor)	(Autor)

_____

Ing. Manuel Alejandro Gil Martin.

(Tutor)

Agradecimientos

De Yanisleidis y Javier:
Estamos agradecidos de todas aquellas personas que de forma u otra han apoyado la realización de este trabajo. Muchas gracias en especial a nuestros familiares y amigos que han sabido apoyarnos siempre, en todos estos años de estudio.
Agradecemos a nuestro tutor Chony por todo el apoyo brindado y por habernos guiado en cada momento en que lo solicitamos.
A todos los maestros y profesores que han ayudado a nuestra formación desde los primeros instantes.
Muy agradecidos estamos de la ayuda ofrecida por compañeros del proyecto, principalmente por Ledián, Luis, Carlos y León.
A Susel, muchas gracias por estar presente y dispuesta siempre que la necesitamos.
Agradecemos a la profe Aleida por luchar tanto por todos nosotros.

De Alexander:
Difícil ha sido el camino recorrido durante tantos años de estudio y sacrificio para lograr ser alguien en la vida, y no es hasta el final del camino cuando te das cuenta de que todo lo que hiciste era construir tu futuro y hacer realidad el sueño de muchas personas importantes para ti.
Exactamente por eso, quiero dar mis más sinceros agradecimientos a todas aquellas vidas que desde que era niño y justo hasta hoy, me han apoyado en las distintas etapas de mi vida, han tenido fe en mí y me han ayudado a encontrar mi camino.
A mi mejor amigo Michel, por su apoyo incondicional y sus consejos correctos en los momentos luminosos y oscuros de la carrera.
A mis compañeros de tesis Yanisleidis y Javier, si no fuera por ustedes hubiera sido muy difícil hacer el trabajo desde cero tan tarde.
A Ledián, Blanco, Yaisy, Susel, Elizabeth, Aleida, Chony, Carlos, Luis Angel Santos por la ayuda ofrecida en todas las gestiones de la tesis.
A todos nuestros profesores que dieron lo mejor de sí para formarnos como mejores profesionales.
A mis buenos y entrañables amigos en los diferentes cursos de mi vida: del pre-universitario y la universidad.

De Yanisleidis:

A mis padres por tanto amor y cariño que me han entregado y sobre todo por mostrarme el camino a seguir, guiando cada uno de mis pasos con su entrega desmedida...debo a ustedes lo que soy.

A mi pequeña princesita que es mi tesoro y ha sido mi inspiración.

A mis abuelos que sin ellos no habría podido crecer.

A Javier que me ha dado su amor y compañía desde mi primer año de universidad y ha sabido estar conmigo en cada momento…y a su familia que también es mía.

A mis tíos Betty y Camilo, en fin, a toda mi familia y a todos aquellos que me apoyaron y creyeron en mí.

De Javier:

A mis padres por todo el sacrificio hecho en todos estos años de estudio.

A mi hermana y a Pedrito por apoyarme siempre, siempre, siempre.

A mis sobrinos malcriados (Pedro D. y Abner) por alegrarme la vida.

A abuelita por brindarme tanto amor y cariño.

A pipo por su gran ejemplo en la vida.

A tía Martha, Mandi, Padrino, Zule y Jorgito por estar siempre ahí, para lo que haga falta.

A Yanis por todo lo que ha luchado junto a mí, y a su familia por ayudarme tanto.

A todos mis amigos por estar presentes todos estos años, especialmente a los que vienen acompañándome desde otros tiempos, estoy desesperado por salir de ustedes.

De Alexander:

A la memoria de tío Corcho y a la de mi abuelo Casito, siempre los llevo en mi corazón.

A Mima y Papito por ser los mejores padres del mundo, por toda la ternura y el amor que me han dado, por ser mi fuente de inspiración, por existir.

A tía Deisy, la persona más humana que he conocido, sorprendente y absolutamente maravillosa.

A mis abuelos, que están muy ilusionados con verme graduado, por todo su afecto.

A personas excepcionales en mi vida como tío Pepe y Rafelito, su ejemplo fue siempre mi guía durante todos estos años.

A mi hermana Liener, por ser una de las personas que más quiero en el mundo.

A Rachel y Alejandro, dos niños encantadores que me quieren mucho y alegran mis días.

En la Universidad de las Ciencias Informáticas (UCI) existen una gran cantidad de proyectos productivos, de ellos, un porciento significativo se dedican al desarrollo de servicios Web. Como proyección futura la universidad se propone ampliar la disponibilidad de este tipo de servicios, convirtiéndose en una necesidad el garantizar la buena organización e interoperabilidad de los mismos. Por lo que se decide desarrollar un sistema Universal Description, Discovery and Integration (UDDI) para resolver dicha situación, ya que con este directorio de servicios Web se puede acceder a cualquier información relacionada con los mismos.

El objetivo del presente trabajo de diploma es implementar los servicios de búsqueda y seguridad del registro UDDI, permitiendo el descubrimiento de servicios Web en la UCI y proporcionando seguridad al sistema.

Para alcanzar los resultados deseados, fue necesario e importante, seguir reglamentaciones internacionales que ofrecen una guía sobre cómo se debe implementar el sistema. En el trabajo se exponen detalles del estudio realizado a la última versión de estas especificaciones.

Se espera integrar satisfactoriamente el subsistema implementado al registro UDDI, para posibilitar una mejor localización de servicios Web, así como establecer políticas de seguridad que le brinden confiabilidad al sistema. Además posee como premisa el cumplimiento de las libertades del software.

Índice

Índice de Figuras

Introducción

En la actualidad, el desarrollo tecnológico junto a la producción de software es de gran importancia para todas las actividades realizadas por la humanidad. Actualmente existen numerosos modelos de arquitectura de software que aportan mejoras en la creación y los cambios, en los procesos del negocio de forma ágil. La Arquitectura Orientada a Servicios (en inglés Service Oriented Architecture, SOA) es uno de los modelos más usados, pues brinda soporte a los requerimientos de software que utilizan servicios Web. El correcto uso de estos servicios Web provee mayor eficiencia en el empleo de las tecnologías, ya que son una colección de protocolos y estándares que permiten el intercambio de datos entre aplicaciones, en una red, sin importar en que lenguajes se desarrollaron o sobre que plataformas se ejecutan [1]. Una de las razones por las que SOA es muy utilizada es debido a que se puede construir sobre servicios Web estándares, los cuales gozan de gran aceptación industrial. Dentro de los principales servicios Web estándares que utiliza SOA, se encuentra Universal Description Discovery and Integration (UDDI), este permite conocer los servicios Web, favoreciendo el desarrollo de los mismos. Hace varios años surge la iniciativa UDDI como respuesta a interrogantes que surgían en las estructuras directivas de diversas entidades [2] como por ejemplo: ¿Cómo se descubren los servicios Web? ¿Cómo categorizar la información de forma coherente? ¿Cómo se puede garantizar la interoperabilidad del mecanismo de descubrimiento? ¿Cómo se puede interactuar en tiempo de ejecución con este mecanismo de descubrimiento cuando una aplicación depende de un servicio Web? Como respuesta a estas cuestiones, numerosas empresas unieron sus esfuerzos y desarrollaron una especificación basada en estándares abiertos y tecnologías no propietarias que permitían resolver varios de los retos anteriores.

UDDI cuenta con diversas Application Programming Interface (API) estas constituyen elementos fundamentales para llevar a cabo sus acciones pues permiten estandarizar el comportamiento y la comunicación entre clientes y registros UDDI. En cada API se implementan una serie de funciones que posibilitan el correcto trabajo de UDDI, brindando un servicio más completo que facilita el descubrimiento, compartición y reutilización de servicios Web.

En la Universidad de las Ciencias Informáticas (UCI) existen varios proyectos productivos dedicados al desarrollo de aplicaciones Web que brindan servicios a todas las personas dentro de la misma, cada día el

crecimiento de estos servicios va en ascenso. Para lograr la interoperabilidad, organización, publicación y búsqueda de los servicios Web que sean desarrollados, se hace necesario el desarrollo de una UDDI en la UCI, la cual debe cumplir con los estándares de la *Organization for the Advancement of Structured Information Standards* (OASIS), como institución encargada de definir las especificaciones a seguir. Esto muestra el siguiente **problema** a solucionar, ¿Cómo resolver el descubrimiento de servicios Web y la seguridad, en UDDI?

Este trabajo es antecedido por documentos que permiten conocer las principales características que deben tener la búsqueda y seguridad en UDDI. Documentos que muestran las especificaciones formales de los estándares de OASIS. Además de trabajos de diploma realizados en la UCI como: Estudio UDDI [3] y Diseño de un catálogo de servicios Web basado en las especificaciones y normas de la UDDI para la plataforma de informatización en la UCI [4]

El **objeto de estudio** del problema planteado se enmarca en: Aplicaciones UDDI.

Abarcando como **campo de acción**: Servicios de Búsqueda y Seguridad en UDDI.

El **objetivo general** para solucionar lo planteado anteriormente es: Implementar las APIs de Búsqueda y Seguridad para una UDDI, siguiendo las reglamentaciones definidas por los estándares de OASIS.

El objetivo general se ha desglosado en varios **objetivos específicos** para su mejor cumplimiento:

- Describir estándares de OASIS.
- Analizar el funcionamiento de las APIs en UDDI.
- Implementar la UDDI Inquiry API.
- Implementar la UDDI Security API.
- Analizar el funcionamiento de algoritmos complejos.
- Validar la solución implementada.

Para el buen desarrollo de los objetivos específicos se plantean las siguientes **tareas** a seguir:

- Revisión y análisis crítico de la bibliografía encontrada, que estén relacionadas con aplicaciones UDDI.
- Estudio profundo de la técnica de programación y lenguaje a utilizar para comprobar su capacidad de soportar el subsistema óptimamente.
- Determinación de las herramientas adecuadas para una mejor implementación.
- Implementación de las APIs de Búsqueda y Seguridad en UDDI.
- Análisis y validación de cada API implementada.

Como resultado del trabajo se espera poder brindar las APIs necesarias para que los Sistemas de Búsqueda y Seguridad en UDDI, funcionen correctamente, optimizando así el descubrimiento de los servicios Web en la universidad, y la seguridad en UDDI.

El cuerpo de nuestro trabajo consta de tres capítulos distribuidos del siguiente modo:

Capítulo 1. Fundamentación teórica: Se plasma un profundo análisis del estado del arte para este tipo de aplicaciones, como también las herramientas y tecnologías a usar en el desarrollo y la implementación de estas APIs, las distintas técnicas de programación en cuanto a las tendencias actuales, metodologías y otras funcionalidades relacionadas con dichas técnicas; seleccionando las mejores teniendo en cuenta las ventajas que aportan para el desarrollo del sistema.

Capítulo 2. Descripción y análisis de la solución propuesta: Se hace una valoración crítica del diseño utilizado para la implementación y se describe cómo se ha implementado el sistema, usando todos los elementos necesarios.

Capítulo 3. Validación de la solución propuesta: Se hace un profundo análisis, donde se valora y valida la solución implementada según la correspondencia de los objetivos planteados, con los resultados obtenidos.

Capítulo 1. Fundamentación Teórica

1.1 Introducción

En este capítulo se hace un análisis de varios temas relacionados con los servicios Web y la arquitectura de estos, tales como lenguajes de descripción de servicios Web y software servidores de los mismos. Se analizan los distintos paradigmas de programación existentes en la actualidad y su evolución, para así elegir correctamente cuál de ellos es el más adecuado para el desarrollo del subsistema. También se estudian algunas herramientas necesarias a utilizar en la implementación de las APIs.

1.2 Estado de las tecnologías orientadas a servicios a nivel internacional

En la actualidad, la tecnología avanza vertiginosamente y a cada caso de la vida se le da una vista técnica de desarrollo. Existen diversas variantes de tecnologías desarrolladas que definen los marcos de referencia sobre como accionar en la construcción de un software, cada una según sus propias técnicas y metodologías asociadas. SOA ha ayudado considerablemente al desarrollo de software, ya que esta se basa en el uso de servicios Web, los cuales facilitan disímiles actividades y tareas necesarias del ser humano. Las ventajas de estos servicios conllevan al incremento del uso de los mismos a gran escala, por lo que surge la necesidad de su descubrimiento y organización de su publicación. Esta fue la razón que motivó la creación de UDDI como un servicio Web estándar que permite la búsqueda, interoperabilidad, organización y publicación de dichos servicios. UDDI ha sido la solución perfecta para esta situación, pues no es un repositorio de servicios Web, lo cual sería imposible por el crecimiento acelerado de estos a diario, sino un registro para guardar información referente a los mismos, es el lugar a donde debemos acudir para conocer qué servicios ofrecen las organizaciones, de qué tipo son los mismos, cómo se pueden utilizar, dentro de otras funcionalidades y ventajas que aporta.

1.3 Estado de las tecnologías orientadas a servicios en la Universidad

En la actualidad la UCI, fomenta el desarrollo de proyectos productivos que crean servicios Web, ya sea para el uso propio de la misma o para distintas instituciones nacionales e internacionales que se benefician con sus productos. La aparición de nuevas necesidades e ideas creativas en este plano, da lugar al incremento y desarrollo de estos servicios. Ello traerá aparejado la necesidad de una mejor organización e interoperabilidad del mecanismo de descubrimiento de servicios Web, lo cual tributará al

desarrollo y optimización de estos servicios en la universidad y por consiguiente su aplicabilidad en el país.

1.4 Modelo de Arquitectura Orientada a Servicios (SOA)

En su definición, OASIS, se refiere a SOA como lo siguiente:

"Es un paradigma para organizar y utilizar capacidades distribuidas y bajo el control de diferentes propietarios y dominios. Provee una manera uniforme de ofrecer, descubrir, interactuar y usar dichas capacidades para producir los efectos deseados de manera consistente y medible." [5]

SOA es un concepto de arquitectura de software que da soporte a los requerimientos de software que utilizan servicios Web, formado por servicios de aplicación débilmente acoplados y altamente interoperables y diseñados para enlazar negocios y recursos computacionales.

Posibilita la creación y los cambios de los procesos de negocio de forma ágil empleando funcionalidades contenidas en infraestructuras expuestas bajo la forma de servicios Web mediante la composición de nuevos procesos. SOA se puede construir sobre estándares de servicios Web proporcionando mayor interoperabilidad además de ser posible implementarlo utilizando cualquier tecnología basada en servicios, pero en su mayoría, las definiciones de SOA identifican la utilización de servicios Web. A estos servicios están ligados SOAP, WSDL y UDDI y aunque por ser SOA orientado a servicios no necesariamente tiene que usarlos, pero se aconseja su utilización.

Define capas de software que se encuentran especificadas como:

- Aplicaciones básicas: Consistente en sistemas desarrollados bajo cualquier arquitectura o tecnología.
- Exposición de funcionalidades: En las cuales las funcionalidades de la capa aplicativas son expuestas en forma de servicios.
- Integración de servicios: Proporcionan el flujo de datos entre elementos de la capa aplicativa. Composición de procesos: Establece el proceso en términos del negocio y sus necesidades.
- Entrega: Donde los servicios son expandidos a los usuarios finales.

SOA se considera adoptable por factores importantes como son las interacciones entre recursos computacionales y el intento de reutilizar estos, puede también simplificar las interconexiones, la reutilización de los sistemas antiguos o *legacy systems* y las mejoras en los tiempos de realización de cambios en los procesos.

En la actualidad muchas empresas enfrentan sus sistemas con SOA, que facilita el desarrollo de servicios comerciales que se pueden integrar y reutilizar fácilmente, creando una infraestructura de tecnología informática considerablemente flexible y adaptable.

Ventajas de SOA:

- Minimiza el tiempo de desarrollo, ya que los servicios se pueden reutilizar fácilmente y pueden convertirse en nuevas aplicaciones compuestas.
- Reducen los costos de mantenimiento, los servicios reutilizables reducen el grado de complejidad interna de los servicios.
- Aumenta la calidad de los servicios, porque una mayor reutilización de servicios crea servicios de mejor calidad.
- Rebaja los costos de integración, los servicios estandarizados pueden trabajar en conjunto, permitiendo que aplicaciones distintas se conecten con rapidez y facilidad.
- Reduce el riesgo porque menos servicios reutilizables brindan mayor control, y reducen el riesgo general relacionado con el cumplimiento.

1.5 Servicios Web

Los servicios Web son una colección de protocolos y estándares que permiten el intercambio de datos entre aplicaciones en una red, sin importar en qué lenguajes se desarrollaron o sobre que plataformas se ejecutan. Estos servicios en sí, proporcionan mecanismos de comunicación predefinidos entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario. Para proporcionar interoperabilidad y extensibilidad entre estas aplicaciones, y que al mismo tiempo sea posible su combinación para realizar operaciones complejas, es necesaria una arquitectura de referencia estándar. Para esto existen estándares abiertos desarrollados por organismos encargados de definirlos, algunos ejemplos de estos estándares son:

- XML: Formato estándar para los datos a intercambiar.
- SOAP: Protocolos sobre los que se establece el intercambio.
- WSDL: Lenguaje de la interfaz pública para los servicios Web.
- UDDI: Protocolo para publicar la información de los servicios Web.

Los servicios deben de especificarse, para que una aplicación conozca de forma automática qué formato usar para comunicarse con un servicio. Para ello se usa principalmente WSDL, que permite especificar la dirección de un servicio y el interfaz que se usa para acceder a él, sea SOAP o HTML.

Cada día aumenta la tendencia y la necesidad de construir grandes y complejos proyectos mediante pequeños componentes, los servicios Web cuentan con una flexibilidad muy importante con relación a lo anterior, y es que pueden aportar gran independencia entre el servicio y la aplicación que lo usa.

Poseen muchas ventajas en el ambiente informático:

- Aprovechan los sistemas de seguridad firewall sin tener que cambiar las reglas de filtrado, ya que se apoyan en el protocolo HTTP.
- Posibilitan la fácil combinación de servicios y software de diferentes partes del mundo, para proveer servicios integrados.
- Aportan interoperabilidad entre aplicaciones de software.
- Hacen más fácil el acceso a sus contenidos y la comprensión de sus funcionamientos, ya que fomentan los estándares y protocolos basados en texto que lo posibilitan.
- Están orientados al desarrollo de aplicaciones distribuidas, en una arquitectura multinivel.

También presentan algunos inconvenientes.

- Si son comparados con otros modelos de computación distribuida, se puede decir que su rendimiento es bajo.
- No tienen un grado de desarrollo comparable con los estándares abiertos de computación distribuida como *Common Object Request Broker Architecture* (CORBA), en cuanto a las transacciones.
- Apoyándose en HTTP, pueden obviar medidas de seguridad basadas en firewall.

1.6 Universal Description, Discovery and Integration (UDDI)

Es un servicio Web estándar utilizado por SOA, define un método de publicación y descubrimiento de servicios Web, acciones fundamentales en el desarrollo de los mismos. En especificaciones descritas por OASIS se permite y utiliza el término UDDI para referirse tanto al protocolo de acceso al registro de servicios Web, como al propio registro. UDDI no actúa como un repositorio de servicios Web, que era lo que se quería inicialmente, sino como un registro de los mismos, por lo que no cuenta con los propios servicios sino con información y referencias a estos [6].

Su principal objetivo es definir un conjunto de servicios que describan y permitan el descubrimiento de proveedores de servicios, de los servicios Web que proporcionan y de la información técnica que permite el acceso a los mismos. Estos datos de proveedores y servicios también son conocidos como:

- Páginas Blancas: Sección para la descripción de datos de proveedores.
- Páginas Amarillas: Sección para la descripción de datos de los servicios.
- Páginas Verdes: Sección para la descripción de información técnica sobre los servicios Web [6].

Un registro UDDI está formado por nodos, cada uno de estos son como un conjunto de servicios Web que soportan, al menos, una de las APIs de nodos de UDDI. Un nodo permite el acceso y el manejo de toda la información del registro, y solo puede pertenecer a un registro.

UDDI es accedido por mensajes SOAP y da paso 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.

1.6.1 Estructura de Datos UDDI

La estructura de datos UDDI permite almacenar en el registro, de forma persistente, todos los elementos que se necesitan para su correcto funcionamiento. Partiendo de estos elementos se pueden implantar las funcionalidades que necesita el registro para la publicación y búsqueda de servicios Web, entre otras [7].

