ESCUELA POLITÉCNICA DEL EJÉRCITO

FACULTAD DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

DESARROLLO DE UNA APLICACIÓN "WEB SERVICES PILOTO" PARA DINERS CLUB DEL ECUADOR USANDO ADVANTAGE PLEX Y WEBSYDIAN

Previa la obtención del Título de:

INGENIERO EN SISTEMAS E INFORMÁTICA

Por: **Christian Benalcázar Lagos**
Lorena Cañizares Zúñiga

Sangolquí, 06 de Enero del 2005

INDICE DE CONTENIDOS

LISTADO DE TABLAS

LISTADO DE CUADROS

LISTADO DE FIGURAS

RESUMEN

El presente documento trata de la investigación e implementación de un Web Service con la herramienta Advantage Plex y Websydian para Diners Club del Ecuador

Se comienza a realizar un estudio general de los Web Services, así como los elementos que conllevan al mismo, su estructura y normas. Para fines prácticos se ha desarrollado la implementación de un Web Service piloto para Diners Club del Ecuador en la herramienta Advantage Plex y Websydian, basado en el intercambio de información entre plataformas y sistemas diferentes, bajo el formato XML.

La investigación de los Web Services y la implementación bajo la herramienta Plex / Websydian, ha sido documentada paso a paso a fin de que el lector pueda entender de mejor manera acerca de este tema. La construcción del Web Services se basa en el modelo propio del Plex y su metodología, donde el Web Service es el encargado de interpretar y recuperar la información perteneciente a los movimientos hechos por un Tarjetahabiente determinado; la petición se la hace mediante el cliente browser quien envía los parámetros bajo los cuales el Web Services ha sido diseñado.

Esta tesis concluye con la puesta en práctica del sistema piloto, para lo cual se ha desarrollado el cliente (sistema independiente) en la herramienta Punto Net de Microsoft; el cliente será el que invocará al Web Service para verificar el intercambio de información entre diferentes tipos de lenguajes de desarrollo.

SUMMARY

This document is about the research and implementation of a Web Service for Diners Club del Ecuador using the Advantage Plex and Websydian tools.

It begins with the general investigation about Web Services, their elements, structure and rules. With the purpose of demonstrate their functionality we have developed a small prototype to Diners Club del Ecuador with the Advantage Plex and Websydian, based on the information interchange between disparate platforms and systems all under the XML format.

The investigation about Web Services and their implementation under the Plex and Websydian tools has been documented step by step so the reader can understand in a better way about the Web Services topic. The Web Services development is based on the own Plex methodology, where the Web Services has the duty of the interpretation and reload the information which belongs to the transactional movements that a determinate card holder has done . The request is done thought the browser client who sends the parameters under the design of the Web Service.

This thesis finishes putting the prototype in deployment, for this reason the client has been developed independently in the Microsoft Dot Net tool. The client will be the one who request to the Web Services to verify the information interchange between disparate development languages.

CAPÍTULO I

INTRODUCCIÓN

En las empresas hoy en día, la información es uno de los activos más valiosos que poseen cada una, las tecnologías siguen evolucionando de forma impactante, las empresas acumulan cantidades cada vez más grandes de datos en una gran variedad de formatos y en diferentes plataformas y aplicaciones. Para hacer compatibles los modelos eBusiness que aprovechan estas nuevas tecnologías, las empresas deben crear e integrar aplicaciones, procesos y sistemas en todo su concepto, obteniendo datos de diversas fuentes, y proporcionando a las personas un mejor servicio.

En el Ecuador actualmente el desarrollo de Web Services no ha sido explotado a gran nivel debido a la complejidad que implica el entendimiento de los términos y tecnología nueva que éste involucra; mientras que en el mercado internacional ya existen algunas empresas que están haciendo uso de las ventajas que brindan los Web Services.

La globalización ha creado e incrementado un ambiente de negocio complejo y dinámico que obliga a las compañías a enfocarse en su núcleo competitivo y a tercerizar las áreas más importantes de su cadena, donde no poseen ventajas sustancialmente competitivas. Esto incrementa la demanda de comunicación e interoperatividad de negocios en tiempo real entre Sistemas de Información no compatibles.