- **businessEntity:** Estructura encargada de almacenar la descripción de los proveedores de servicios Web.
- **businessService:** Estructura que almacena la descripción de los servicios Web ofrecidos por el proveedor descrito en el **businessEntity**, por lo que se encuentra contenida dentro de esta.
- **bindingTemplate:** Estructura que almacena la descripción de la información técnica de acceso a un servicio y las estructuras de tipo **tModel**.
- **tModel:** Estructura que almacena la descripción de modelos técnicos que representan conceptos como: el tipo de servicio Web, un protocolo usado por los servicios o un sistema de categorización.

1.7 Protocolo de Simple Acceso a Objetos (SOAP)

El Protocolo de Simple Acceso a Objetos (en inglés Simple Object Access Protocol, SOAP) es un protocolo basado en XML que se utiliza para el intercambio de información estructurada en servicios Web, es un derivado de un protocolo llamado XML-RPC creado en 1998 [8]. Es mucho más robusto y flexible

que los métodos get y post de HTTP, mediante los cuales también puede interactuar un servicio con otros. Es posible utilizarlo en una amplia variedad de sistemas modulares para facilitar la interoperabilidad entre componentes de software heterogéneos, debido a que posee un modelo de empaquetamiento modular y mecanismos para la codificación de los datos dentro de los módulos. Es una especificación para la invocación de métodos en servidores, servicios, componentes y objetos, y codifica la práctica existente de utilizar XML y HTTP como un mecanismo de invocación de métodos.

Mediante la información de un paquete SOAP se puede invocar un método. SOAP no define la forma de convocar un método. Además, permite el intercambio de información entre clientes y servidores de HTTP, sin importar las plataformas y aplicaciones existentes en estos. Esta información es codificada utilizando XML.

Un mensaje SOAP está compuesto por un sobre que contiene el cuerpo del mensaje y cualquier información de cabecera utilizada para describir el mensaje.

Ventajas de SOAP:

- No está asociado con ningún lenguaje.
- Puede transportarse utilizando cualquier protocolo capaz de trasmitir texto, ya que no es más que un documento XML.
- Aprovecha los estándares existentes en la industria.
- Permite la comunicación entre entornos diferentes.

1.8 Lenguaje de Descripción de Servicios Web (WSDL)

El Lenguaje de Descripción de Servicios Web (en inglés *Web Services Description Language,* WSDL) esta basado en XML y es extensible, permite la descripción de los servicios Web desplegados, la localización y ubicación de estos servicios en Internet y describe parámetros y métodos que soporta.

WSDL describe la interfaz pública a los servicios Web y además se refiere a los servicios de red como colecciones de puntos finales de comunicación capaces de intercambiar mensajes y proporcionan documentación para sistemas distribuidos y sirven como fórmula para automatizar los detalles que toman parte en la comunicación entre aplicaciones. Dentro de las aplicaciones de WSDL tenemos el empleo por parte de la UDDI para la descripción de las interfaces de los servicios Web.

Los puertos definen un punto de conexión al servicio Web, así como las operaciones posibles mediante dicho servicio y los mensajes vinculados, además de cumplir una función similar a la de una función de

biblioteca en programación o a la de una clase en programación orientada a objetos. Los mensajes definen los datos que participan en una operación. Los enlaces o vínculos permiten la definición de los formatos de mensaje y de protocolo para los servicios Web.

Un fichero WSDL contiene una descripción de todo lo que implica una llamada a un servicio Web SOAP [9]:

- La URL y el espacio de nombres del servicio
- El tipo de servicio Web
- La lista de funciones disponibles
- Los argumentos de cada función
- El tipo de dato de cada argumento
- Los valores de retorno de cada función, y el tipo de dato de cada uno

En WSDL el elemento *message* define los datos que participan en una operación. Cada mensaje puede tener una o varias partes, y cada parte puede considerarse como si fuera los parámetros que se pasan en la llamada a una función en programación clásica o un método en programación orientada a objetos.

WSDL define cuatro tipos de operaciones [10]:

- Request-response (*petición-respuesta*): Comunicación en la que el cliente realiza una petición y el servidor envía la correspondiente respuesta.
- One-way (*un-sentido*): Comunicación del estilo documento en la que el cliente envía un mensaje pero no recibe una respuesta del servidor indicando el resultado del mensaje procesado.
- Solicit-response (*solicitud-respuesta*): El servidor envía una petición y el cliente le envía de vuelta una respuesta.
- Notification (*Notificación*): El servidor envía una comunicación del estilo documento al cliente.

1.9 Paradigmas de programación

La informática alcanza un auge significativo y dentro de esta, los lenguajes de programación, siendo sumamente importantes en el desarrollo del software. Estos lenguajes han evolucionado rápidamente debido a la creación de nuevos paradigmas de programación que de cierta forma han acotado etapas evolutivas.

Estos modelos de programación no son más que una colección de patrones conceptuales que juntos modelan el proceso de diseño y, finalmente determinan la estructura del programa.

Existen variedades de criterios en cuanto a la organización de estos paradigmas, resaltando las clasificaciones establecidas por Ambler [11].

Existen tres categorías fundamentales su clasificación:

- Los que soportan técnicas de programación de bajo nivel, o sea los que su programación se acerca al funcionamiento de una computadora.
- Los que soportan métodos de diseño de algoritmos, llamados también de medio nivel.
- Los que soportan soluciones de programación de alto nivel, los cuales son más fáciles de aprender porque están formados por elementos de lenguajes naturales.

El último de los mencionados anteriormente es el que será analizado posteriormente debido a que es el tipo programación en la que estamos inmersos. Los paradigmas de programación de alto nivel se dividen en tres categorías de acuerdo con la solución que aportan para resolver un problema y a la vez estas se dividen en subcategorías.

1.9.1 Paradigmas de programación operacional o procedimental

Estos tal vez sean los más conocidos y utilizados en el proceso de programación, especifican la programación como un conjunto de secuencias computacionales que se ejecutan paso a paso. Algunos lenguajes de programación apoyan a estos paradigmas proporcionando recursos para pasar argumentos a las funciones y devolviendo valores de las mismas. Se hace muy útil utilizar este tipo de programación al desarrollar grandes proyectos, ya que se crea una inmensa biblioteca de funciones especiales para procedimientos utilizados con frecuencia dentro del programa. Estos paradigmas se dividen en varias categorías como: paradigma imperativo, paradigma orientado a aspectos y paradigma orientado a objetos.

1.9.1.1 Paradigma imperativo

La programación imperativa describe la programación en términos del estado del programa y sentencias que cambian dicho estado. Los programas imperativos son un conjunto de instrucciones que le indican al computador cómo realizar una tarea.

Existen varios lenguajes que utilizan este paradigma, como por ejemplo: FORTRAN, COBOL, BASIC, C, Ada, Pascal.

Los lenguajes imperativos evolucionaron de la siguiente manera:

- Programación no estructurada: El programa principal opera directamente sobre datos globales.

- Programación procedimental: Permite combinar las secuencias de instrucciones repetibles en un solo lugar.
- Programación estructurada: Permite escribir un programa de forma clara, donde se utiliza únicamente tres estructuras: secuencial, selectiva e iterativa, siendo innecesario y no permitido utilizar instrucciones de transferencia incondicional.
- Programación modular: Los procedimientos con una funcionalidad común son agrupados en módulos, por lo que un programa cuenta con varias secciones en la que cada una desempeña una tarea necesaria para el correcto funcionamiento del programa global.
- Tipos abstractos de datos: Son usados para almacenar información y cada uno de ellos se caracteriza por su estructura y por sus métodos de acceso.

1.9.1.2 Paradigma orientado a aspectos

La programación orientada a aspectos trata de solucionar el problema de separación de asuntos en el desarrollo de software, el cual fue descubierto en la década de los años 1970. El principio de separación de asuntos plantea que un problema determinado, involucra varios asuntos, los cuales deben ser identificados y separados. Uno de los asuntos fundamentales que se presenta y que se debe resolver es la determinación de la función específica que debe realizar una aplicación, aunque pueden existir otros como, distribución, persistencia, replicación, sincronización, etc. Si se separan estos asuntos se puede disminuir la complejidad a la hora de ser tratados y a la vez se puede cumplir con los requerimientos de calidad deseados.

El principal objetivo de este paradigma es permitir una adecuada modularización de las aplicaciones y posibilitar una mejor separación de conceptos.

1.9.1.3 Paradigma orientado a objetos

El paradigma orientado a objetos evoluciona naturalmente de los lenguajes que fomentan el paradigma imperativo, o sea, la evolución del tipo abstracto al objeto. Este consiste en definir cuáles son los objetos adecuados y las interacciones entre ellos para diseñar aplicaciones y programas de computadora, dándole solución a un problema presentado. Los objetos son entidades que combinan estado, comportamiento e identidad. Esta programación expresa que un programa es un conjunto de objetos que se ayudan

mutuamente para realizar una tarea, lo que permite hacer más fácil de escribir, mantener y reutilizar dicho programa.
Se puede definir o identificar un objeto mediante la información que contiene y la comunicación entre varios objetos se puede hacer a través de mecanismos de interacción llamados métodos.
Existe una gran lista de lenguajes de programación que utilizan este modelo, pero los más usados en el mundo actualmente son: C++, C#, Delphi, Java, Perl, PHP, Phyton, entre otros.

1.9.2 Paradigmas de programación declarativos

Estos paradigmas se basan en hechos, reglas, restricciones, ecuaciones, transformaciones u otras propiedades que debe tener el conjunto de valores que constituyen la solución. Aquí no existe una descripción paso a paso para llegar a la solución, sino que al tener propiedades de las anteriormente mencionadas, el sistema debe ser capaz de derivar un esquema de evaluación para computar la solución. Se suele incluir características operacionales para mejorar la eficiencia. Dentro del paradigma declarativo podemos distinguir el paradigma funcional, el lógico y el relacional.

1.9.2.1 Paradigma funcional

La programación funcional proporciona la capacidad para que un programa se modifique a sí mismo y se basa en la utilización de funciones matemáticas, cada una de ellas devuelve un solo valor dada una lista de parámetros. Persigue el objetivo de obtener lenguajes expresivos en los que no sea necesario bajar al nivel de la máquina para describir el proceso llevado a cabo por el programa, logrando que la secuencia de computaciones se rijan solamente por la reescritura de definiciones usando lo que se denominan "definiciones dirigidas" y no permite asignaciones globales llamados efectos colaterales.
Es caracterizada principalmente por su constitución basada únicamente en definiciones de funciones matemáticas, en las que se verifican algunas propiedades como transparencia referencial consistente en que el significado de una expresión depende del significado de sus subexpresiones.

1.9.2.2 Paradigma lógico

La programación lógica consiste en la aplicación del *corpus* de conocimiento sobre lógica para el diseño de lenguajes de programación. Se basa en la Lógica y los programas están construidos únicamente por

expresiones lógicas que son ciertas o falsas en oposición a un expresión interrogativa o expresiones imperativas [12].

La programación lógica es muy usada en aplicaciones de inteligencia artificial. Este tipo de programación tiene varias ventajas en ciertos sistemas ya que sirve para representar conocimiento, permite la representación de información necesaria para resolver un problema efectivamente.

1.9.2.3 Paradigma relacional

Este paradigma está centrado en los datos, basándose en la lógica de predicados y la teoría de conjuntos. Es el modelo que más se utiliza en la actualidad para modelar problemas reales y administrar datos dinámicamente. Aquí los datos se almacenan en relaciones, donde cada una de estas son un conjunto de datos de lo cuales no importa el orden en que se almacenen. Los datos pueden ser modificados o verificados mediante consultas. La información puede ser manejada a través de un lenguaje relacional. En la actualidad existen dos lenguajes de este tipo: el Álgebra Relacional y el Cálculo Relacional.

1.9.3 Paradigmas de programación demostrativos

El paradigma demostrativo o también conocido como programación por ejemplos no especifica como construir la solución sino que presenta soluciones a problemas similares que suelen ser procedimentales. Simula secuencias procedimentales y su principal problema es reconocer cuando un programa es correcto. Tiene varias categorías, paradigma de redes de neuronas y paradigma genético.

1.9.3.1 Paradigmas de redes de neuronas

Este paradigma está basado en de redes de neuronas o redes neuronales las cuales está guiadas por el funcionamiento del sistema nervioso de los animales, ya que simula propiedades observadas en los sistemas biológicos a través de modelos matemáticos. El principal objetivo de estas redes es que las máquinas den respuestas parecidas a las que el cerebro es capaz de dar. Las neuronas son unidades que componen la red neuronal, las cuales mediante interconexiones reciben una serie entradas y emiten una salida.

El propósito de este paradigma es lograr que una red aprenda automáticamente las propiedades que se deseen. El entrenamiento es el proceso por el que los parámetros de la red se adecuan a la solución de cada problema.

1.9.3.2 Paradigmas genético

John Koza propuso el paradigma de programación genético derivado de algoritmos genéticos diferenciándose en la forma de representar a los individuos de la población consistente en que en lugar de utilizar cadenas de longitud fija emplea programas de computadora [13].

Como objetivo central tiene lograr que las computadoras resuelvan, sin ser programadas, problemas hasta obtener soluciones satisfactorias y además, generar automáticamente soluciones a un determinado problema partiendo de la inducción de programas.

La metodología de la programación genética proporciona características muy importantes para el diseño, de manera robusta, de sistemas que actúen sobre condiciones inestables en ambientes cambiantes.

1.10 Estudio del paradigma de programación utilizado

La programación orientada a objetos ha sido el paradigma seleccionado para el desarrollo del subsistema deseado ya que es el modelo que presenta las condiciones necesarias para cumplir con los objetivos trazados. Ayuda a los programadores a agilizar sus tareas, permitiendo la creación de programas más estables, claros y mantenibles en un menor tiempo. Además de que actualmente está siendo considerada la primera posible introducción a los paradigmas de programación futuros.

Al principio de la programación los métodos computacionales estaban guiados principalmente a la realización de cálculos numéricos, por lo que todo dato era visto como un valor numérico simple. El posterior desarrollo de la programación en el transcurso del tiempo dio lugar al concepto actual de la programación orientada a objetos. Sobre la cual hay una gran cantidad de opiniones, pero lo más importante es que ha despertado entusiasmo en los programadores desde los años 1980 y se ha convertido en un tema muy común en el ámbito de la programación. En este modelo, los programas se basan en entidades que combinan las propiedades de los datos y de los métodos que actúan sobre ellos llamadas objetos. Los objetos son instancias de un tipo o una clase, por lo que puede verse o usarse. La programación orientada a objetos no se puede considerar como un modelo en el cual todos los datos son abstractos, ya que los objetos tienen un tipo y pueden ser manejados como una estructura de datos dentro del lenguaje.

La programación orientada a objetos describe un paradigma de programación en el cual los agentes que actúan son entidades independientes cada uno con su propia estructura interna, que se comunican

mutuamente respondiendo o solicitando. Estos objetos, están constituidos de propiedades y operaciones. La estructura interna de un objeto no es accesible por otro objeto o programa.

Los objetos se comunican con otros a través de mensajes. Las peticiones realizadas en forma de mensajes pueden ser para crear nuevos objetos, modificarlos o preguntar sobre el valor de una propiedad determinada, ejecutar operaciones específicas, etc. Para cada mensaje existe un método, el cual es una orden que manipula el mensaje.

Existen dos vías diferentes de crear un objeto. Una forma, es usando como prototipo objetos ya existentes. También se pueden crear a través de las clases, donde todo objeto es una instancia de una clase, y una clase no es más que un patrón para el objeto en el cual se establecen ciertas propiedades y métodos, esta es la vía más usada.

Este tipo de programación ha desarrollado varias técnicas para su mejor desenvoltura, tales técnicas son: la herencia, la modularidad, el polimorfismo y el encapsulamiento.

Herencia

La herencia es una relación entre clases, también llamada relación “es un”, ya que cuando una clase hereda de otra, obtiene por herencia todos los atributos y métodos de la primera. Gracias a esto podemos hacer más fácil el diseño e implementación de un software ya que se puede reutilizar código, o sea los hijos heredan de los padres datos y acciones y ya no es necesario implementarlos en ellos. Todo esto hace de la herencia, algo más que una simple relación.

También sirve como un medio de clasificación. Los objetos sirven para agrupar los métodos con los datos que manipulan, las clases sirven para agrupar objetos y crear colecciones de los mismos, y la herencia permite organizar colecciones de clases.

Existen dos tipos de herencia, la múltiple y la simple, si dos clases B y C heredan de una clase A, y B hereda de C o viceversa, es herencia múltiple; pero si entre B y C no existe relación de herencia entonces es simple. La primera de estas permite que un objeto tenga más de un padre y la segunda no lo admite.

La herencia múltiple tiene la ventaja de que si una clase hereda de varias, el código que se puede reutilizar es mayor, pero trae la gran desventaja de que pueden formarse conflictos que incrementarían la complejidad del sistema.

Esta técnica en general le ofrece muchas ventajas a la programación orientada a objetos:

- Permite representar una jerarquía de clases para describir el dominio en un sistema.

- Es un mecanismo para la reutilización de código.
- Soporta una programación por diferencia, donde se crea una clase que hereda de otra pero describiendo las diferencias entre ellas.
- Sirven para modelar conceptos en una aplicación de clases haciendo más fácil el manejo de las relaciones jerárquicas entre conceptos.
- Ayuda a simplificar el diseño de la aplicación.

También tiene algunas desventajas si no se le sabe dar un adecuado uso:

- A veces se usa exageradamente la herencia, creando clases que heredan mucho comportamiento inapropiado con el objetivo de ganar una característica determinada.
- Se usa para crear clases complejas con poca posibilidad de re-usar el código queriéndose resolver problemas similares.
- Es bueno heredar métodos, pero no así variables, por lo que es mejor heredar de una clase abstracta que de una concreta.

La herencia no permite excluir una operación heredada, por lo que si deseamos no heredar un propiedad determinada, debemos declararla privada en la superclase o clase padre.

Esta técnica es la principal característica que distingue la programación orientada a objetos de otros paradigmas de programación.

Modularidad

Es el proceso de crear partes de un todo que se integran perfectamente entre sí para que funcionen por un objetivo general, y a las cuales se les pueden agregar más componentes que se acoplen perfectamente al todo, o extraerle componentes sin afectar su funcionamiento. En el caso que se requiera actualizar un módulo, no hay necesidad de hacer cambios en otras partes del todo.

La modularidad es el atributo más sencillo del software, que permite a un programa ser manejable intelectualmente. Es comúnmente aceptada tanto para análisis como para diseño.

Tiene grandes ventajas porque nos proporciona:

- Calidad de diseño.
- Facilidad de instrumentación.
- Facilidad de pruebas
- Facilidad de documentación.

- Facilidad de mantenimiento.

Polimorfismo

Uno de los objetivos fundamentales de varias técnicas de la programación orientada a objetos es la reutilización de código, sin embargo algunas operaciones necesitan adaptarse para resolver tareas propias, debido a que en ocasiones una superclase y una subclase necesitan alcanzar los mismos objetos, pero con mecanismos diferentes, esta técnica es llamada polimorfismo.

En general es la habilidad de aceptar más de una forma. Un lenguaje orientado a objetos trata el polimorfismo como la posibilidad de que objetos de tipos diferentes respondan al mismo mensaje con su propio comportamiento, o sea según esté implementado en su interior.

En muchos de los sistemas que soportan la programación orientada a objetos el polimorfismo es generalizado en el sentido de que en la jerarquía de clases cualquier mensaje puede ser polimorfo.

El polimorfismo aporta grandes ventajas a este paradigma, ya que:

- Permite referirse a objetos de subclases usando una referencia a la superclase.
- Puede usarse variables referencia de una superclase para almacenar direcciones de objetos de una subclase.
- Se puede hacer una solicitud de una operación sin conocer el método que debe ser llamado, quedando ocultos estos detalles de la implementación.

Encapsulamiento

Este término muy ligado a la abstracción de datos, a los tipos de datos abstractos y a los objetos. Se utiliza para denominar la información oculta, donde los datos se empaquetan junto a operaciones que actúan sobre ellos.