Los Sistemas de Información no compatibles han sido hasta cierto punto la tarea más dificultosa causante de no poder brindar información en línea; tal es el caso de Diners Club del Ecuador que en la actualidad necesita que la información de los estados de cuenta a nivel de sus clientes se encuentre actualizada a todo momento para lo cual debe proveer de esta información a su tercerizadora Web quien es la que presenta al usuario final los datos requeridos a través del Internet; sin embargo esto hoy no sucede ya que la actualización de la de la misma se lo realiza a través de medios magnéticos que por sí solos hacen que el proceso de mostrarlo en línea no sea el deseado, causando así el descontento de los clientes que hacen uso del Internet para realizar sus consultas y observar que la información no se encuentra actualizada a la fecha.

Mediante los Web Services, las empresas como Diners podrán compartir servicios de software con sus clientes y sus socios de negocio. Esto ayudará a Diners a escalar sus negocios, reduciendo el coste en desarrollo y mantenimiento de software, y sacando los productos al mercado con mayor rapidez.

La integración de aplicaciones hará posible obtener la información requerida en línea, acelerando el proceso de decisiones. La evolución de Internet hacia los Web Services, mejorará los resultados globales de las empresas, reduciendo sus gastos y guiándolas hacia una mejora progresiva de la calidad.

La investigación y creación de un Web Service basado en la herramienta Advantage Plex y Websydian nace de la necesidad de explotar estas herramientas de última generación adaptables al desarrollo de nuevas tecnologías Web y al de solucionar a futuro el intercambio

de información entre sistemas no compatibles como lo es Diners Club del Ecuador y empresas tercerizadoras Web.

1.1 Alcance

- Desarrollar un Web Service Piloto utilizando la herramienta Websydian con Advantage Plex, el cual proporcionará información en línea del estado de cuenta Diners al Tarjetahabiente (1).
- El Plan Piloto está orientado al intercambio de información en tiempo real en base a una petición que la hará el subscritor (empresa cliente) hacia el publicador (propietario del Web Services).
- El tema de seguridad no será abarcada en su contexto, ya que son conceptos muy amplios y se tomarán puntos generales para discusiones en lo posterior.

1.2 Objetivos

1.2.1 General

Desarrollar un Web Service Piloto de consulta del estado de cuenta de la tarjeta de crédito Diners, usando la herramienta Websydian y Advantage Plex.

1.2.2 Específicos

- Analizar las características, estructura, estándares y construcción de los Web Services.
- Investigar la arquitectura, configuración y requerimientos del Advantage Plex y Websydian para poder realizar un Web Service.
- Implementar el Web Service Piloto.

Tarjetahabiente (1): Persona natural dueña de una tarjeta de crédito.

1.3 Justificación

Actualmente Diners Club del Ecuador ofrece a sus clientes la posibilidad de consultar sus estados de cuenta a través del Internet; servicio dado por una empresa tercerizadora con alto nivel en el manejo de seguridad de datos; la actualización de los datos se la realiza mediante el envio de información cada cierto tiempo y en medio magnético debido a la diferencia de plataformas y lenguajes que las dos partes manejan.

Esto provoca que la información mostrada al tarjetahabiente no sea la información en línea ni actual causando el descontento especialmente de los clientes que dependen de la información real de sus movimientos.

Diners siempre preocupado de brindar el mejor servicio a sus clientes, ha visto la necesidad de adquirir nuevas herramientas que se adapten a las nuevas tecnologías Web para poder cubrir los inconvenientes antes mencionados.

Teniendo esto como precedente, Diners Club ha realizado la inversión en la compra de herramientas de última generación, sin embargo no se ha explotado el potencial que brindan por no existir expertos en el tema tanto de Web Services como el del manejo de la herramienta Websydian y Advantage Plex; esta investigación es inminente para la empresa ya que será la base para realizar futuras aplicaciones Web tanto para la empresa como para los desarrolladores, los cuales reutilizarán las funcionalidades existentes, y la empresa tendrá la ventaja de poder realizar negocios con terceros sin limitaciones en cuanto a plataformas o lenguajes.