Esta técnica puede ser usada para evitar el directo acceso por parte de las clases hijas a las variables de las clases padres, ya que desde el punto de vista de protección, ningún objeto debe acceder a los datos internos de otro, y en el modelo orientado a objetos se presentan algunas interferencias.

Contribuye con muchas ventajas a este paradigma:

- Posibilita que la implementación de un objeto se pueda cambiar sin afectar a las aplicaciones que lo utilizan.
- El programador puede manejar las estructuras sin preocuparse de su implementación física.

- Los programas son legibles y fáciles de mantener.
- Las estructuras encapsuladas son fáciles de reutilizar en otros programas.

El paradigma orientado a objetos es tan flexible, legible y productivo que muchos lenguajes son basados en él. Más a delante se hace un análisis del lenguaje que ha sido elegido para la implementación de los servicios de consulta, seguridad y notificación de cambios en UDDI.

1.11 Lenguaje de programación utilizado (PHP)

El lenguaje de programación usado es un lenguaje script del lado del servidor conocido como PHP, es utilizado para la creación de páginas Web dinámicas. Para muchos no es un lenguaje totalmente orientado a objetos, lo que es realmente cierto, porque también puede utilizarse la programación estructurada tradicional, aunque en la actualidad se ha enfocado más al modelo orientado a objetos.

Fue inicialmente diseñado en Perl, su sintaxis en gran parte proviene de C, Java y Perl, aunque contiene características propias. En 1994 fue programado por Rasmus Lerdorf con el objetivo de guardar algunos datos, al Rasmus ver el inmenso tráfico que recibía su página, publicó en 1995 el Personal Home Page Tools ya combinado con su Form Interpreter. En el año 1997 dos programadores israelíes rescribieron el analizador sintáctico creando la base del PHP3 y modificando el nombre a su estado actual, PHP Hypertext Pre-processor, el cuál fue publicado oficialmente en junio del 1998. Posteriormente se han lanzado otras versiones como, PHP4 en mayo del 2000 y PHP5 en julio del 2004, este último con mejor soporte para la programación orientada a objetos, MySQL y XML, con soporte para SQLite y SOAP, y con iteradores de datos y manejo de excepciones [14].

El gran parecido que tiene PHP con los lenguajes en los que se apoyó, permite que los programadores no pasen mucho trabajo al desarrollar proyectos complejos, ya que no tienen que aprender todo un grupo de nuevas funciones. Además de facilitar la creación de páginas Web, también posibilita crear interfaz gráfica para el usuario.

El programa PHP es ejecutado en el servidor enviando el resultado al navegador, a diferencia de otros lenguajes script, por esto es que permite acceder a los recursos que posea el servidor. Generalmente el resultado es una página HTML. Es independiente del navegador que se use, o sea, no es necesario que el navegador lo soporte debido a que se ejecuta en el servidor, pero si lo debe soportar el servidor.

Es un lenguaje potente que puede acceder a ficheros, ejecutar comandos y abrir comunicaciones de red en el servidor, todo esto hace que lo que se ejecute en el servidor Web sea seguro por defecto. Si se hace

una adecuada selección de las opciones de configuración de tiempo de compilación y ejecución se llega a la perfecta mezcla de libertad y seguridad, porque ha sido diseñado para ser muy seguro.

Presenta gran cantidad de características que lo hacen ser una herramienta ideal para crear páginas Web dinámicas [14]:

- Permite generar documentos PDF y analizar códigos XML, porque integra varias librerías externas.
- Posibilita la implementación de servicios Web.
- Es más fácil de mantener y de actualizar que el código en otros lenguajes.
- Como PHP es un producto de código abierto, hace que sus fallos sean encontrados y reparados rápidamente por un gran grupo de programadores en todo el mundo.
- Es multiplataforma, lo que permite que se pueda utilizar en la mayoría de los sistemas operativos y que pueda interactuar con muchos servidores Web.
- Posibilita la conexión a muchos manejadores de bases de datos que se utilizan en la actualidad.
- Posee una amplia documentación, destacando la ayuda, donde se explican claramente todas las funciones que tiene.
- Todos tienen acceso a él, ya que es libre.
- Maneja excepciones.

PHP5 es la versión ideal del lenguaje, para el desarrollo del subsistema de búsqueda, seguridad y suscripción para el registro UDDI.

¿Por qué PHP5?

Las dos primeras versiones de PHP habían logrado una plataforma estable y potente para la implementación de páginas del lado del servidor. Los desarrolladores se han apoyado mucho en estas versiones, tanto así que PHP es el lenguaje más usado en el mundo para el desarrollo de páginas dinámicas.

Sin embargo, existían ciertos problemas que se han tratado de solucionar con PHP5, principalmente la programación orientada a objetos, que solo se ofrecían pocas características de ella en las versiones anteriores. El mejoramiento de este paradigma ha sido el fundamental objetivo de PHP5.

PHP es una alternativa a las tecnologías de Microsoft ASP y ASP.NET las que utilizan C# y VB.NET como lenguajes, a ColdFusion de la compañía Adobe antes de ser de Macromedia, a JSP y Java de Sun Microsystems, y a CGI y Perl. Su creación y desarrollo se da en el ámbito de los sistemas libres, bajo la licencia GNU, pero existe un IDE comercial llamado Zend Studio que ha dado las soluciones más

completas para el desarrollo de PHP y es el más adecuado para la implementación de las APIs, ya que ofrece elementos óptimos y necesarios para trabajar.

También PHP permite la conexión a diferentes tipos de servidores de bases de datos tales como MySQL, Postgres, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite.

1.12 Entorno de desarrollo (Zend Studio)

Zend Studio es un Integrated Development Environment (IDE) propietario destinado a desarrolladores profesionales, el cual es compatible con las plataformas GNU/Linux, MAC y Windows. Para el ciclo de vida de una aplicación PHP, presenta todos los componentes necesarios. Incluye editor, análisis, depuración, optimizadores de código y herramientas de base de datos. Agiliza el desarrollo Web y simplifica proyectos complejos.

Existen dos ediciones de Zend Studio: Standard y Professional [15]. Es el soporte más completo en herramientas para la creación de aplicaciones altamente fiables en desarrollos y pruebas de PHP.

Cuenta con un entorno de prueba que agiliza la seguridad de la calidad, integración y las etapas de los procesos. Proporciona las extensiones requeridas para el desarrollo de aplicaciones PHP. Presenta un excelente completamiento de código y coloreado de sintaxis para facilitar el trabajo de los programadores y para evitar posibles errores. Posee un manual de PHP. Simplifica el despliegue con la integración de FTP y SFTP lo que permite subir y descargar archivos de forma segura hacia y desde servidores remotos.

Puede conectarse directamente a muchas de las bases de datos utilizadas en el desarrollo de proyectos tales como: IBM DB2/Cloudscape/ Derby/, MySQL, Oracle, Microsoft SQL Server, PostgreSQL y SQlite [16]. También escribir y realizar consultas a servidores conectados. Puede visualizar las estructuras de la base de datos y administrar el contenido.

La depuración es otro aspecto muy tenido en cuenta en este IDE. La depuración incluye características avanzadas como condiciones límites, visualización de errores y variables. Depura de forma local y remota. Depura y analiza el código directamente desde un navegador.

Facilita el trabajo en equipo mediante la administración efectiva del código fuente a través de CVS y subversión.

1.13 Servidor Web (Apache)

En la actualidad el servicio más utilizado en Internet son los servicios Web, por lo que los servidores Web son tan importantes y se usan tanto. El servidor Web Apache es una de los más usados, ya que lo usan el 60% de los administradores de la red. Es la plataforma de servidores Web de código abierto más potente del mundo, la cual puede ser usada por GNU/Linux, Macintosh, Windows, entre otros sistemas operativos. Su desarrollo comenzó en el año 1995 dentro del proyecto HTTP Server, y su nombre se debe a la firmeza y energía de una tribu aborigen que llevaba este nombre. Su primera versión fue la 0.6.2 y se dio a conocer en Abril de 1995, en diciembre de ese mismo año ya estaba disponible la versión 1.0 con muchos errores corregidos de la primera versión.

Es un servidor con diseño modular altamente configurable y ampliar sus funcionalidades es algo muy sencillo. Sus módulos se pueden clasificar en tres categorías [17]:

- **Módulos Base**: Son los que tienen las funcionalidades básicas de Apache.
- **Módulos Multiproceso**: Son los que se encargan de la unión con los puertos de las máquinas, aceptando y atendiendo peticiones.
- **Módulos Adicionales**: Son los que le añadan una funcionalidad al servidor.

Existen varios módulos multiproceso para cada uno de los sistemas operativos sobre los que se ejecuta el Apache, optimizando el rendimiento y rapidez del código.

El resto de las funcionalidades del servidor se consiguen por medio de módulos adicionales que se pueden cargar. Para añadir un conjunto de utilidades al servidor, simplemente hay que añadirle un módulo, de forma que no es necesario volver a instalar el software.

Trabaja con la mayoría de los lenguajes Web de la actualidad como CGI, Perl, PHP y muchos otros lenguajes script.

La importante labor desplegada por sus desarrolladores permite que en Apache se pueda depositar gran confianza, ya que es muy rápido y estable, además de ser fácilmente adaptable a nuevas tecnologías y protocolos. El carácter comunitario de este servidor permite que muchas personas se involucren en su desarrollo y aporten módulos que al final benefician los usuarios.

Las principales características que presenta Apache son [18]:

- Es utilizable tanto en sistemas Unix como en sistemas Windows, constando con distintas versiones para cada uno de ellos.

- Es de código abierto y gratuito es su totalidad.
- Se arquitectura modular posibilita la construcción del propio servidor mediante paquetes pequeños.
- Su configuración se basa en su archivo httpd.conf, el cual es editable con cualquier editor de textos.
- Soporta host virtuales.
- Contiene autenticación HTTP.
- Soporta SSL.
- Posee mensajes de error altamente configurables, base de datos de autenticación y negociado de contenido.

Apache incluye muchos programas ejecutables internamente para su buen funcionamiento, algunos de estos programas son:

- **httpd**: Servidor Apache del Protocolo de Transmisión de Hipertexto (HTTP)
- **apachectl**: Interfaz de control del servidor HTTP Apache
- **apxs**: Herramienta de Extensión de Apache
- **configure**: Configuración de la estructura de directorios de Apache
- **htpasswd**: Crea y actualiza los ficheros de autentificación de usuarios para autentificación básica
- **rotatelogs**: Renueva los logs de Apache sin parar el servidor

1.14 Conclusiones

En este capítulo se han presentado y fundamentado todo una serie de aspectos relacionados con el desarrollo e implementación de las APIs de búsqueda y seguridad de UDDI. Estos procesos necesitan de un grupo de condiciones para su correcto progreso, las cuales se han seleccionado de la siguiente manera:

- Modelo de arquitectura SOA, ya que permite organizar y utilizar capacidades distribuidas, y provee una manera uniforme de ofrecer, descubrir, interactuar y usar estas capacidades para producir los efectos deseados de manera consistente y medible.
- El lenguaje de programación PHP5 por ser un lenguaje script del lado del servidor, utilizado para la creación de páginas Web dinámicas y por basarse principalmente en la programación orientada a objetos.

- Zend Studio como entorno de desarrollo porque es el soporte más completo en herramientas para la creación de aplicaciones altamente fiables en desarrollos y pruebas, de PHP.
- Como servidor Web, Apache, porque es la plataforma de servidores Web de código abierto más potente del mundo.

Capítulo 2. Descripción y análisis de la solución propuesta

2.1 Introducción

En este capítulo se tratan varios aspectos ya más inmersos en el desarrollo del subsistema. Se comienza realizando una valoración del diseño que se propone en las especificaciones de UDDI 3.0.2 [7] el cual se ha seguido para la implementación de la APIs de búsqueda y seguridad de UDDI. También se describe y examina un algoritmo no trivial de los que son considerados como más importantes dentro del registro, incluyendo el análisis de la complejidad del mismo. Por último se hace una descripción de clases y operaciones utilizadas.

2.2 Valoración del diseño propuesto

El desarrollo del trabajo ha sido guiado por el diseño planteado en especificaciones de UDDI 3.0.2 [7], definidas por OASIS. En este diseño se explica claramente el desempeño de las estructuras y funcionalidades de UDDI, que han sido sujetas a mejoras en cada una de sus versiones.

La información de UDDI se encuentra dividida en tres secciones fundamentales (páginas blancas, amarillas y verdes), en las cuales se almacenan los datos de sus principales estructuras.

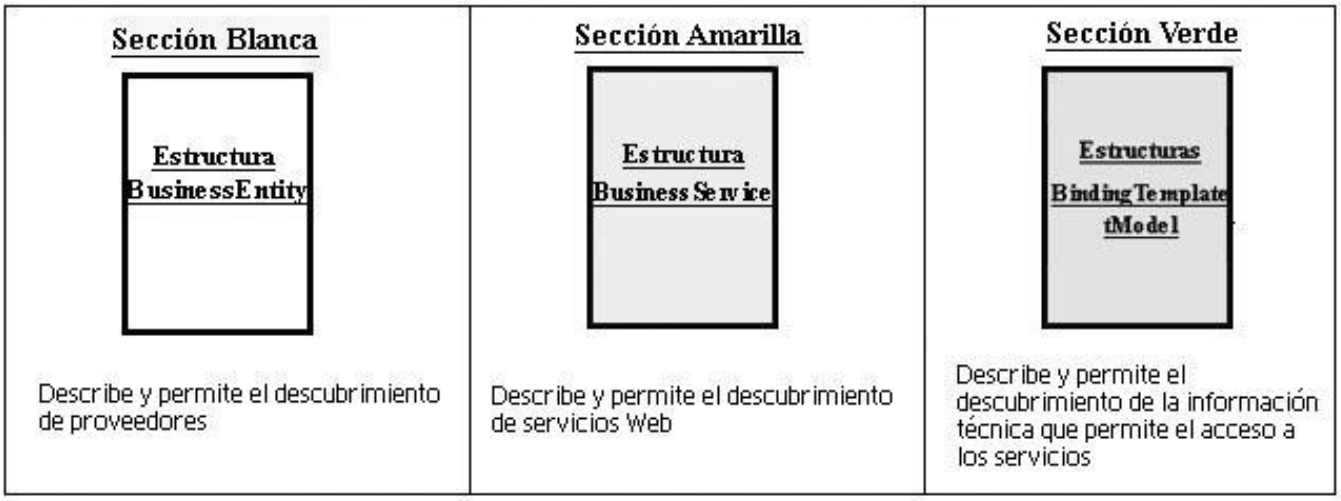


Figura # 1: Secciones de UDDI.

En este diseño se proponen estas estructuras mencionadas anteriormente como las clases principales de la implementación de un registro UDDI, sobre las cuales se comienza a desarrollar un conjunto de funcionalidades que necesitan de la implementación de otras importantes clases.

2.2.1 Análisis y descripción de las estructuras generales

BusinessEntity

La estructura BusinessEntity contiene una cierta cantidad de atributos que permiten almacenar los datos necesarios de los proveedores de servicios Web.

- **businessKey:** identifica la entidad de negocio, puede ser opcional ya que el publicador puede dejar que sea generado por el propio registro UDDI.
- **discoveryURLs:** contiene una serie de URLs que apuntan a mecanismos de descubrimiento de servicios Web.
- **name:** contiene el nombre del proveedor, el cual puede expresarse en múltiples idiomas.
- **description:** contiene la descripción del proveedor de servicios Web, la cual puede expresarse en múltiples idiomas.
- **contacs:** contiene una serie de contactos de personas o puestos de trabajo del proveedor.
- **businessServices:** contiene una lista de servicios que ofrece.
- **identifierBag:** permite identificar a los proveedores según una serie de estructuras. keyedReference, las cuales en este caso, cada una representa una identificación.
- **categoryBag:** permite categorizar a los proveedores según una serie de estructuras. keyedReference, las cuales en este caso, cada una representa una categorización.
- Signature: permite almacenar información de la firma digital de un proveedor.

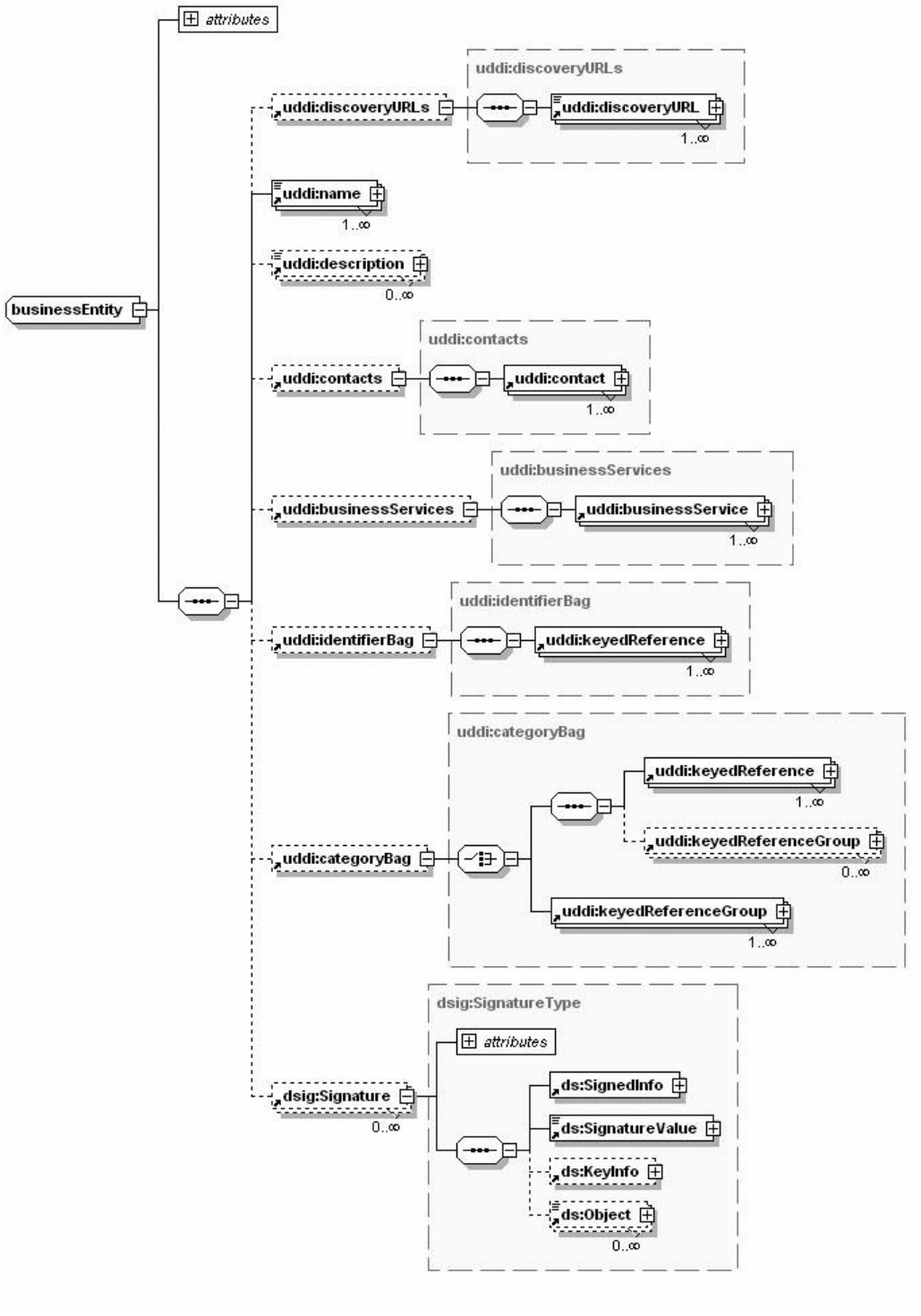


Figura # 2: Esquema representativo de businessEntity.

BusinessService

Esta estructura contiene la información descriptiva de un servicio Web.

- **businessKey:** contiene el identificador del proveedor de servicios que contiene.
- **serviceKey:** contiene el identificador del propio servicio Web.
- **name:** contiene el nombre del servicio Web, el cual puede expresarse en múltiples idiomas.
- **description:** contiene la descripción del servicio Web, la cual puede expresarse en múltiples idiomas.
- **bindingTemplates:** contiene una lista de descripciones técnicas del servicio Web.
- **categoryBag:** permite categorizar a los servicios Web según una serie de estructuras.
- Signature: permite almacenar información de la firma digital de un proveedor.