Se propone la creación de un Web Services Piloto basado en la herramienta Advantage Plex y Websydian que nace de la necesidad de explotar nuevas tecnologías adaptables al desarrollo y de solucionar a futuro el intercambio de información entre sistemas no compatibles como lo es Diners Club y su Tercerizadora Web.

El objetivo de realizar el desarrollo de este prototipo es el de satisfacer las necesidades que tiene Diners en cuanto a poner en línea la información mediante la utilización de Web Services con las herramientas ya adquiridas; de tal forma que puedan cubrir los requerimientos Web que la empresa necesita actualmente y a futuro para satisfacer a sus clientes.

CAPÍTULO II

MARCO TEÓRICO

2.1 XML

2.1.1 Visión general de XML (Extensible Markup Language)

El código HTML (Hypertext Markup Language) permite insertar menús, tablas, imágenes o información proveniente de bases de datos en los documentos, pero no permite al usuario que maneje esos elementos como mejor le convenga con la poderosa ayuda del ordenador. Esa es la principal novedad que XML aporta.

Con HTML se pueden hacer accesos a información comparativa por ejemplo en diferentes tiendas, pero no podrá hacer nada más; sin embargo con XML el usuario podrá ordenar los datos o actualizarlos en tiempo real o realizar un pedido; es decir el usuario podrá interactuar de forma directa con la información.

La información que manejan las empresas es uno de sus principales activos. Normalmente esta información se encuentra fragmentada en diferentes departamentos y ordenadores. El reto ahora está en interrelacionar toda esta información y sacar todo su potencial, y ponerlo a trabajar para aumentar los beneficios o reducir los costos. Para realizar esto se necesita un estándar de almacenamiento estructurado que es lo que nos ofrece XML.

2.1.2 Definición

XML (Extensible Markup Language) es un conjunto de reglas para definir etiquetas semánticas que nos organizan un documento en diferentes partes. XML es un metalenguaje que define la sintaxis utilizada para definir otros lenguajes de etiquetas estructurados.

En teoría HTML (HyperText Markup Language) es un subconjunto de XML especializado en presentación de documentos para la Web, mientras que XML es un subconjunto de SGML (Standard Generalized Markup Language) [(1)] especializado en la gestión de información para la Web. En la práctica XML contiene a HTML aunque no en su totalidad. La definición de HTML contenido totalmente dentro de XML y por lo tanto que cumple a rajatabla la especificación SGML es XHTML (Extensible, Hypertext Markup Language).

2.1.3 Historia y objetivos

XML fue creado al amparo del World Wide Web Consortium (W3C) organismo que vela por el desarrollo de WWW partiendo de las amplias especificaciones de SGML. Su desarrollo se comenzó en 1996 y la primera versión salió el 10 de febrero de 1998. La primera definición que apareció fue: *"Sistema para defini, validar y compartir formatos de documentos en la Web"*.

Durante el año 1998 XML tuvo un crecimiento exponencial, y con ello nos referimos a sus apariciones en medios de comunicación, menciones en páginas Web, soporte software, etc. Sus objetivos son:

- XML debe ser directamente utilizable sobre Internet.

SGML (1): (Standardized Generalized Markup Language) Estándar internacional para la definición de métodos de representación de texto en formato electrónico

- XML debe soportar una amplia variedad de aplicaciones.
- XML debe ser compatible con SGML.
- Debe ser fácil la escritura de programas que procesen documentos XML.
- Los documentos XML deben ser legibles por humanos y razonablemente claros.
- El diseño de XML debe ser preparado rápidamente.
- El diseño de XML debe ser formal y conciso.
- Los documentos XML deben ser fácilmente creables.
- La concisión en las marcas XML es de mínima importancia.

Esta especificación, junto con los estándares asociados (Unicode e ISO/IEC 10646 para caracteres, Internet RFC 1766 para identificación de lenguajes, ISO 639 para códigos de nombres de lenguajes, e ISO 3166 para códigos de nombres de países), proporciona toda la información necesaria para entender la Versión 1.0 de XML y construir programas de computador que los procesen.

2.1.4 Principales características

- Es una arquitectura más abierta y extensible. No se necesita versiones para que puedan funcionar en futuros navegadores. Los identificadores pueden crearse de manera simple y ser adaptados en el acto en internet/intranet por medio de un validador de documentos (parser).
- Mayor consistencia, homogeneidad y amplitud de los identificadores descriptivos del documento con XML (los RDF Resource Description FrameWork), en comparación a los atributos de la etiqueta <META> del HTML.