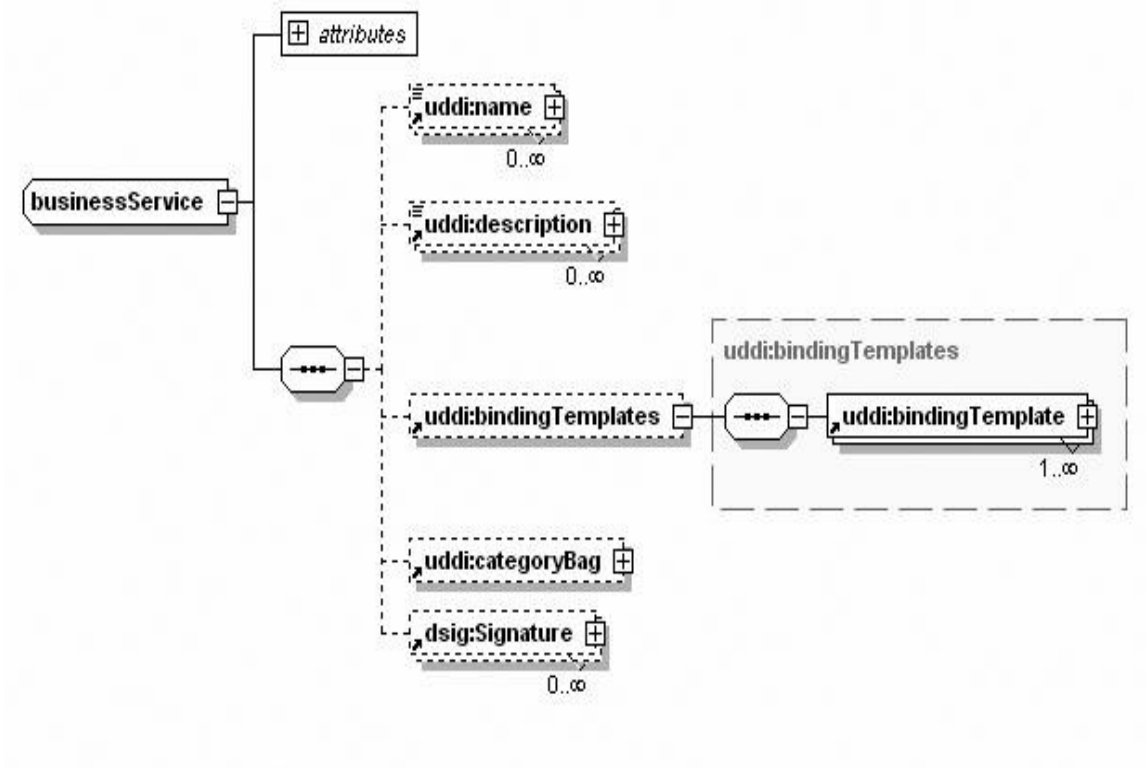


Figura # 3: Esquema representativo de businessService.

BindingTemplate

Esta estructura describe técnicamente un servicio Web.

- **serviceKey:** contiene el identificador del servicio Web al que pertenece.

- **bindingKey:** contiene el identificador la propia descripción técnica.
- **description:** contiene la descripción de esta descripción técnica.
- **accessPoint:** cadena que contiene la URL de invocación al servicio Web.
- **tModelInstanceDetails**: contiene una serie de elementos **tModelInstanceInfo**, los cuales contienen datos del **tModel** que representa.
- **instanceDetails**: permite especificar alguna configuración concreta del **tModel**.
- **categoryBag:** contiene categorizaciones que describen aspectos del mismo.
- **Signature**: permite almacenar información de la firma digital de un proveedor.

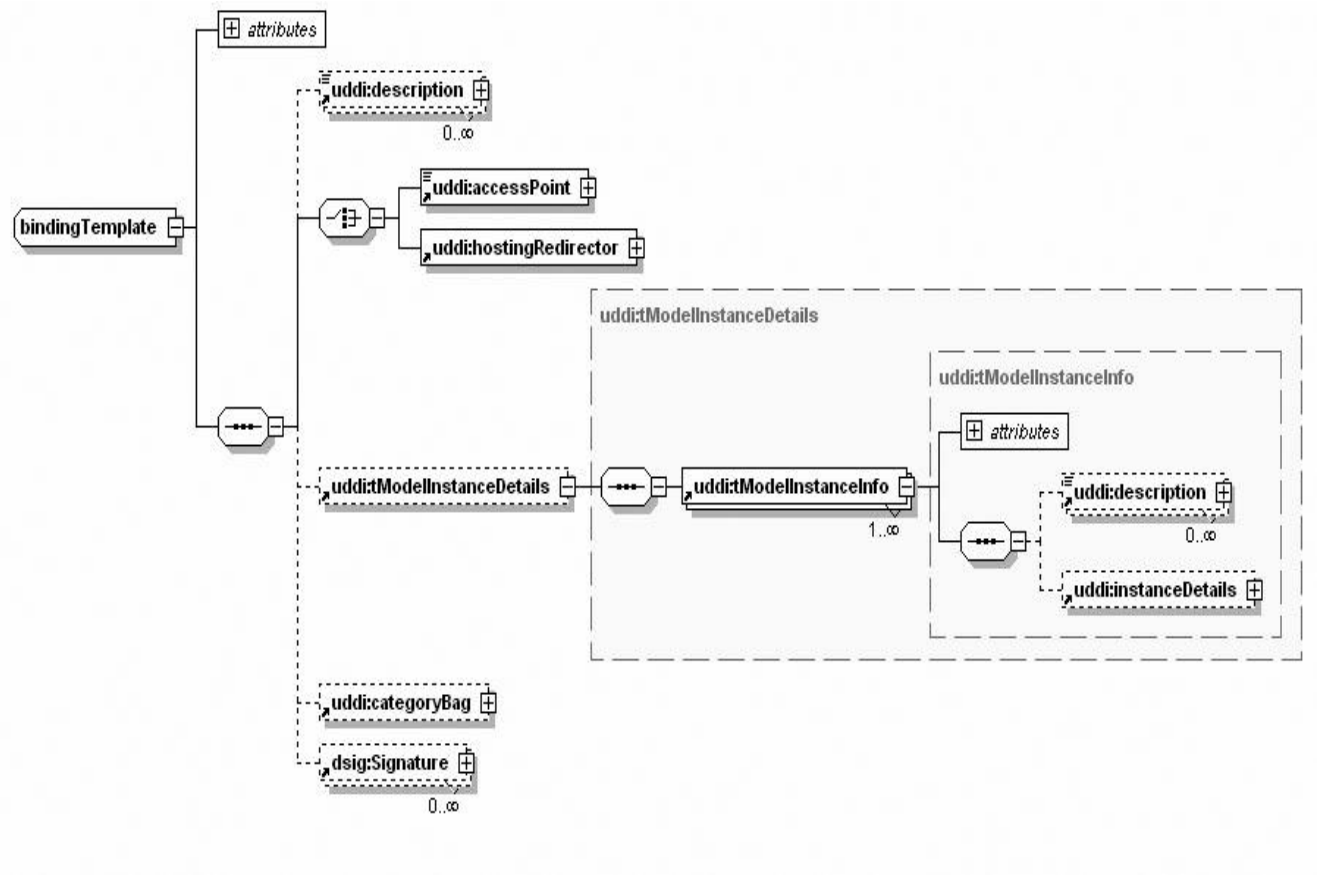


Figura # 4: Esquema representativo de bindingTemplate.

TModel

Esta estructura permite describir especificaciones, conceptos o diseños compartidos.

- **name:** contiene el nombre del modelo técnico, el cual no puede ser en varios idiomas.

- **description:** contiene la descripción del modelo técnico.
- **overviewDoc:** permite almacenar referencias a descripciones generales remotas o instrucciones relativas al modelo técnico.
- **identifierBag:** permite identificar a los modelos técnicos según una serie de estructuras.
- **categoryBag:** permite categorizar a los modelos técnicos según una serie de estructuras.
- Signature: permite almacenar información de la firma digital de un proveedor.

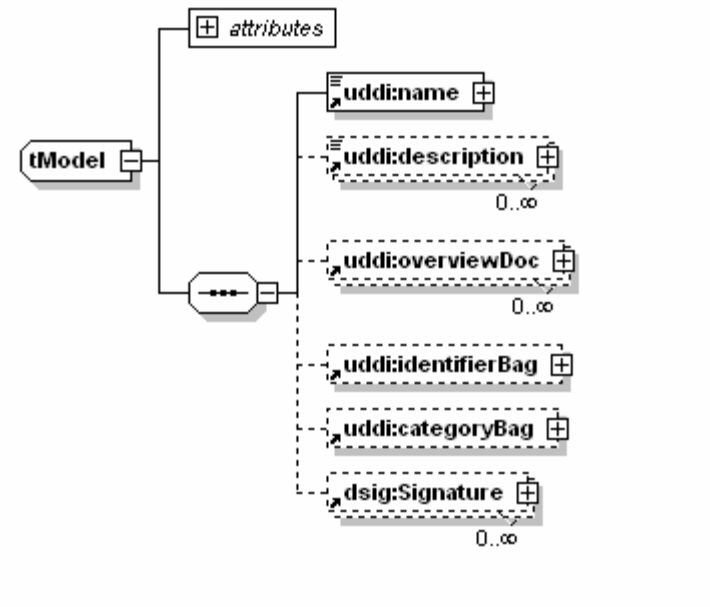


Figura # 5: Esquema representativo de tModel.

En la descripción anterior se muestran las principales clases de UDDI propuestas por OASIS plasmadas en un diseño en el cual un BusinessEntity (*Entidad de Negocio*) contiene BusinessServices (*Servicios Web*). Un BusinessService contiene BindingTemplates (*Descripciones Técnicas*) y estos a su vez TModels (*Modelos Técnicos*). Estas dependencias entre clases se corresponden con las relaciones lógicas que existen entre estas estructuras.

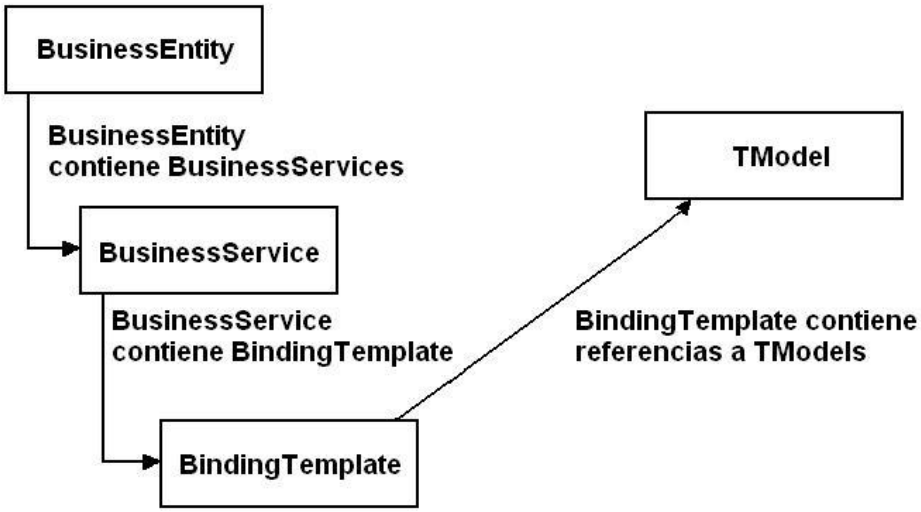


Figura # 6: Relación entre las estructuras de UDDI.

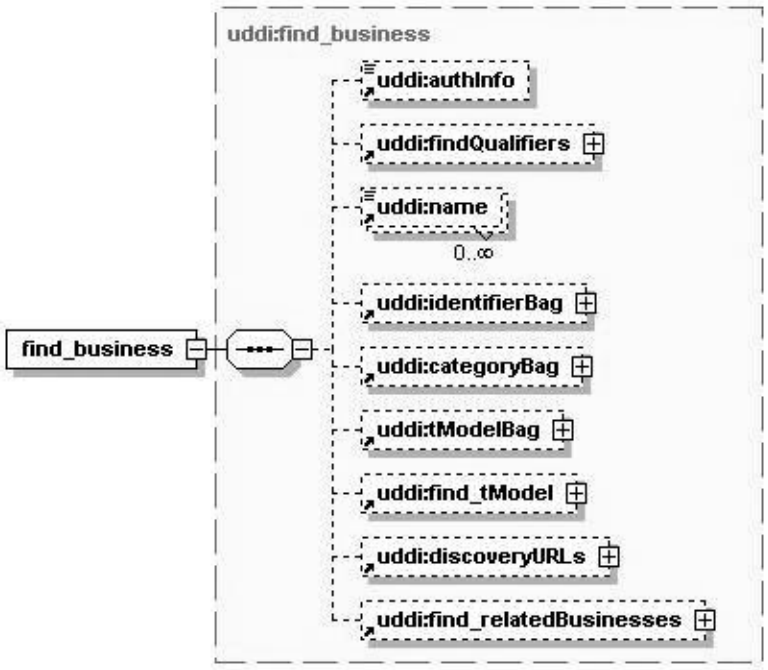
2.2.2 Análisis y descripción de las principales clases y funcionalidades

Las especificaciones de UDDI 3.0.2 [7] proponen funcionalidades para las APIs, Inquiry y Security que se implementan mediante clases que hacen posible una mejor factibilidad de estas funciones.

2.2.2.1 Clases y funcionalidades de Inquiry API

Inquiry API posibilita una configuración simple y completa de interfaces de programación que es empleada en la búsqueda de registros UDDI para localizar entradas a estos detallando las particularidades técnicas o necesidades de negocio, además de brindar información minuciosa de cada entrada a registro cuando son halladas. La versión 3.0.2 [7] brinda nueve funcionalidades establecidas para esta importante API mostrándose como coberturas de programación necesarias para un mejor resultado del producto.

find_business: Clase que es utilizada por la función **FindBusinessFunction** para encontrar elementos **businessEntity.**

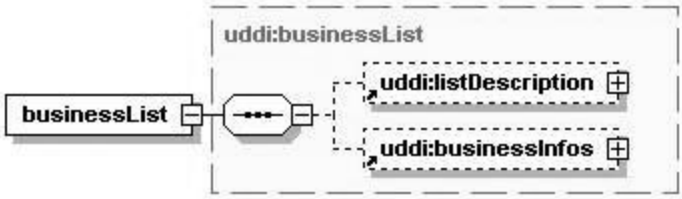


Atributos

Name	Use
maxRows	optional
listHead	optional

Figura # 7: Esquema representativo de find_business.

FindBusinessFunction: Devuelve un **businessList** según los criterios de búsqueda especificados, el cual contiene información acerca de los proveedores, incluyendo resúmenes de sus servicios Web.

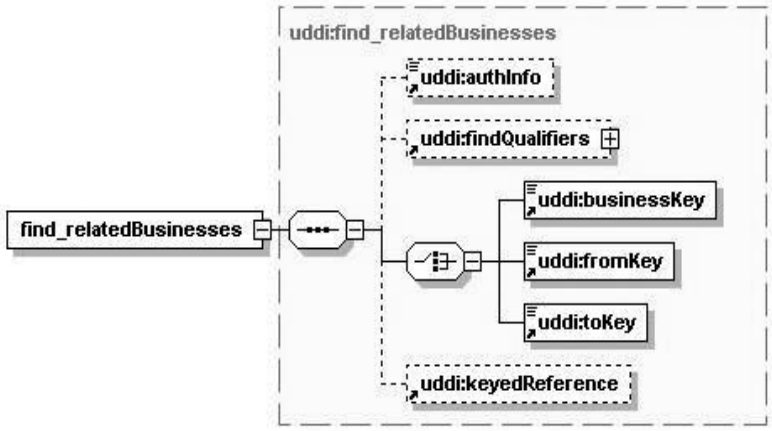


Atributos

Name	Use
truncated	optional

Figura # 8: Esquema representativo de businessList.

find_relatedBusinesses: Clase usada por la función **FindRelatedBusinessesFunction** para encontrar **businessEntity** relacionados con otros especificados en los criterios de búsqueda.

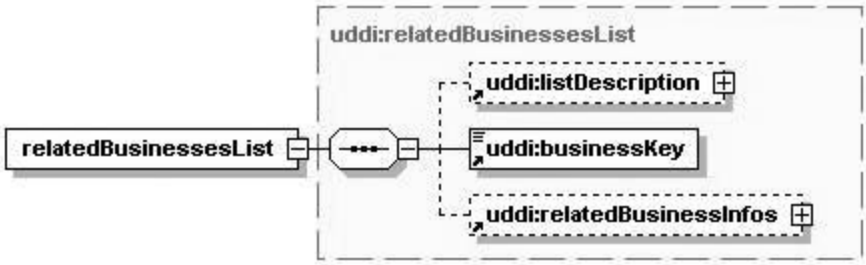


Atributos

Name	Use
maxRows	optional
listHead	optional

Figura # 9: Esquema representativo de find_relatedBusinesses.

FindRelatedBusinessesFunction: Devuelve una estructura **relatedBusinessesList**.

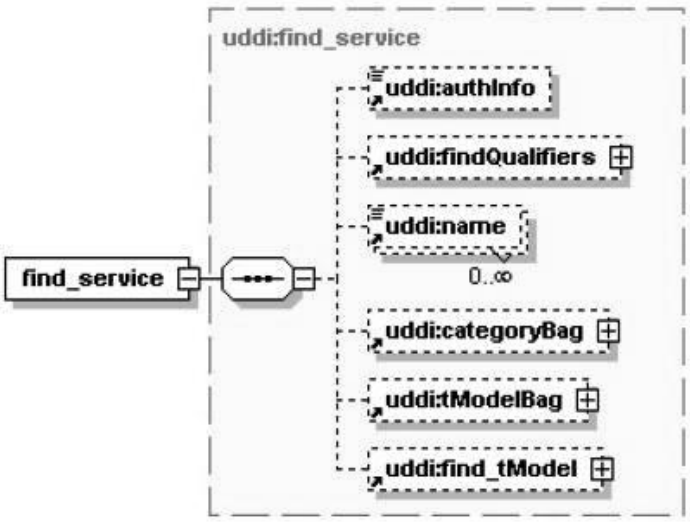


Atributos

Name	Use
truncated	optional

Figura # 10: Esquema representativo de relatedBusinessesList.

find_service: Clase utilizada por la función **FindServiceFunction** para encontrar elementos **businessService.**

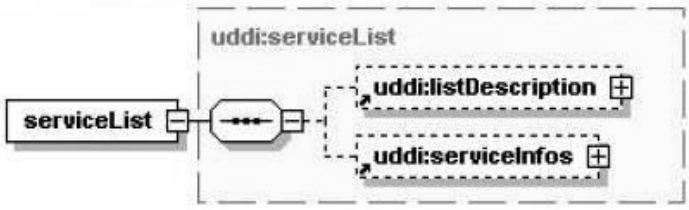


Atributos

Name	Use
maxRows	optional
businessKey	optional
listHead	optional

Figura # 11: Esquema representativo de find_service.

FindServiceFunction: Retorna una estructura **serviceList** según las condiciones especificadas en los argumentos. Si la búsqueda no es exitosa la **serviceList** es vacía.

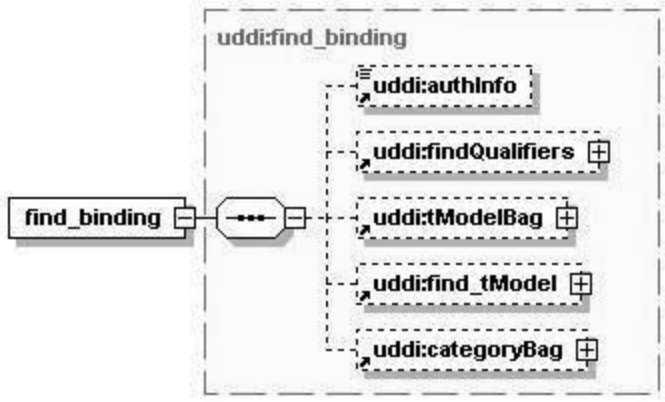


Atributos

Name	Use
truncated	optional

Figura # 12: Esquema representativo de serviceList.

find_binding: Clase utilizada por la función **FindBindingFunction** para encontrar elementos **bindingTemplate**.

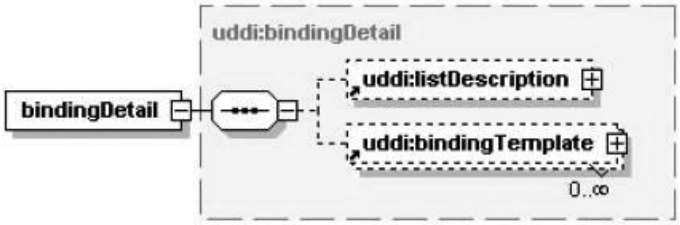


Atributos

Name	Use
maxRows	optional
serviceKey	optional
listHead	optional

Figura # 13: Esquema representativo de find_binding.

FindBindingFunction: Devuelve un **bindingDetail** que contiene cero o más estructuras **bindingTemplate** según los criterios especificados en la lista de argumentos. En caso que no se encuentren coincidencias con los criterios especificados, se retorna un **bindingDetail** vacío.

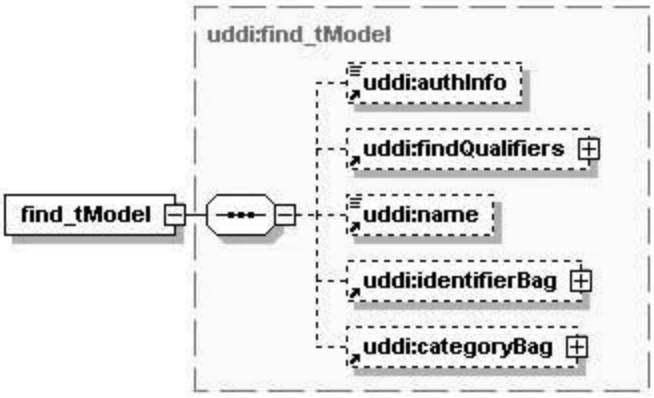


Atributos

Name	Use
truncated	optional

Figura # 14: Esquema representativo de bindingDetail.

find_tModel: Clase usada por la función **FindTModelFunction** para buscar elementos **tModel**.



Atributos

Name	Use
maxRows	optional
listHead	optional

Figura # 15: Esquema representativo de find_tModel.

FindTModelFunction: Retorna un resumen de sus datos en una estructura **tModelList**.

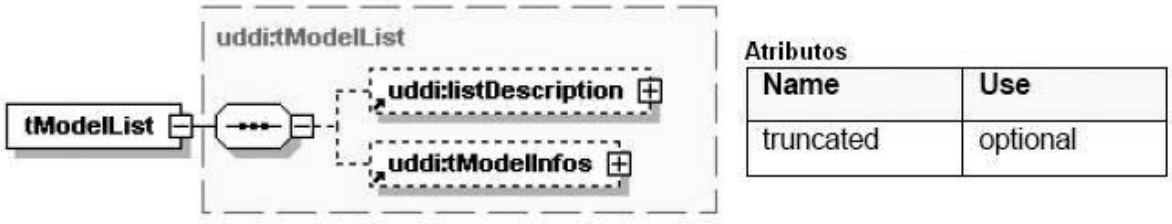


Figura # 16: Esquema representativo de tModelList.

get_bindingDetail: Clase usada por la función **GetBindingDetailFunction** para obtener información completa de **bindingTemplate** correspondiente a cada uno de los valores especificados por **bindingKey**, los datos obtenidos se caracterizan por su adaptabilidad para realizar una o más peticiones de servicio.

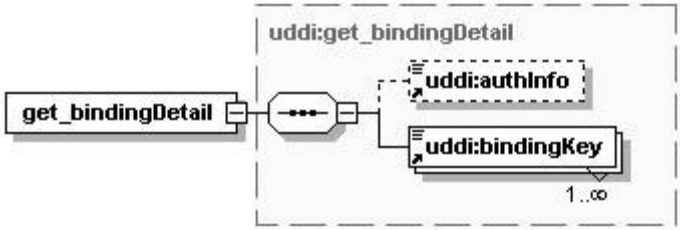


Figura # 17: Esquema representativo de get_bindingDetail.

get_businessDetail: Clase usada por la función **GetBusinessDetailFunction** para obtener información completa de los **businessEntity** correspondientes a cada uno de los valores especificados por **businessKey** para uno o más negociadores u organizaciones.

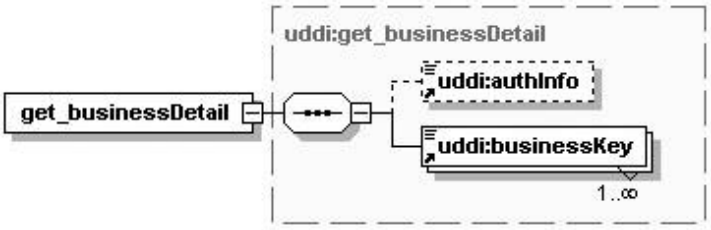
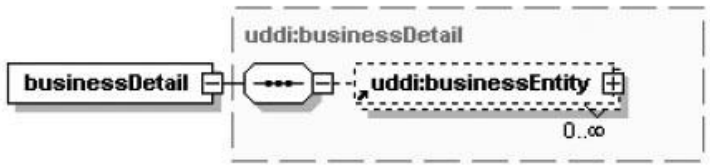


Figura # 18: Esquema representativo de get_ businessDetail.

GetBusinessDetailFunction: Retorna una estructura **businessDetail**.



Atributos

Name	Use
truncated	optional

Figura # 19: Esquema representativo de businessDetail.

get_businessDetailExt: Clase usada por la función **GetBusinessDetailExtFunction** para obtener información extendida de **businessEntity**.

get_serviceDetail: Clase usada por la función **GetServiceDetailFunction** para obtener detalles completos para una situación dada de una fecha registrada sobre un **businessService** correspondiente a cada uno de los valores especificados por **serviceKey**.

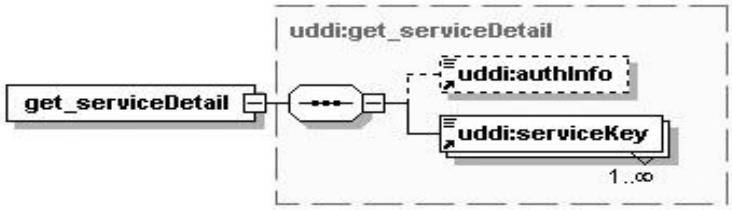
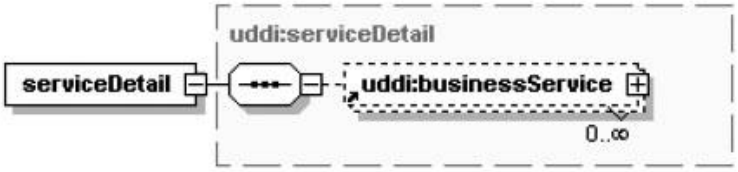


Figura # 20: Esquema representativo de get_serviceDetail.

GetServiceDetailFunction: Retorna una estructura **serviceDetail**.



Atributos

Name	Use
truncated	optional

Figura # 21: Esquema representativo de serviceDetail.

get_tModelDetail: Clase usada por la función **GetTModelDetailFunction** para obtener detalles completos de un **tModel** registrado.

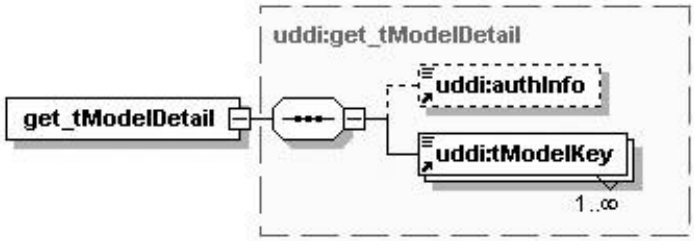
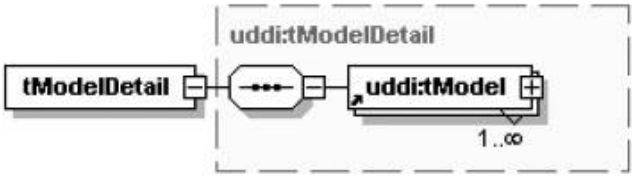


Figura # 22: Esquema representativo de get_tModelDetail.

GetTModelDetailFunction: Retorna una estructura **tModelDetail**.



Atributos

Name	Use
truncated	optional

Figura # 23: Esquema representativo de get_tModelDetail.

2.2.2.2 Clases y funcionalidades de Security Policy API

La seguridad establecida para UDDI se refiere a la incorporación de políticas de autenticación y la autorización de entradas al registro obteniendo credenciales de seguridad imprescindibles para la utilización de una parte o de la totalidad del sistema que distinguen a cada uno de los usuarios.

Entre las funcionalidades necesarias para un correcto desempeño de las políticas de seguridad se encuentran:

discard_authToken: Clase usada por la función **DiscardAuthTokenFunction** para informar a un nodo UDDI que una previa autenticación de Token no es necesaria y debe ser considerada nula si se utiliza después de que esta llamada sea recibida.

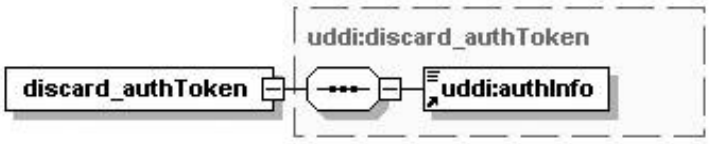


Figura # 24: Esquema representativo de discard_authToken.

get_authToken: Clase usada por la función **GetAuthTokenFunction** para solicitar un token de autenticación en la forma de un elemento **authInfo** de un nodo UDDI. Un elemento **authInfo** puede ser necesario cuando se utilizan las llamadas API.

Name	Use
userID	Required
cred	Required

Figura # 25: Esquema representativo de get_authToken.

2.3 Descripción de un algoritmo no trivial. Análisis de complejidad

Para poder darle el empleo y la funcionalidad correcta a las APIs implementadas fue necesario hacer un profundo análisis de algunos algoritmos implementados que juegan un papel fundamental en el momento de la utilización del subsistema. A continuación se expone el análisis de un algoritmo de búsqueda, donde se incluye la complejidad que presenta.

El algoritmo no trivial seleccionado para ser descrito y analizado es el que implementa la funcionalidad **FindServiceFunction**, ya que el funcionamiento de una UDDI como directorio de servicios Web depende en gran medida de la correcta localización de estos. Esta funcionalidad se nutre de atributos necesarios para la obtención de la información deseada. Se estableció que el atributo businessKey fuera tomado en cuenta para encontrar con mayor agilidad una entidad de negocio (*businessEntity*) y así llegar a los servicios Web que ofrece. La implementación de este método posibilita un retorno de una lista de servicios (*servicelist*). Los criterios de búsqueda pueden ser diversos, por ejemplo, si lo que se desea es obtener todos los servicios de un determinado proveedor se debe especificar como criterio de búsqueda el businessKey y entonces la estructura servicelist devuelta es exclusivamente de los servicios Web brindados por el proveedor solicitado. Si la búsqueda no tiene éxito se devuelve la lista de servicios vacía.

Para el cálculo de la complejidad del algoritmo se deben llevar a cabo una serie de pasos muy importantes [19]:

Primeramente es necesario marcar con un número cada instrucción o grupo de instrucciones del código para representar cada lugar del camino que puede seguir la secuencia del algoritmo.

```
function execute($regObject)
 {
   $request = $regObject; 1
   $generic = $request->getGeneric(); 1
   $businessKey = $request->getBusinessKey(); 1
   $nameVector = $request->getNameVector(); 1
   $categoryBag = $request->getCategoryBag(); 1
   $tModelBag = $request->getTModelBag(); 1
   $qualifiers = $request->getFindQualifiers(); 1
   $maxRows = $request->getMaxRows(); 1

   if((($businessKey == null) || (sizeof($businessKey) == 0)) && (($nameVector == null) || (sizeof($nameVector) == 0))
      && (($categoryBag == null) || (sizeof($categoryBag) == 0)) && (($tModelBag == null)   || (sizeof($tModelBag) == 0))) 1
   {
     $list = new ServiceList(); 2
     $list->setServiceInfos(new ServiceInfos()); 2
     $list->setGeneric($generic); 2
     $list->setOperator($Config->getOperator()); 2
     $list->setTruncated(false); 2
     return $list;  2
   } 3
   $tModel = new TModel(); 4
   if ($categoryBag != null) 4
   {
     $keyedRefVector = $categoryBag->getKeyedReferenceVector(); 5
     if ($keyedRefVector != null) 5
     {
       $vectorSize = sizeof($keyedRefVector); 6
       if ($vectorSize > 0) 6
       {
         for ($i=0; $i<$vectorSize; $i++) 7
        {
          $keyedRef = $keyedRefVector[$i]; 8
          $key = $keyedRef->getTModelKey();8
            if (($key == null) || (strlen(trim($key)) == 0)) 8
            $keyedRef->setTModelKey($tModel-> GENERAL_KEYWORDS_TMODEL_KEY); 9
        }
```

```
    } 10
   } 11
  } 12
$dataStore = new JDBCDataStore(); 13

  if ($nameVector != null) 13
  {
    $maxNames = $Config->getMaxNameElementsAllowed(); 14
    if (($nameVector != null) && (sizeof($nameVector) > $maxNames)) 14
      throw new TooManyOptionsException("find_service: "."names=".sizeof($nameVector).", "."maxNames=" + $maxNames); 15

    $maxNameLength = $Config->getMaxNameLengthAllowed(); 16
    for ($i=0; $i<sizeof($nameVector); $i++) 17
    {
      $name = $nameVector[$i]->getValue(); 18
       if (strlen($name) > $maxNameLength) 18
        throw new NameTooLongException("find_service: "."name=".$name.", "."length=".strlen($name).", ".
                                                "maxNameLength=".$maxNameLength); 19
    }
  } 20
  $findQualifier = new FindQualifier(); 21
  if ($qualifiers != null) 21
  {
    $qVector = $qualifiers->getFindQualifierVector(); 22
    if (($qVector!=null) && (sizeof($qVector) > 0)) 22
    {
      for ($i=0; $i<sizeof($qVector); $i++) 23
      {
        $qualifier = $qVector[$i]; 24
        $qValue = $qualifier->getValue();24

        if ((!$qValue->equals($findQualifier->EXACT_NAME_MATCH))
        && (!$qValue->equals($findQualifier->CASE_SENSITIVE_MATCH))
        && (!$qValue->equals($findQualifier->OR_ALL_KEYS))
        && (!$qValue->equals($findQualifier->OR_LIKE_KEYS))
        && (!$qValue->equals($findQualifier->AND_ALL_KEYS))
        && (!$qValue->equals($findQualifier->SORT_BY_NAME_ASC))
```

```
        && (!$qValue->equals($findQualifier->SORT_BY_NAME_DESC))
        && (!$qValue->equals($findQualifier->SORT_BY_DATE_ASC))
        && (!$qValue->equals($findQualifier->SORT_BY_DATE_DESC))
        && (!$qValue>equals($findQualifier->SERVICE_SUBSET))
        && (!$qValue->equals($findQualifier->COMBINE_CATEGORY_BAGS))) 24
          throw new UnsupportedException("find_service: "."findQualifier=".$qValue); 25
       }
      } 26
    } 27
    $infoVector = null; 28
    $truncatedResults = false; 28
    $keyVector = $dataStore-> findService($businessKey, $nameVector, $categoryBag, $tModelBag,$qualifiers); 28
    if (($keyVector != null) && (sizeof($keyVector) > 0)) 28
    {
      $rowCount = sizeof($keyVector); 29
      if (($maxRows > 0) && ($maxRows < $rowCount)) 29
      {
        $rowCount = $maxRows; 30
        $truncatedResults = true; 30
      } 31

      $infoVector = new Vector($rowCount); 32
      for ($i=0; $i<$rowCount; $i++) 33
        $infoVector->addElement($dataStore->fetchServiceInfo($keyVector[$i])); 34
    } 35
    $infos = new ServiceInfos(); 36
    $infos->setServiceInfoVector($infoVector); 36
    $list = new ServiceList();36
    $list->setGeneric($generic); 36
    $list->setServiceInfos($infos); 36
    $list->setOperator($Config->getOperator());36
    $list->setTruncated($truncatedResults); 36
    return $list; 36
  } 37
```

Se debe representar el grafo de flujo asociado al algoritmo utilizando algunos componentes como nodos, aristas y regiones.

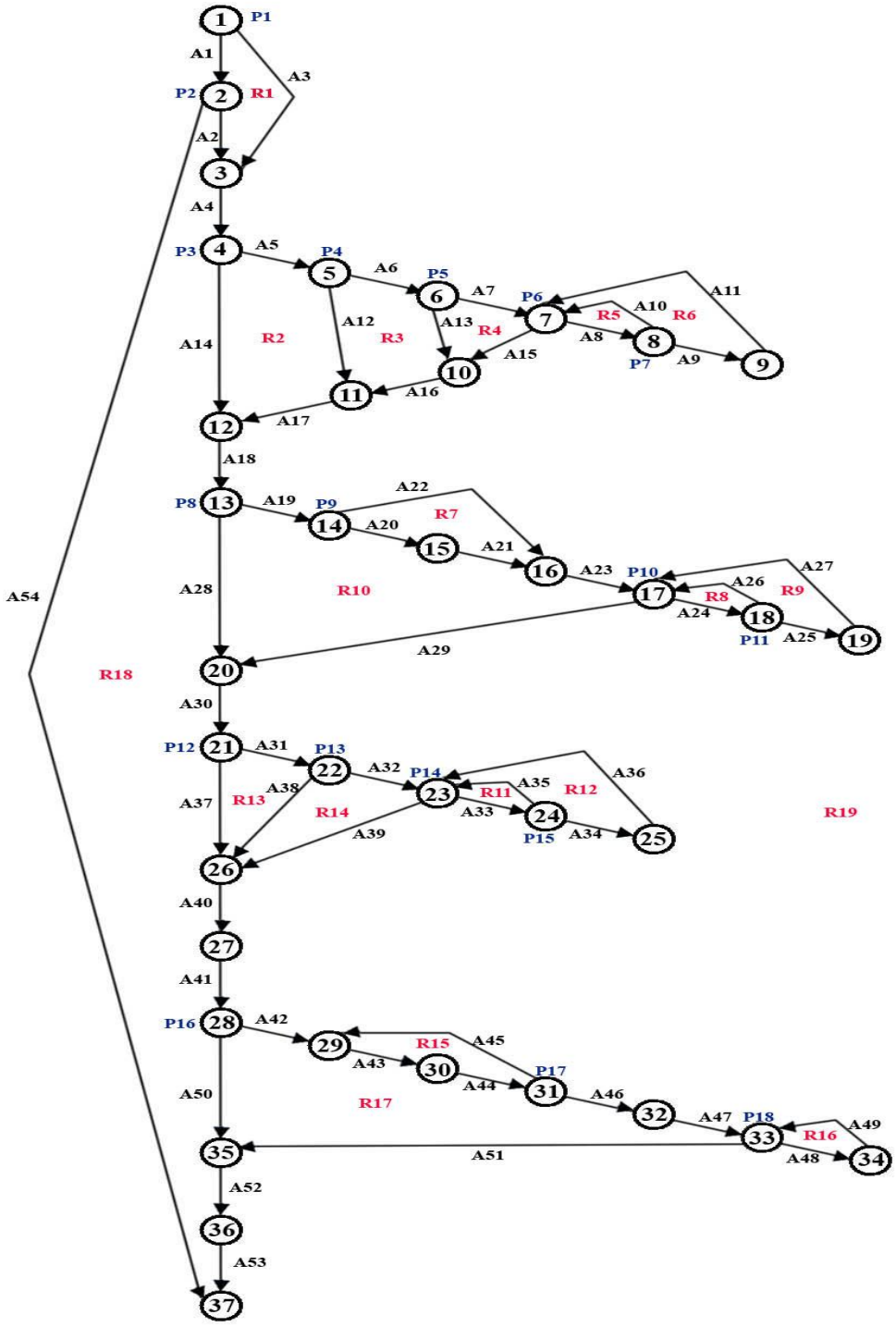


Figura # 26: Grafo de flujo del algoritmo FindBusinessFunction.