- Integración de los datos de las fuentes más dispares. Se podrá hacer el intercambio de documentos entre las aplicaciones tanto en el propio PC como en una red local o extensa.
- Datos compuestos de múltiples aplicaciones. La extensibilidad y flexibilidad de este lenguaje nos permitirá agrupar una variedad amplia de aplicaciones, desde páginas web hasta bases de datos.
- Gestión y manipulación de los datos desde el propio cliente web.
- Los motores de búsqueda devolverán respuestas más adecuadas y precisas, ya que la codificación del contenido web en XML consigue que la estructura de la información resulte más accesible.
- Se desarrollarán de manera extensible las búsquedas personalizables y subjetivas para robots y agentes inteligentes. También conllevará a que los clientes web puedan ser más autónomos para desarrollar tareas que actualmente se ejecutan en el servidor.
- Se permitirá un comportamiento más estable y actualizable de las aplicaciones web, incluyendo enlaces bidireccionales y almacenados de forma externa.
- El concepto de "hipertexto" se desarrollará ampliamente, permitirá denominación independiente de la ubicación, enlaces bidireccionales, enlaces que pueden especificarse y gestionarse desde fuera del documento, hiperenlaces múltiples, enlaces agrupados, atributos para los enlaces, etc. Creado a través del Lenguaje de enlaces extensible (XLL).
- Exportabilidad a otros formatos de publicación (papel, web, CD-Rom, etc.). El documento maestro de la edición electrónica podría ser un documento XML que se integraría en el formato deseado de manera directa.

En la tabla 2.1 se detalla las diferencias significativas con respecto a los otros lenguajes que se ha mencionado.

Tabla 2.1: (Cuadro comparativo de lenguajes Web)

	HTML/DHTML	**XML**	**SGML**
Gramática	Fija y no ampliable	Extensible	Extensible
Estructura	Monolítica	Jerárquica	Jerárquica
Nº de marcas	Fijas	Sin límite	Sin límite
Complejidad	Baja	Mediana	Alta
Diseño de páginas	Fijado por tags. Etiquetas con atributos CSS en DHTML	CSS o XSL	DSSSL(1)
Enlaces	Simples enlaces	Poderosos enlaces (XLL)	HyTime
Exportabilidad (formatos/aplicaciones)	No	Sí	Sí
Validación	Sin validación	Pueden validarse	Obligatorio DTD
Búsquedas	Simple y a veces resuelta por *scripts* o CGI	Potente búsqueda. Con capacidad para personalizarla	Son posibles potentes búsquedas.
Indización/Catalogación de páginas web	Sólo lo permite los atributos de la etiqueta <META>, e implementaciones como DC.	Una descripción abierta y personalizable con el RDF.	Algún proyecto como TEI, DLI, etc.

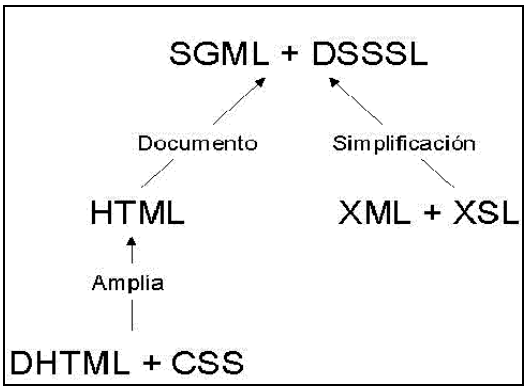


Fig. 2.1: (Esquema de relaciones entre lenguajes)

DSSSL (1): Document Style Semantics and Specification Language, ISO/IEC 10179

2.1.5 Semántica de XML

XML establece que se puede estructurar la información en un árbol. Es decir imaginar como por ejemplo a una receta de cocina como un componente, que a su vez esta formado de componentes, y así sucesivamente. Cada componente podría tener texto y/o más componentes. Una posible estructura sería imaginar que la receta tiene un componente llamado "necesitamos". No todo el texto estaría dentro de "necesitamos", solamente aquellas cosas que el cocinero de la receta necesitaría para llevarla a cabo. Dentro podríamos tener uno o más componentes llamados "ingrediente". Veamos como se ve esto (usando ya la sintaxis de XML).

Cuadro 2.1: (Estructura de etiquetas)

```
<receta>
...
<necesitamos>
  <ingrediente>2 cucharadas de azúcar</ingrediente>
  <ingrediente>3 manzanas</ingrediente>
</necesitamos>
```

Es simplemente encerrar al texto que pertenece a un componente entre <componente> y </componente>. En definitiva son "tags".

Para escribir documentos XML bien formados, hay que seguir unas reglas básicas:

- Solo puede haber un elemento raíz
- Toda etiqueta abierta hay que cerrarla: <etiqueta vacia/>

- Es sensitivo (mayúsculas y minúsculas), etiquetas correspondientes se tienen que escribir igual.
- No se pueden intercalar etiquetas:

  <libro><pagina>Applied XML <libro>12<pagina>
- Una “Tag” (etiqueta) puede tener atributos cerrados entre comillas:

  <nombre estado=”casado” hijos=”3”>Juan</nombre>
- El nombre de las etiquetas empiezan con una letra , o con uno o mas signos de puntuación : <Nombre>; <nombre>; <!Entity>; <?Ejecuta>
- Los comentarios van encerrados entre: <!-comentario ->

2.1.6 Estructura de XML

El *metalenguaje* XML consta de cuatro especificaciones (el propio XML sienta las bases sintácticas y el alcance de su implementación).

2.1.6.1 DTD (Document Type Definition)

Definición del tipo de documento. Es, en general, un archivo/s que encierra una definición formal de un tipo de documento y, a la vez, especifica la estructura lógica de cada documento. Define tanto los elementos de una página como sus atributos. El DTD del XML es opcional. En tareas sencillas no es necesario construir una DTD, entonces se trataría de un documento "bien formado"(*well-formed*) y si lleva DTD será un documento "validado" (*valid*).

2.1.6.2 XSL (eXtensible Stylesheet Language)

Define o implementa el lenguaje de estilo de los documentos escritos para XML. Desde el verano de 1997 varias empresas informáticas como Arbortext, Microsoft e Inso vienen trabajando en una propuesta de XSL (antes llamado "xml-style") que presentaron a W3C. Permite modificar el aspecto de un documento. Se puede lograr múltiples columnas, texto girado, orden de visualización de los datos de una tabla, múltiples tipos de letra con amplia variedad en los tamaños. Este estándar está basado en el lenguaje de semántica y especificación de estilo de documento (DSSSL, *Document Style Semantics and Specification Language*, ISO/IEC 10179) y, por otro lado, se considera más potente que las hojas de estilo en cascada (CSS, *Cascading Style Sheets*), usado en un principio con el lenguaje DHTML. Se espera que el CSS sea usado para visualizar simples estructuras de documentos XML (actualmente se ha conseguido mayor integración en XML con el protocolo CSS2 (*Cascading Style Sheets, level 2*) ofreciendo nuevas formas de composición y una más rápida visualización) y, por otra parte, XSL pueda ser utilizado donde se requiera más potencia de diseño como documentos XML que encierran datos estructurados (tablas, organigramas, etc.).

2.1.6.3 XLL (eXtensible Linking Language)

Define el modo de enlace entre diferentes enlaces. Se considera que es un subconjunto de HyTime (*Hipermedia/Timed-based structuring Language* o Lenguaje de estructuración hipermedia/basado en el tiempo, ISO 10744) y sigue algunas especificaciones del TEI (*Text Encoding Initiative* o Iniciativa de codificación de texto). Desde marzo de 1998 el W3C trabajó en los enlaces y direccionamientos del XML. Provisionalmente se le renombró como *Xlink* y a partir de junio se le denomina XLL. Este lenguaje de enlaces extensible tiene dos importantes componentes: *Xlink* y el *Xpointer*. Va más allá de los enlaces simples que sólo

soporta el HTML y se podrá implementar con enlaces extendidos. Jon Bosak establece los siguientes mecanismos hipertextuales que soportará esta especificación:

- Denominación independiente de la ubicación.
- Enlaces que pueden ser también bidireccionales.
- Enlaces que pueden especificarse y gestionarse desde fuera del documento a los que se apliquen (Esto permitirá crear en un entorno intranet/extranet un banco de datos de enlaces en los que se puede gestionar y actualizar automáticamente. No habrá más errores del tipo "404 Not Found").
- Hiperenlaces múltiples (anillos, múltiples ventanas, etc.).
- Enlaces agrupados (múltiples orígenes).
- Transclusión (el documento destino al que apunta el enlace aparece como parte integrante del documento origen del enlace).
- Se pueden aplicar atributos a los enlaces (tipos de enlaces).