Existen tres formas fundamentales de calcular la complejidad, las cuales deben dar el mismo resultado para comprobar que el cálculo de la complejidad ha sido correcto:

1) V(Grafo) = número de regiones del grafo de flujo = 19
2) V(Grafo) = A - N + 2 = 54 – 37 + 2 = 19

Donde A es el número de aristas del grafo y N es el número de nodos.

3) V(Grafo) = P + 1 = 18 + 1 = 19

Donde P es el número de nodos predicado (nodo del cual salen varias aristas) contenidos en el grafo.

Después de realizado el cálculo se puede apreciar que existen como máximo 19 caminos lógicos por donde recorrer el algoritmo.

Son muchos lo criterios que se han definidos para los rangos de complejidad de los programas y los algoritmos, pero muchos autores lo han concebido de la siguiente forma [19]:

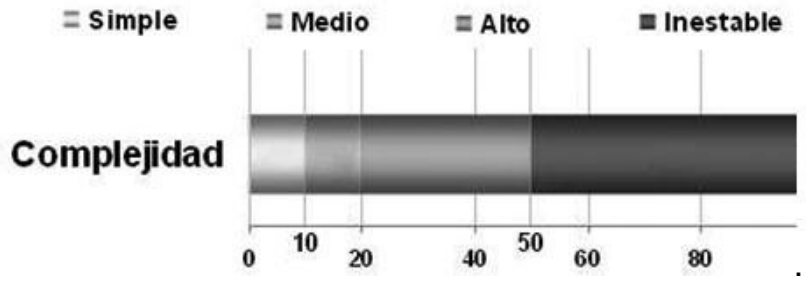


Figura # 27: Gráfica de rangos de complejidad.

A través del cálculo de la complejidad ciclomática, se puede apreciar según el valor resultante, que el algoritmo es medianamente complejo.

2.4 Descripción de clases y funcionalidades utilizadas

2.4.1 Descripción de clases generales

BusinessEntity	
Entidad	
Atributo	**Tipo**
businessKey	
authorizedName	
operator	
discoveryURLs	
nameVector	
descVector	
contacts	

businessServices	
identifierBag	
categoryBag	
Para cada responsabilidad:	
Nombre:	Descripción:
BusinessEntity()	Constructor de la Clase
setBusinessKey($key)	Este método se usa para establecer la clave de una entidad de negocio.
getBusinessKey()	Este método se usa para obtener la clave de una entidad de negocio.
setAuthorizedName($name)	Este método se usa para establecer el nombre autorizado de una entidad de negocio.
getAuthorizedName()	Este método se usa para obtener el nombre autorizado de una entidad de negocio.
setOperator($operator)	Este método se usa para establecer el operador de una entidad de negocio.
addDiscoveryURL($url)	Este método se usa para adicionar una nueva URL.
setDiscoveryURLs($urls)	Este método se usa para establecer las URLs de una entidad de negocio.
getDiscoveryURLs()	Este método se usa para obtener las URLs que tiene una entidad de negocio.
addDescription_in_BusinessEntity($descr)	Este método se usa para adicionar la descripción de una entidad de negocio.
setDescriptionVector_in_BusinessEntity($descr)	Este método se usa para establecer la descripción de una entidad de negocio.
addContact_in_BusinessEntity($contact)	Este método se usa para adicionar un contacto de una entidad de negocio.
setContacts_in_BusinessEntity($contacts)	Este método se usa para establecer los contactos de una entidad de negocio.
addName_in_BusinessEntity($name)	Este método se usa para adicionar el nombre de una entidad de negocio.
setNameVector_in_BusinessEntity($names)	Este método se usa para establecer el nombre de una entidad de negocio.
addBusinessService_in_BusinessEntity($service)	Este método se usa para adicionar un servicio Web a una entidad de negocio.
setBusinessServices_in_BusinessEntity($services)	Este método se usa para establecer los servicios Web que ofrece una entidad de negocio.
addIdentifier_in_BusinessEntity($keyref)	Este método se usa para adicionar un identificador a una entidad de negocio.
setIdentifierBag($bag)	Este método se usa para establecer el identificador de una entidad de negocio.
addCategory_in_BusinessEntity($keyref)	Este método se usa para adicionar una categoría

	a una entidad de negocio.
setCategoryBag_in_BusinessEntity($bag)	Este método se usa para establecer las categorías de una entidad de negocio.

Tabla #1: Descripción de la Clase BusinessEntity

BusinessService	
Entidad	
Atributo	**Tipo**
businessKey	
serviceKey	
nameVector	
descrVector	
bindingTemplates	
categoryBag	
Para cada responsabilidad:	
Nombre:	Descripción:
BusinessService ()	Constructor de la Clase
setBusinessKey_in_BusinessService($key)	Este método se usa para establecer la clave de la entidad de negocio a que pertenece el servicio Web.
setServiceKey_in_BusinessService($key)	Este método se usa para establecer la clave de un servicio Web.
addName_in_BusinessService($name)	Este método se usa para adicionar el nombre de un servicio Web.
addDescription_in_BusinessService($desc)	Este método se usa para adicionar la descripción de un servicio Web.
setNameVector_in_BusinessService($names)	Este método se usa para establecer el nombre de un servicio Web.
setDescriptionVector_in_BusinessService($descs)	Este método se usa para establecer la descripción de un servicio Web.
addBindingTemplate_in_BusinessService($binding)	Este método se usa para adicionar la descripción técnica de un servicio Web.
setBindingTemplates_in_BusinessService($bindings)	Este método se usa para establecer la descripción técnica de un servicio Web.
addCategory_in_BusinessService($ref)	Este método se usa para adicionar una categoría a un servicio Web.
setCategoryBag_in_BusinessService($bag)	Este método se usa para establecer la categoría de un servicio Web.

Tabla #2: Descripción de la Clase BusinessService

BindingTemplate	
Entidad	
Atributo	**Tipo**
bindingKey	
serviceKey	
descVector	
accessPoint	
hostingRedirector	
tModelInstanceDetails	
categoryBag	
Para cada responsabilidad:	
Nombre:	Descripción:
BindingTemplate ()	Constructor de la Clase
setBindingKey_in_BindingTemplate($key)	Este método se usa para establecer la clave de la descripción técnica de un servicio Web.
setServiceKey_in_BindingTemplate($key)	Este método se usa para establecer la clave de un servicio Web al que pertenece la descripción técnica.
addDescription_in_BindingTemplate($desc)	Este método se usa para adicionar los detalles de una descripción técnica.
setDescriptionVector_in_BindingTemplate($descs)	Este método se usa para establecer los detalles de una descripción técnica.
setAccessPoint_in_BindingTemplate($point)	Este método se usa para establecer el punto de acceso al servicio Web.
getAccessPoint()	Este método se usa para obtener el punto de acceso al servicio Web.
setHostingRedirector($director)	Este método se usa para establecer redireccionamiento a un servicio Web.
getHostingRedirector()	Este método se usa para obtener redireccionamiento a un servicio Web.
setTModelInstanceDetails_in_BindingTemplate($details)	Este método se usa para establecer una instancia de un modelo técnico en una descripción técnica.
addCategory_in_BindingTemplate($ref)	Este método se usa para adicionar una categoría a una descripción técnica.
setCategoryBag_in_BindingTemplate($bag)	Este método se usa para establecer la categoría de una descripción técnica.

Tabla #3: Descripción de la Clase BindingTemplate

TModel	
Entidad	
Atributo	**Tipo**
TYPES_TMODEL_KEY	
NAICS_TMODEL_KEY	
UNSPSC_TMODEL_KEY	
UNSPSC_73_TMODEL_KEY	
ISO_CH_TMODEL_KEY	
GENERAL_KEYWORDS_TMODEL_KEY	
OWNING_BUSINESS_TMODEL_KEY	
RELATIONSHIPS_TMODEL_KEY	
OPERATORS_TMODEL_KEY	
D_U_N_S_TMODEL_KEY	
THOMAS_REGISTER_TMODEL_KEY	
IS_REPLACED_BY_TMODEL_KEY	
SMTP_TMODEL_KEY	
FAX_TMODEL_KEY	
FTP_TMODEL_KEY	
TELEPHONE_TMODEL_KEY	
HTTP_TMODEL_KEY	
HOMEPAGE_TMODEL_KEY	
tModelKey	
authorizedName	
operator	
nameValue	
descVector	
overviewDoc	
identifierBag	
categoryBag	
Para cada responsabilidad:	
Nombre:	Descripción:
TModel()	Constructor de la Clase
setTModelKey_in_TModel($key)	Este método se usa para establecer la clave de un modelo técnico.
setAuthorizedName_in_TModel($name)	Este método se usa para establecer el nombre autorizado de un modelo técnico.
setName_in_TModel($name)	Este método se usa para establecer el nombre de un modelo técnico.
addDescription_in_TModel($descr)	Este método se usa para adicionar la descripción de un modelo técnico.
setDescriptionVector_in_TModel($descriptions)	Este método se usa para establecer la descripción de un modelo técnico.

setOverviewDoc_in_TModel($doc)	Este método se usa para establecer referencias a descripciones generales remotas o instrucciones relativas al modelo técnico.
setOperator_in_TModel($operator)	Este método se usa para establecer el operador de un modelo técnico.
addIdentifier_in_TModel($keyed)	Este método se usa para adicionar un identificador a un modelo técnico.
setIdentifierBag_in_TModel($bag)	Este método se usa para establecer el operador de un modelo técnico.
addCategory_in_TModel($keyedRef)	Este método se usa para adicionar una categoría a un modelo técnico.
setCategoryBag($bag)	Este método se usa para establecer la categoría de un modelo técnico.

Tabla #4: Descripción de la Clase TModel

2.4.2 Descripción de clases y funcionalidades de UDDI Inquiry API

FindBusiness	
Atributo	**Tipo**
generic	
nameVector	
identifierBag	
categoryBag	
tModelBag	
discoveryURLs	
findQualifiers	
maxRows	
Para cada responsabilidad:	
Nombre:	Descripción:
FindBusiness ()	Constructor de la Clase
setGeneric_in_FindBusiness($genericValue)	Este método se usa para establecer el valor genérico de un proveedor de servicios Web.
addName_in_FindBusiness($nameValue)	Este método se usa para adicionar el nombre de un proveedor de servicios Web.
setNameVector_in_FindBusiness($names)	Este método se usa para establecer el nombre de un proveedor de servicios Web.
addIdentifier_in_FindBusiness($ref)	Este método se usa para adicionar el identificador de un proveedor de servicios Web.
setIdentifierBag_in_FindBusiness($bag)	Este método se usa para establecer el identificador de un proveedor de servicios Web.
addCategory_in_FindBusiness($ref)	Este método se usa para adicionar una categoría de un proveedor de servicios Web.

setCategoryBag_in_FindBusiness($bag)	Este método se usa para establecer una lista de categorías de un proveedor de servicios Web.
addTModelKey_in_FindBusiness($key)	Este método se usa para adicionar una clave de un modelo técnico de un proveedor de servicios Web.
setTModelBag_in_FindBusiness($bag)	Este método se usa para establecer una lista de modelos técnicos de un proveedor de servicios Web.
addDiscoveryURL_in_FindBusiness($url)	Este método se usa para adicionar una URL de un proveedor de servicios Web.
setDiscoveryURLs_in_FindBusiness($urls)	Este método se usa para establecer una lista URLs de un proveedor de servicios Web.
setMaxRows_in_FindBusiness($maxRows)	Este método se usa para establecer el número límite de resultados.
addFindQualifier_in_FindBusiness($findQualifier)	Este método se usa para adicionar un cualificador de búsqueda.
setFindQualifiers_in_FindBusiness($findQualifiers)	Este método se usa para establecer un cualificador de búsqueda.

Tabla #5: Descripción de la Clase FindBusiness

FindBusinessHandler	
Controladora	
Atributo	**Tipo**
TAG_NAME	
maker	
Para cada responsabilidad:	
Nombre:	Descripción:
_construct($maker)	Constructor de la Clase
unmarshal($element)	Este método devuelve un objeto de tipo **FindBusiness** que ha sido modelado, según la información sacada de un elemento XML con estructura de un mensaje SOAP, que es recibido como parámetro.
marshal($object,$parent)	Este método crea un elemento XML con la información que extrae de un objeto recibido, y lo incorpora como hijo de otro elemento XML también recibido por parámetros.

Tabla #6: Descripción de la Clase FindBusinessHandler

FindBusinessFunction	
Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
_construct()	Constructor de la Clase
execute($regObject)	Este método se usa realizar una búsqueda de proveedores de servicios.

Tabla #7: Descripción de la Clase FindBusinessFunction

FindService	
Atributo	**Tipo**
generic	
businessKey	
nameVector	
categoryBag	
tModelBag	
findQualifiers	
maxRows	
Para cada responsabilidad:	
Nombre:	Descripción:
FindService()	Constructor de la Clase
setBusinessKey_in_FindService($key)	Este método se usa para establecer la clave del proveedor del servicio Web.
setGeneric_in_FindService($genericValue)	Este método se usa para establecer el valor genérico de un servicio Web.
addName_in_FindService($nameValue)	Este método se usa para adicionar el nombre de un servicio Web.
setNameVector_in_FindService($names)	Este método se usa para establecer el nombre de un servicio Web.
addCategory_in_FindService($ref)	Este método se usa para adicionar una categoría de un servicio Web.
setCategoryBag_in_FindService($bag)	Este método se usa para establecer una lista de categorías de un servicio Web.
addTModelKey_in_FindService($key)	Este método se usa para adicionar una clave de un modelo técnico de un servicio Web.
setTModelBag_in_FindService($bag)	Este método se usa para establecer una lista de modelos técnicos de un servicio Web.
setMaxRows_in_FindService($maxRows)	Este método se usa para establecer el número

	límite de resultados.
addFindQualifier_in_FindService($findQualifier)	Este método se usa para adicionar un cualificador de búsqueda.
setFindQualifiers_in_FindService($findQualifiers)	Este método se usa para establecer un cualificador de búsqueda.

Tabla #8: Descripción de la Clase FindService

BusinessServiceHandler	
Controladora	
Atributo	**Tipo**
TAG_NAME	
maker	
Para cada responsabilidad:	
Nombre:	Descripción:
_construct($maker)	Constructor de la Clase
unmarshal($element)	Este método devuelve un objeto de tipo **FindServiceHandler** que ha sido modelado, según la información sacada de un elemento XML con estructura de un mensaje SOAP, que es recibido como parámetro.
marshal($object,$parent)	Este método crea un elemento XML con la información que extrae de un objeto recibido, y lo incorpora como hijo de otro elemento XML también recibido por parámetros.

Tabla #9: Descripción de la Clase BusinessServiceHandler

FindServiceFunction	
Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
_construct ()	Constructor de la Clase
execute($regObject)	Este método se usa realizar una búsqueda de servicios Web.

Tabla #10: Descripción de la Clase FindServiceFunction

FindBinding	
Atributo	**Tipo**
serviceKey	
generic	
categoryBag	
tModelBag	
findQualifiers	
maxRows	
Para cada responsabilidad:	
Nombre:	Descripción:
FindBinding()	Constructor de la Clase
setServiceKey_in_FindBinding($key)	Este método se usa para establecer la clave del servicio Web de la unión.
setGeneric_in_FindBinding($genericValue)	Este método se usa para establecer el valor genérico de una unión.
addCategory_in_FindBinding($ref)	Este método se usa para adicionar una categoría de una unión.
setCategoryBag_in_FindBinding($bag)	Este método se usa para establecer una lista de categorías de una unión.
addTModelKey_in_FindBinding($key)	Este método se usa para adicionar una clave de un modelo técnico de una unión.
setTModelBag_in_FindBinding($bag)	Este método se usa para establecer una lista de modelos técnicos de una unión.
setMaxRows_in_FindBinding($maxRows)	Este método se usa para establecer el número límite de resultados.
addFindQualifier_in_FindBinding($findQualifier)	Este método se usa para adicionar un cualificador de búsqueda.
setFindQualifiers_in_FindBinding($findQualifiers)	Este método se usa para establecer un cualificador de búsqueda.

Tabla #11: Descripción de la Clase FindBinding

FindBindingHandler	
Controladora	
Atributo	**Tipo**
TAG_NAME	
maker	
Para cada responsabilidad:	
Nombre:	Descripción:
_construct($maker)	Constructor de la Clase

unmarshal($element)	Este método devuelve un objeto de tipo **FindBinding** que ha sido modelado, según la información sacada de un elemento XML con estructura de un mensaje SOAP, que es recibido como parámetro.
marshal($object,$parent)	Este método crea un elemento XML con la información que extrae de un objeto recibido, y lo incorpora como hijo de otro elemento XML también recibido por parámetros.

Tabla #12: Descripción de la Clase FindBindingHandler

FindBindingFunction	
Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
_construct ()	Constructor de la Clase
execute($regObject)	Este método se usa realizar una búsqueda de descripciones técnicas.

Tabla #13: Descripción de la Clase FindBindingFunction

FindTModel	
Atributo	**Tipo**
generic	
name	
identifierBag	
categoryBag	
findQualifiers	
maxRows	
Para cada responsabilidad:	
Nombre:	Descripción:
FindTModel()	Constructor de la Clase
setGeneric_in_FindTModel($genericValue)	Este método se usa para establecer el valor genérico de un modelo técnico.
setName_in_FindTModel()	Este método se usa para establecer el nombre de un modelo técnico.
getNameString_in_FindTModel()	Este método se usa para obtener el nombre de un modelo técnico.
addIdentifier_in_FindTModel($ref)	Este método se usa para adicionar un identificador de un modelo técnico de un modelo técnico.

setIdentifierBag_in_FindTModel($bag)	Este método se usa para establecer una lista de identificadores de un modelo técnico.
addCategory_in_FindTModel($ref)	Este método se usa para adicionar una categoría de un modelo técnico de un modelo técnico.
setCategoryBag_in_FindTModel($bag)	Este método se usa para establecer una lista de categorías de un modelo técnico.
setMaxRows_in_FindTModel($maxRows)	Este método se usa para establecer el número límite de resultados.
addFindQualifier_in_FindTModel($findQualifier)	Este método se usa para adicionar un cualificador de búsqueda de un modelo técnico de un modelo técnico.
setFindQualifiers_in_FindTModel($qualifiers)	Este método se usa para establecer una lista de cualificadores de búsqueda de un modelo técnico.

Tabla #14: Descripción de la Clase FindTModel

FindTModelHandler	
Controladora	
Atributo	**Tipo**
TAG_NAME	
maker	
Para cada responsabilidad:	
Nombre:	Descripción:
_construct($maker)	Constructor de la Clase
unmarshal($element)	Este método devuelve un objeto de tipo **FindTModel** que ha sido modelado, según la información sacada de un elemento XML con estructura de un mensaje SOAP, que es recibido como parámetro.
marshal($object,$parent)	Este método crea un elemento XML con la información que extrae de un objeto recibido, y lo incorpora como hijo de otro elemento XML también recibido por parámetros.

Tabla #15: Descripción de la Clase FindTModelHandler

FindTModelFunction	
Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
_construct ()	Constructor de la Clase

execute($regObject)	Este método se usa realizar una búsqueda de modelos técnicos.

Tabla #16: Descripción de la Clase FindTModelFunction

FindRelatedBusinesses	
Atributo	**Tipo**
businessKey	
generic	
keyedReference	
findQualifiers	
maxRows	
Para cada responsabilidad:	
Nombre:	Descripción:
FindRelatedBusinesses ()	Constructor de la Clase
setBusinessKey_in_FindRelatedBusinesses($key)	Este método se usa para establecer la clave de los proveedores relacionados.
setGeneric_in_FindRelatedBusinesses($genericValue)	Este método se usa para establecer el valor genérico de los proveedores relacionados.
setKeyedReference_in_FindRelatedBusinesses($keyedRef)	Este método se usa para establecer la descripción de la relación de proveedores.
setMaxRows_in_FindRelatedBusinesses($maxRows)	Este método se usa para establecer el número límite de resultados.
addFindQualifier_in_FindRelatedBusinesses($findQualifier)	Este método se usa para adicionar un cualificador de búsqueda.
setFindQualifiers_in_FindRelatedBusinesses($findQualifiers)	Este método se usa para establecer una lista de cualificadores.

Tabla #17: Descripción de la Clase FindRelatedBusinesses

FindRelatedBusinessesHandler	
Controladora	
Atributo	**Tipo**
TAG_NAME	
maker	
Para cada responsabilidad:	
Nombre:	Descripción:

_construct($maker)	Constructor de la Clase
unmarshal($element)	Este método devuelve un objeto de tipo **FindRelatedBusinesses** que ha sido modelado, según la información sacada de un elemento XML con estructura de un mensaje SOAP, que es recibido como parámetro.
marshal($object,$parent)	Este método crea un elemento XML con la información que extrae de un objeto recibido, y lo incorpora como hijo de otro elemento XML también recibido por parámetros.

Tabla #18: Descripción de la Clase FindRelatedBusinessesHandler

FindRelatedBusinessesFunction	
Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
_construct ()	Constructor de la Clase
execute($regObject)	Este método se usa realizar una búsqueda de relaciones entre proveedores de servicios Web.

Tabla #19: Descripción de la Clase FindRelatedBusinessesFunction

GetBusinessDetail	
Atributo	**Tipo**
generic	
businessKeyVector	
Para cada responsabilidad:	
Nombre:	Descripción:
GetBusinessDetail()	Constructor de la Clase
setGeneric_in_GetBusinessDetail($genericValue)	Este método se usa para establecer el valor genérico de un proveedor.
addBusinessKey_in_GetBusinessDetail($key)	Este método se usa para adicionar la clave de un proveedor.
setBusinessKeyVector_in_GetBusinessDetail($keys)	Este método se usa para establecer una lista de claves de proveedores.

Tabla #20: Descripción de la Clase GetBusinessDetail

GetBusinessDetailHandler	
Controladora	
Atributo	**Tipo**
TAG_NAME	
maker	
Para cada responsabilidad:	
Nombre:	Descripción:
_construct ()	Constructor de la Clase
unmarshal($element)	Este método devuelve un objeto de tipo **GetBusinessDetail** que ha sido modelado, según la información sacada de un elemento XML con estructura de un mensaje SOAP, que es recibido como parámetro.
marshal($object, $parent)	Este método crea un elemento XML con la información que extrae de un objeto recibido, y lo incorpora como hijo de otro elemento XML también recibido por parámetros.

Tabla #21: Descripción de la Clase GetBusinessDetailHandler

GetBusinessDetailFunction	
Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
_construct ()	Constructor de la Clase
execute($regObject)	Este método se usa mostrar los datos detallados de un proveedor de servicios.

Tabla #22: Descripción de la Clase GetBusinessDetailFunction

GetServiceDetail	
Atributo	**Tipo**
generic	
serviceKeyVector	
Para cada responsabilidad:	
Nombre:	Descripción:
GetServiceDetail()	Constructor de la Clase
setGeneric_in_GetServiceDetail($genericValue)	Este método se usa para establecer el valor genérico de un servicio Web.

addServiceKey_in_GetServiceDetail()	Este método se usa para adicionar la clave de un servicio Web.
setServiceKeyVector_in_GetServiceDetail($keys)	Este método se usa para establecer una lista de claves de servicios Web.

Tabla #23: Descripción de la Clase GetServiceDetail

GetServiceDetailHandler	
Controladora	
Atributo	**Tipo**
TAG_NAME	
maker	
Para cada responsabilidad:	
Nombre:	Descripción:
_construct ()	Constructor de la Clase
unmarshal($element)	Este método devuelve un objeto de tipo **GetServiceDetail** que ha sido modelado, según la información sacada de un elemento XML con estructura de un mensaje SOAP, que es recibido como parámetro.
marshal($object, $parent)	Este método crea un elemento XML con la información que extrae de un objeto recibido, y lo incorpora como hijo de otro elemento XML también recibido por parámetros.

Tabla #24: Descripción de la Clase GetServiceDetailHandler

GetServiceDetailFunction	
Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
_construct ()	Constructor de la Clase
execute($regObject)	Este método se usa mostrar los datos detallados de un servicio Web.

Tabla #25: Descripción de la Clase GetServiceDetailFunction

GetBindingDetail	
Atributo	**Tipo**
generic	

bindingKeyVector	
Para cada responsabilidad:	
Nombre:	Descripción:
GetBindingDetail()	Constructor de la Clase
setGeneric_in_GetBindingDetail($genericValue)	Este método se usa para establecer el valor genérico de una unión.
addBindingKey_in_GetBindingDetail()	Este método se usa para adicionar la clave de una unión.
setBindingKeyVector_in_GetBindingDetail($keys)	Este método se usa para establecer una lista de claves de una unión.
getServiceKeyVector()	Este método se usa para obtener una lista de claves de una unión.

Tabla #26: Descripción de la Clase GetBindingDetail

GetBindingDetailHandler	
Controladora	
Atributo	**Tipo**
TAG_NAME	
maker	
Para cada responsabilidad:	
Nombre:	Descripción:
_construct ()	Constructor de la Clase
unmarshal($element)	Este método devuelve un objeto de tipo **GetBindingDetail** que ha sido modelado, según la información sacada de un elemento XML con estructura de un mensaje SOAP, que es recibido como parámetro.
marshal($object, $parent)	Este método crea un elemento XML con la información que extrae de un objeto recibido, y lo incorpora como hijo de otro elemento XML también recibido por parámetros.

Tabla #27: Descripción de la Clase GetBindingDetailHandler

GetBindingDetailFunction	
Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
_construct ()	Constructor de la Clase

execute($regObject)	Este método se usa mostrar los datos detallados de una descripción técnica.

Tabla #28: Descripción de la Clase GetBindingDetailFunction

GetTModelDetail	
Atributo	**Tipo**
generic	
tModelKeyVector	
Para cada responsabilidad:	
Nombre:	Descripción:
GetTModelDetail()	Constructor de la Clase
setGeneric_in_GetTModelDetail($genericValue)	Este método se usa para establecer el valor genérico de un modelo técnico.
addTModelKey_in_GetTModelDetail()	Este método se usa para adicionar la clave de un modelo técnico.
setTModelKeyVector_in_GetTModelDetail($keys)	Este método se usa para establecer una lista de claves de un modelo técnico.

Tabla #29: Descripción de la Clase GetTModelDetail

GetTModelDetailHandler	
Controladora	
Atributo	**Tipo**
TAG_NAME	
maker	
Para cada responsabilidad:	
Nombre:	Descripción:
_construct ($maker)	Constructor de la Clase
unmarshal($element)	Este método devuelve un objeto de tipo **GetTModelDetail** que ha sido modelado, según la información sacada de un elemento XML con estructura de un mensaje SOAP, que es recibido como parámetro.
marshal($object, $parent)	Este método crea un elemento XML con la información que extrae de un objeto recibido, y lo incorpora como hijo de otro elemento XML también recibido por parámetros.

Tabla #30: Descripción de la Clase GetTModelDetailHandler

GetTModelDetailFunction	
Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
_construct ()	Constructor de la Clase
execute($regObject)	Este método se usa mostrar los datos detallados de un modelo técnico.

Tabla #31: Descripción de la Clase GetTModelDetailFunction

GetBusinessDetailExt	
Atributo	**Tipo**
generic	
businessKeyVector	
Para cada responsabilidad:	
Nombre:	Descripción:
GetBusinessDetailExt()	Constructor de la Clase
setGeneric_in_GetBusinessDetailExt($genericValue)	Este método se usa para establecer el valor genérico de un proveedor de servicios Web extendido.
addBusinessKey_in_GetBusinessDetailExt()	Este método se usa para adicionar la clave de un proveedor de servicios Web extendido.
setBusinessKeyVector_in_GetBusinessDetailExt($keys)	Este método se usa para establecer una lista de claves de un proveedor de servicios Web extendido.

Tabla #32: Descripción de la Clase GetBusinessDetailExt

GetBusinessDetailExtHandler	
Controladora	
Atributo	**Tipo**
TAG_NAME	
maker	
Para cada responsabilidad:	
Nombre:	Descripción:
_construct ($maker)	Constructor de la Clase
unmarshal($element)	Este método devuelve un objeto de tipo

	GetBusinessDetailExt que ha sido modelado, según la información sacada de un elemento XML con estructura de un mensaje SOAP, que es recibido como parámetro.
marshal($object, $parent)	Este método crea un elemento XML con la información que extrae de un objeto recibido, y lo incorpora como hijo de otro elemento XML también recibido por parámetros.

Tabla #33: Descripción de la Clase GetBusinessDetailExtHandler

GetBusinessDetailExtFunction	
Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
_construct ()	Constructor de la Clase
execute($regObject)	Este método se usa mostrar los datos detallados de un proveedor de servicios extendido.

Tabla #34: Descripción de la Clase GetBusinessDetailExtFunction

2.4.3 Descripción de clases y funcionalidades de UDDI Security API

GetAuthToken	
Atributo	**Tipo**
generic	
userID	
credential	
Para cada responsabilidad:	
Nombre:	Descripción:
GetAuthToken()	Constructor de la Clase
setGeneric_in_GetAuthToken($genericValue)	Este método se usa para establecer el valor genérico.
setUserID_in_GetAuthToken($idValue)	Este método se usa para establecer el identificador de un usuario.
setCredential($credValue)	Este método se usa para establecer la credencial de un usuario.
getCredential()	Este método se usa para obtener la credencial de un usuario.

Tabla #35: Descripción de la Clase GetAuthToken

GetAuthTokenHandler	
Controladora	
Atributo	**Tipo**
TAG_NAME	
maker	
Para cada responsabilidad:	
Nombre:	Descripción:
_construct ($maker)	Constructor de la Clase
unmarshal($element)	Este método devuelve un objeto de tipo **GetAuthToken** que ha sido modelado, según la información sacada de un elemento XML con estructura de un mensaje SOAP, que es recibido como parámetro.
marshal($object, $parent)	Este método crea un elemento XML con la información que extrae de un objeto recibido, y lo incorpora como hijo de otro elemento XML también recibido por parámetros.

Tabla #36: Descripción de la Clase GetAuthTokenHandler

GetAuthTokenFunction	
Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
_construct ()	Constructor de la Clase
execute($regObject)	Este método se usa para obtener un token de autenticación.

Tabla #37: Descripción de la Clase GetAuthTokenFunction

DiscardAuthToken	
Atributo	**Tipo**
generic	
authInfo	
Para cada responsabilidad:	
Nombre:	Descripción:
DiscardAuthToken()	Constructor de la Clase
setGeneric_in_DiscardAuthToken($genericValue)	Este método se usa para establecer el valor genérico.
setAuthInfo_in_DiscardAuthToken($info)	Este método se usa para establecer la información

	de autenticación.

Tabla #38: Descripción de la Clase DiscardAuthToken

DiscardAuthTokenHandler	
Controladora	
Atributo	**Tipo**
TAG_NAME	
maker	
Para cada responsabilidad:	
Nombre:	Descripción:
_construct ()	Constructor de la Clase
unmarshal($element)	Este método devuelve un objeto de tipo **DiscardAuthToken** que ha sido modelado, según la información sacada de un elemento XML con estructura de un mensaje SOAP, que es recibido como parámetro.
marshal($object, $parent)	Este método crea un elemento XML con la información que extrae de un objeto recibido, y lo incorpora como hijo de otro elemento XML también recibido por parámetros.

Tabla #39: Descripción de la Clase DiscardAuthTokenHandler

DiscardAuthTokenFunction	
Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
_construct ()	Constructor de la Clase
execute($regObject)	Este método se usa para descartar un token de autenticación.

Tabla #40: Descripción de la Clase DiscardAuthTokenFunction

2.5 Conclusiones

Este capítulo ha sido de gran importancia debido a que se ha hecho un profundo estudio del diseño que se ha seguido para la implementación de las APIs, sobre el que se puede decir que está muy bien pensado y estructurado, además de estar claramente explicado en las especificaciones de UDDI 3.0.2. Se han descritos las principales clases y funcionalidades de las mismas, las cuales permiten que esta parte del sistema funcione correctamente. El análisis de uno de los algoritmos no triviales permite tener una visión de la complejidad de algunos procedimientos de los que dependen grandemente los objetivos del registro UDDI.

Capítulo 3. Validación de la solución propuesta

3.1 Introducción

Una etapa vital en el desarrollo del software son las pruebas, guiadas fundamentalmente al descubrimiento de errores que se puedan encontrar o bien para verificar y revelar la calidad de un producto de software. En este capítulo se exponen una serie de pruebas que se les han realizado a las APIs de búsqueda, seguridad y suscripción de UDDI, con el objetivo de mostrar la viabilidad de la solución propuesta. El tipo de prueba presentado es la prueba de caja blanca, uno de los tipos principales y más factibles.

3.2 Descripción de las pruebas

Las pruebas tienen como objetivo detectar y documentar problemas de software, validar que este trabaje como fue establecido y que los requisitos fueron implementados correctamente, todo esto antes de que se pase a la siguiente fase y de esta forma prevenir la afectación del usuario final. Se concibe una buena prueba cuando esta descubre errores que no han sido encontrados durante la etapa de desarrollo. Deben realizarse sistemáticamente, esto posibilita una mejor comprobación de software y por tanto, mejores resultados. También determinan la calidad y fiabilidad del software, en el caso de encontrar errores simples se puede adjudicar que el proceso llevado a cabo es aceptable, de lo contrario, se cuestiona el procedimiento.

Para un mejor cumplimiento del propósito adjudicado a las pruebas se determinan una serie de actividades que representan la guía de desarrollo [20]:

- Definir la misión de evaluación, en la que se identifica el método de prueba adecuado a la situación presente y se determina como se hará el monitoreo y la evaluación del proceso.
- Verificación del enfoque de prueba, en la que se logra el entendimiento de las restricciones y las limitaciones de cada técnica de prueba para el caso de estudio presente y así mitigar el riesgo de encontrar una técnica falible cuando ya el proceso se encuentre un punto clave.
- Validar la estabilidad del build, en esta se definen detalles de prueba, se implementa se ejecuta y por último se determinan los fallos.
- Prueba y evaluación, en la que se realizan las pruebas a lo ancho y en profundidad para permitir una evaluación suficiente de los elementos que son objetivo de estas.

- Mejoramiento de la calidad de las pruebas, en la que su función fundamental es mantener y mejorar las evaluaciones.

Las pruebas tienen una cierta conformación jerárquica debido a que su constitución conceptual muestra argumentos como métodos, tipos, niveles y estrategias de prueba que se relacionan entre sí.

En el caso de los tipos de prueba cada uno tiene un objetivo específico y una técnica que lo soporte. Dentro de estos se encuentran la funcionalidad, usabilidad, fiabilidad, rendimiento y soportabilidad.

Como las pruebas son aplicadas para diferentes tipos de objetivos y en diferentes escenarios se determinan los niveles de pruebas conocidos como de unidad, integración, sistema y de aceptación.

Los casos de pruebas son como un mecanismo de ayuda para asegurar el completamiento de las pruebas y lograr una probabilidad de descubrimiento de errores mayor, es decir, una búsqueda más exhaustiva de defectos del producto.

En la estrategia de prueba se describe el enfoque y los objetivos generales de las actividades de prueba y debe seguir un procedimiento de planificación, diseño y ejecución de estas. Dentro de sus funcionalidades define las técnicas de prueba y las herramientas a usar, los criterios de éxito y la culminación de las evaluaciones y las consideraciones especiales relacionadas con los recursos necesarios para realizar las pruebas.

Las pruebas de unidad son realizables a módulos del software, estas se simplifican cuando se diseña con un alto grado de cohesión. Procede a comprobar las interfaces de usuarios, la estructura de los datos locales, las condiciones límites, los caminos independientes y los caminos de manejo de errores. Dentro de esta podemos encontrar dos fases fundamentales, la fase conocida como informal previa encargada de ejecutar pequeños ejemplos y la fase de prueba sistemática que contiene las técnicas de pruebas de caja negra centrada a la comprobación de las especificaciones funcionales y las técnicas de prueba de caja blanca que requieren para su aplicación del conocimiento de la estructura interna del programa derivadas a partir de las especificaciones internas del diseño o del código.

Como la implementación propuesta son códigos que solo representan una parte del producto, las pruebas que se aplicaran son las unitarias y dentro de esta el técnica de caja blanca.

Las pruebas de caja blanca realizan un minucioso análisis de los detalles procedimentales del software, proporcionan casos de prueba que garanticen que se ejercite por lo menos una vez todos los caminos independientes para cada módulo, todas las decisiones lógicas en sus vertientes verdaderas y falsas y el aseguramiento de la validez de las estructuras internas del producto. Esta técnica consta de métodos

consistentes en pruebas de camino básico que brinda una medida de la complejidad lógica del diseño procedimental definiendo el conjunto de caminos básicos de ejecución, la prueba de flujo de datos o matrices de grafos que selecciona los caminos de prueba de acuerdo con la ubicación de las definiciones y los usos de las variables del programa, las prueba de condición que ejercita las condiciones lógicas contenidas en el módulo de un programa y la de bucles que se centra exclusivamente en la validez de la construcción de bucles. Estas pruebas controlan la estructura y no la funcionalidad.

3.3 Aplicación de pruebas de caja blanca

Para la realización de las pruebas de caja blanca a un determinado algoritmo es necesario efectuar anteriormente el análisis de la complejidad ciclomática del mismo, el cual nos permite definir el número de caminos independientes del algoritmo y nos da el límite superior del número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez.

Para calcular la complejidad ciclomática del procedimiento es necesario seguir los pasos descritos en el capítulo 2, epígrafe 2.3.

En este caso se ha seleccionado un algoritmo muy importante para el funcionamiento de UDDI, que ha sido descrito en el capítulo 2, epígrafe 2.2.2.1 “Funcionalidades de Inquiry API”, el **GetBusinessDetail**.

```
function execute($regObject)
{
 $request = $regObject; 1
 $generic = $request->getGeneric(); 1
 $businessKeyVector = $request->getBusinessKeyVector(); 1
 $dataStore = new JDBCDataStore(); 1

   for ($i=0; $i<sizeof($businessKeyVector); $i++)2
  {
     $businessKey = $businessKeyVector[$i]; 3
     if (($businessKey == null) || (strlen($businessKey) == 0) ||(!$dataStore->isValidBusinessKey($businessKey))) 3
           throw new InvalidKeyPassedException("get_businessDetail: "."businessKey=".$businessKey); 4
  }
$businessVector = new  Vector(); 5
for ($i=0; $i<sizeof($businessKeyVector); $i++) 6
{
```

```
  $businessKey = $businessKeyVector[$i]; 7
  $businessVector[] = $dataStore->fetchBusiness($businessKey); 7
 }
 $dataStore->commit(); 8
 $detail = new BusinessDetail(); 8
 $detail->setGeneric($generic);  8
 $detail->setBusinessEntityVector($businessVector);  8
 return $detail;  8
} 9
```

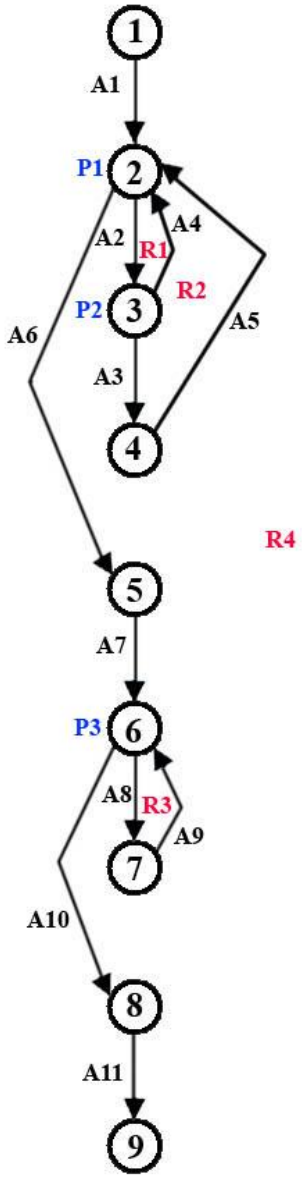


Figura # 28: Grafo de flujo del código de GetServiceDetailFunction.

Ya representado el grafo de flujo, se realiza el cálculo de complejidad ciclomática mediante sus tres formas.

1) V(Grafo) = número de regiones del grafo de flujo = 4

2) V(Grafo) = A - N + 2 = 11 – 9 + 2 = 4
3) V(Grafo) = P + 1 = 3 + 1 = 4

Después de realizado el cálculo de complejidad se puede apreciar que ha resultado 4, igual en las tres formas de calcularse. Ahora se puede decir que se tiene el valor del límite de la cantidad de pruebas que se deben realizar para comprobar que se ejecute cada sentencia del código al menos una vez.

Caminos independientes:

1) 1-2-5-6-8-9
2) 1-2-**3**-5-6-8-9
3) 1-2-**3-4**-5-6-8-9
4) 1-2-3-5-6-**7**-8-9

Los casos de pruebas del algoritmo deben ser ejecutados al finalizar la extracción de los caminos independientes, de los cuales se debe sacar al menos un caso de prueba por cada uno de ellos.

Para la realización de los casos de pruebas es necesario seguir ciertos requerimientos como:

- **Descripción:** Se describe la entrada necesaria de datos, teniendo en cuenta que no se pasen parámetros erróneos.
- **Condición de ejecución:** Se especifican los parámetros con las condiciones deseadas para verificar la efectividad del procedimiento.
- **Entrada:** Se precisan los parámetros de entrada.
- **Resultados esperados:** Se plasman los resultados previstos.

Caso de prueba del primer camino independiente:

Descripción: El parámetro de entrada recibido es un objeto de tipo GetBusinessDetail.

Condición de ejecución: El objeto GetBusinessDetail contiene un atributo llamado businessKeyVector que debe estar vacío.

Entrada:

$request = $regObject; (*a la variable $request se le asigna el objeto de tipo GetBusinessDetail pasado por parámetros*).

$businessKeyVector = $request->getBusinessKeyVector(); (*a la variable $businessKeyVector se le asigna el vector que contiene las claves de los proveedores, en este caso estará vacío, por lo que la longitud es igual 0*).
sizeof($businessKeyVector) =0 (*longitud del vector*).
Resultados esperados: No debe devolver nada.

Caso de prueba del segundo camino independiente:
Descripción: El parámetro de entrada recibido es el mismo que en el primer caso de prueba.
Condición de ejecución: El valor de los $businessKey almacenados en el $businessKeyVector no deben ser nulos, o no vacíos o deben ser válidos.
Entrada:
$request = $regObject; (*a la variable $request se le asigna el objeto de tipo GetBusinessDetail pasado por parámetros*).
$businessKeyVector = $request->getBusinessKeyVector(); (*a la variable $businessKeyVector se le asigna el vector que contiene las claves de los proveedores*).
$ businessKeyVector [$i] = 5252544 (*el valor de la variable $businessKey en la posición i es válido*).
Resultados esperados: Debe mostrar los datos detallados del proveedor especificado.

Caso de prueba del tercer camino independiente:
Descripción: El parámetro de entrada recibido es el mismo que en el primer caso de prueba.
Condición de ejecución: El valor de los $businessKey almacenados en el $businessKeyVector deben ser nulos, vacíos o deben ser no válidos.
Entrada:
$request = $regObject; (*a la variable $request se le asigna el objeto de tipo GetBusinessDetail pasado por parámetros*).
$businessKeyVector = $request->getBusinessKeyVector(); (*a la variable $businessKeyVector se le asigna el vector que contiene las claves de los proveedores*).
$ businessKeyVector [$i] = null (*el valor de la variable $businessKey en la posición i es nulo*).
Resultados esperados: Debe lanzar un error de tipo E_INVALID_KEY_PASSED.

Caso de prueba del cuarto camino independiente:

Descripción: El parámetro de entrada recibido es el mismo que en el primer caso de prueba.

Condición de ejecución: El objeto GetBusinessDetail contiene un atributo llamado businessKeyVector que no debe estar vacío.

Entrada:

$request = $regObject; (*a la variable $request se le asigna el objeto de tipo GetBusinessDetail pasado por parámetros*).

$businessKeyVector = $request->getBusinessKeyVector(); (*a la variable $businessKeyVector se le asigna el vector que contiene las claves de los proveedores*).

sizeof($businessKeyVector) = # (*longitud del vector distinto de 0*).

Resultados esperados: Debe mostrar los datos detallados del proveedor especificado.

3.4 Conclusiones

En este capítulo se ha abordado acerca de una herramienta que juega un papel fundamental en la calidad de un software, como son las pruebas. Se han llevado a cabo una serie de pruebas realizadas a las APIs implementadas, por lo que han sido desarrolladas de manera apropiada a la implementación llevada a cabo.

Conclusiones

El uso de los servicios Web es una problemática en todo el mundo y definir herramientas y reglamentaciones para eliminar su irracional utilización es primordial para el desarrollo de las TIC. A lo largo de este trabajo se han expuesto una serie de pautas importantes enfocadas en mejorar el empleo de los servicios, brindando vías de acceso al desarrollo informático.

De forma general se le dio cumplimiento a las tareas propuestas, alcanzando satisfactoriamente el desempeño requerido por los objetivos propuestos:

- Se logró estudiar, comprender y describir las especificaciones internacionales implantadas sobre el diseño de las APIs de Universal Description Discovery and Integration.
- Se logró implementar las APIs propuestas.
- Se realizó el análisis del funcionamiento de algoritmos no triviales implementados en la solución.
- Se efectuó la validación de la solución propuesta mediante algunas pruebas de software llevadas a cabo.

Recomendaciones

Para alcanzar resultados satisfactorios en el empleo de UDDI se recomienda:

- Hacer un buen uso de las APIs implementadas.
- El estudio y la implementación de otras APIs no desarrolladas tendiendo como marco de referencias las especificaciones establecidas para UDDI.

Referencias Bibliográficas

[1] BRAVO, P. *Web Service*, Junio, 2007. [Disponible en: http://sistemas3.wordpress.com/2007/06/11/web-service/

[2] GONZÁLEZ, B. *UDDI (Universal Description Discovery and Integration)*. Disponible en: http://www.desarrolloweb.com/articulos/1589.php

[3] MARTÍNEZ, L. V. *Estudio UDDI*. Matemática - Computación Ciudad de La Habana, Universidad de La Habana, Junio, 2004. 83. p.

[4] VÁZQUEZ, E. C. and D. T. SIVILA. *Diseño de un catálogo de servicios Web basado en las especificaciones y normas de la UDDI para la plataforma de informatización en la UCI.*: Dirección de Informatización. Ciudad de La Habana, Universidad de las Ciencias Informáticas, Junio, 2007. 154. p.

[5] *SOA (Service-oriented Architecture) Arquitectura Orientada a Servicios*. Noviembre, 2007 [Disponible en: http://www.wadooa.com/doku.php/soa

[6] MARTÍNEZ, P. E. *Extensibilidad de UDDI*. Lenguajes y Sistemas Informáticos. Sevilla, Universidad de Sevilla, Septiembre, 2007. 83. p.

[7] OASIS. *UDDI Version 3.0.2*
UDDI Spec Technical Committee Draft, Octubre, 2004.

[8] *SOAP*. Marzo, 2008. [Disponible en: http://es.wikipedia.org/wiki/SOAP

[9] MEDINA, R. C. *Presentación de WSDL*, 2005. [Disponible en: http://almacen.gulic.org/diveintopython-5.4-es/soap_web_services/index.html

[10] PIÑOL, C. M. I. *WSDL*, 2007. [Disponible en: http://www.uoc.edu/masters/esp/img/873.pdf

[11] REY, E. M. *Programación Orientada a Objetos*. Departamento de Computación. España, Universidad de Coruña: 41. p.

[12] *Programación lógica*. Noviembre, 2007. [Disponible en: http://es.wikipedia.org/wiki/Programaci%C3%B3n_l%C3%B3gica

[13] *GLOSARIO DE TERMINOS*. 2008. [Disponible en: http://www.inei.gob.pe/biblioineipub/bancopub/Inf/Lib5084/GLOSA.HTM

[14] *PHP*. Marzo, 2008. [Disponible en: http://es.wikipedia.org/wiki/PHP

[15] *Zend Studio Enterprise Edition version 5.5* Marzo, 2008. [Disponible en: http://foro-pirate.com/foro/descarga-directas/zend-studio-enterprise-edition-version-5-5/

[16] CORPORATION, Z. *Zend Studio 5 I LAS SOLUCIONES MÁS COMPLETAS PARA EL DESARROLLO DE PHP*, 2007.

[17] CUENCA, C. L. *Arquitectura del servidor Apache*, 2007. [Disponible en: http://www.desarrolloweb.com/manuales/41/

[18] KABIR, M. *La Biblia del Servidor Apache 2.0*. 2003. 845 p. 8441514682

[19] RIZZI, F. M. *COMPLEJIDAD CICLOMÁTICA*, 2004.

[20] INFORMÁTICAS, U. D. L. C. *Entorno Virtual de Aprendizaje*. Disponible en: http://teleformacion.uci.cu/

Bibliografía

El Servidor Apache y Programas de Soporte. 2008. [Disponible en: http://httpd.apache.org/docs/2.0/

Guía Breve de Servicios Web. 2008. [Disponible en: http://www.w3c.es/divulgacion/guiasbreves/ServiciosWeb

Informe sobre redes neuronales. 2008. [Disponible en: http://www.monografias.com/Computacion/Redes/

Programación Lógica. Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga, Curso 2003-2004. 25. p.

AGUILERA, G. *SOA: Flexibilidad y agilidad al servicio del negocio*, FEBRERO 08, 2008 [Disponible en: http://biblioteca.dgsca.unam.mx/cu/productos/boletines/msg00005.html

OLIVENCIA., J. L. L. *Diseño de Algoritmos*. Departamento Lenguajes y Ciencias de la Computación. España, Universidad de Málaga, Curso 2004-2005. p.

QUINTERO, A. M. R. *Visión General de la Programación Orientada a Aspectos*. Departamento de Lenguajes y Sistemas Informáticos. Sevilla, Universidad de Sevilla, Diciembre, 2000. Diciembre, 2000. p.

REY, E. M. *Laboratorio de Investigación y desarrollo en Inteligencia Artificial*. Departamento de Computación. España, Universidade da Coruña. p.

SOA (Service-oriented Architecture) Arquitectura Orientada a Servicios. Noviembre, 2007 [Disponible en: http://www.wadooa.com/doku.php/soa

OASIS. *UDDI Version 3.0.2*
UDDI Spec Technical Committee Draft, Octubre, 2004.

Zend Studio Enterprise Edition version 5.5 Marzo, 2008. [Disponible en: http://foro-pirate.com/foro/descarga-directas/zend-studio-enterprise-edition-version-5-5/

Glosario de Términos

A

API: Interfaz de Programación de Aplicaciones. Es el conjunto de funciones y métodos que ofrece cierta librería para ser utilizado por otro software como una capa de abstracción. Una API representa un interfaz de comunicación entre componentes software. Se trata del conjunto de llamadas a ciertas librerías que ofrecen acceso a ciertos servicios desde los procesos y representa un método para conseguir abstracción en la programación.

Álgebra Relacional: Es un conjunto de operaciones que describen paso a paso como computar una respuesta sobre las relaciones, tal y como éstas son definidas en el modelo relacional. Denominada de tipo **Procedimental**, a diferencia del Cálculo relacional que es de tipo declarativo.

ASP.NET (**Active Server Pages**): Es una tecnología del lado servidor de Microsoft para páginas web generadas dinámicamente, que ha sido comercializada como un anexo a Internet Information Server (IIS).

C

Cálculo Relacional: Es un lenguaje de consulta que describe la respuesta deseada sobre una Base de datos sin especificar como obtenerla, a diferencia del Álgebra relacional que es de tipo procedural, el cálculo relacional es de **tipo declarativo**; pero siempre ambos métodos logran los mismos resultados.

C++ (pronunciado *"ce más más"* o *"ce plus plus"*): Es un lenguaje de programación, como extensión del lenguaje de programación C, abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos. Además está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel.

C# (pronunciado ***"si sharp"***): Es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET; lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma.

ColdFusion: Lenguaje de programación, puede crear y modificar variables igual que en otros lenguajes de programación.

Common Gateway Interface (**CGI**): Tecnología de la World Wide Web que permite a un cliente solicitar datos de un programa ejecutado en un servidor Web.

CORBA: En computación, (*Common Object Request Broker Architecture* — arquitectura común de intermediarios en peticiones a objetos), es un estándar que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos.

D

DB2: Es una **marca comercial**, propiedad de IBM, bajo la cual se comercializa el sistema de gestión de base de datos. Es un motor de base de datos relacional que integra XML de manera nativa, además

permite almacenar documentos completos dentro del tipo de datos xml para realizar operaciones y búsquedas de manera jerárquica dentro de éste, e integrarlo con búsquedas relacionales.

Definición Dirigida: Una definición dirigida por la sintaxis es un pariente cercano de los esquemas de traducción.
Delphi: Es un entorno de desarrollo de software diseñado para la programación de propósito general con énfasis en la programación visual. En Delphi se utiliza como lenguaje de programación una versión moderna de Pascal llamada Object Pascal. En sus diferentes variantes, permite producir archivos ejecutables para Windows, Linux y la plataforma .NET.

Depuración: Es el proceso de identificar y corregir errores de programación.

Documentos PDF: Es un formato de almacenamiento de documentos, desarrollado por la empresa Adobe Systems. Está especialmente ideado para documentos susceptibles de ser impresos, ya que especifica toda la información necesaria para la presentación final del documento.

E

Entorno de Desarrollo Integrado (IDE): Programa compuesto por un conjunto de herramientas para un programador.

Estándar abierto: Es una especificación disponible públicamente para lograr una tarea específica. La especificación debe haber sido desarrollada en proceso abierto a toda la industria y también debe garantizar que cualquiera la puede usar sin necesidad de pagar regalías o rendir condiciones a ningún otro.

F

Firebird: Es una base de datos relacional que ofrece muchas características de SQL ANSI estándar y que funciona en Linux, Windows, MacOSX y una variedad de plataformas UNIX.

Firewall: Un elemento utilizado en redes de computadoras para controlar las comunicaciones, permitiéndolas o prohibiéndolas.

Form Interpreter: Sistema para procesar formularios.

FTP: File Transfer Protocol. Es un protocolo de transferencia de ficheros entre sistemas conectados a una red TCP basado en la arquitectura cliente-servidor, de manera que desde un equipo cliente nos podemos conectar a un servidor para descargar ficheros desde él o para enviarle nuestros propios archivos independientemente del sistema operativo utilizado en cada equipo.

G

GNU/Linux: Es un sistema operativo tipo Unix que se distribuye bajo la Licencia Pública General de GNU o GPL, es decir que es software libre.

H

HTTP (**HyperText Transfer Protocol**): Protocolo de transferencia de hipertexto es el protocolo usado en cada transacción de la Web (WWW). El hipertexto es el contenido de las páginas web, y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceso a una página y la respuesta con el contenido.

HTML (**HyperText Markup Language**)**:** Lenguaje usado para escribir documentos para servidores World Wide Web. Es una aplicación de la ISO Standard 8879:1986. Es un lenguaje de marcas. Los lenguajes de marcas no son equivalentes a los lenguajes de programación aunque se definan igualmente como "lenguajes". Son sistemas complejos de descripción de información, normalmente documentos, que se pueden controlar desde cualquier editor ASCII.

Host Virtual: Se refiere a la práctica de mantener más de un servidor en una sola máquina, así como diferenciarlos por el nombre de servidor que presentan.

I

Inteligencia Artificial: Es la rama de la informática que desarrolla procesos que imitan a la inteligencia de los seres vivos. La principal aplicación de esta ciencia es la creación de máquinas para la automatización de tareas que requieran un comportamiento inteligente.

J

Java: Es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Java Server Pages (**JSP**): Tecnología Java que permite generar contenido dinámico para Web, en forma de documentos HTML, XML o de otro tipo.

L

Legacy Systems: Es un sistema viejo de la computadora o programa de aplicación que sigue usado porque el usuario (típicamente una organización) no quiere reemplazar o rediseñarlo.

Lenguaje Script: Es un lenguaje de programación que fue diseñado para ser ejecutado por medio de un intérprete, en contraste con los lenguajes compilados.

Lenguaje Práctico para la Extracción e Informe (**Perl**): Lenguaje de programación que toma características del C, del lenguaje interpretado shell (sh) y en un grado inferior, muchos otros lenguajes de programación.

Licencia Pública General de GNU: Es una licencia creada por la Free Software Foundation, y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

Logs: Un log es usado para registrar datos o información sobre quién, qué, cuándo, dónde y por qué un evento ocurre para un dispositivo en particular o aplicación.

M

Macromedia: es una empresa de software de gráficos y desarrollo web con centrales en San Francisco, California.

Macintosh (***Mac***): Es el nombre con el que actualmente nos referimos a cualquier computadora personal diseñada, desarrollada, construida y comercializada por Apple Inc.

Manejo de Excepciones: Es una estructura de control de los lenguajes de programación diseñada para manejar condiciones anormales que pueden ser tratadas por el mismo programa que se desarrolla.

Microsoft SQL Server: Es un sistema de gestión de bases de datos relacionales (SGBD) basado en el lenguaje Transact-SQL, y específicamente en Sybase IQ, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea.

Microsoft ASP: Tecnología del lado del servidor para páginas Web generadas dinámicamente. Lanzada por Microsoft.

MySQL: Es un sistema de gestión de bases de datos relacional que cuentan con todas las características de un motor de BD comercial: transacciones atómicas, triggers, replicación, llaves foráneas entre otras. Su ingeniosa arquitectura lo hace extremadamente rápido y fácil de personalizar.

O

OASIS: Acrónimo de (Organization for the Advancement of Structured Information Standards) es un consorcio internacional sin fines de lucro que orienta el desarrollo, la convergencia y la adopción de los estándares sobre comercio electrónico. El consorcio crea estándares abiertos en los ámbitos de servicios Web, seguridad, comercio-electrónico y estandarización en el sector público, así como para mercados de aplicaciones específicas.

ODBC (**Open Database Connectivity**)**:** Es un estándar de acceso a Bases de Datos desarrollado por Microsoft Corporation, el objetivo de *ODBC* es hacer posible el acceder a cualquier dato desde cualquier aplicación, sin importar qué Sistema Gestor de Bases de Datos almacene los datos,

Oracle: Es un sistema de gestión de base de datos relacional, fabricado por Oracle Corporation.

P

Personal Home Page Tools: PHP (Herramientas para Páginas Iniciales Personales)
Lenguaje de programación tipo script para entornos Web utilizado sobre todo en servidores Linux para personalizar la información que se envía a los usuarios que acceden a un sitio web. Es un programa de software libre con unas funciones muy semejantes a las de ASP Y JSP.

Python: Es un lenguaje de programación interpretado, y ahorra un tiempo considerable en el desarrollo del programa, pues no es necesario compilar ni enlazar. El intérprete se puede utilizar de modo interactivo, lo que facilita experimentar con características del lenguaje, escribir programas desechables o probar funciones durante el desarrollo del programa.

Postgres SQL: Es un servidor de base de datos relacional orientada a objetos de software libre, liberado bajo la licencia BSD.

S

Servicios Web: Es una colección de protocolos y estándares que sirve para intercambiar datos entre aplicaciones.

Sun Microsystems: Es una empresa informática de Silicon Valley, fabricante de semiconductores y software.

SFTP (**Security File Transfer Protocol**)**:** Es un protocolo de red que proporciona la transferencia de archivos y la funcionalidad de manipulación de datos fiables sobre cualquier secuencia ya que los datos circulan cifrados por la red.

SSL (**Secure Sockets Layer**)**:** Proporciona autenticación y privacidad de la información entre extremos sobre Internet mediante el uso de criptografía.

SQLite: Es un sistema de gestión de bases de datos relacional compatible con ACID, y que está contenida en una relativamente pequeña (~500kb) librería en C.

U

URL (**Uniform Resource Locator**): Es un localizador uniforme de recurso. Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización.

V

Visual Basic .NET (**VB.NET**): Es un lenguaje de programación orientado a objetos que se puede considerar una evolución de Visual Basic implementada sobre el framework .NET.

W

Windows: Es una familia de sistemas operativos desarrollados y comercializados por Microsoft, con distintas versiones para hogares, empresas, servidores y dispositivos móviles.