2.1.6.4 XUA (XML User Agent)

Es la estandarización de navegadores XML. Todavía está en proceso de creación de borradores de trabajo. Se aplicará a los navegadores para que compartan todas las especificaciones XML.

2.2 WEB SERVICES

Los Web Services son componentes de software que permiten a los usuarios usar aplicaciones de negocio que comparten datos con otros programas modulares, vía Internet. Son aplicaciones independientes de la plataforma que pueden ser fácilmente publicadas, localizadas e invocadas mediante protocolos web estándar, como XML, SOAP, UDDI o WSDL. El objetivo final es el de intercambiar información entre plataformas y lenguajes no compatibles.

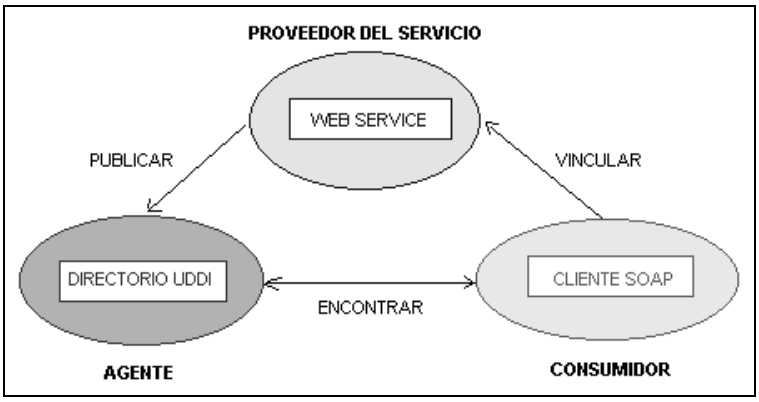


Fig. 2.2: (Proceso del Web Service)

2.2.1 Requisitos de un Web Service

- **Interoperabilidad:** Un servicio remoto debe permitir su utilización por clientes de otras plataformas.

- **Amigabilidad con Internet:** La solución debe poder funcionar para soportar clientes que accedan a los servicios remotos desde Internet.

- **Interfaces fuertemente ligadas:** No debería haber ambigüedad acerca del tipo de dato enviado y recibido desde un servicio remoto. Más aún, los tipos de datos definidos en

el servicio remoto deben poderse corresponder razonablemente bien con los tipos de datos de la mayoría de los lenguajes de programación procedimentales.

- **Posibilidad de aprovechar los estándares de Internet existentes:** La implementación del servicio remoto debería aprovechar estándares de Internet existentes tanto como sea posible y evitar reinventar soluciones a problema que ya se han resuelto. Una solución construida sobre un estándar de Internet ampliamente adoptado puede aprovechar conjuntos de herramientas y productos existentes creados para dicha tecnología.

- **Soporte para cualquier lenguaje:** La solución no debería ligarse a un lenguaje de programación particular Java RMI, por ejemplo, esta ligada completamente a lenguaje Java. Sería muy difícil invocar funcionalidad de un objeto Java remoto desde Visual Basic o PERL. Un cliente debería ser capaz de implementar un nuevo Web Service existente independientemente del lenguaje de programación en el que se halla escrito el cliente.

- **Soporte para cualquier infraestructura de componente distribuida:** La solución no debe estar fuertemente ligada a una infraestructura de componentes en particular. De hecho, no se bebería requerir el comprar, instalar o mantener una infraestructura de objetos distribuidos, solo construir un nuevo servicio remoto utilizando un servicio existente. Los protocolos subyacentes deberían proporcionar un nivel base de comunicación entre infraestructura de objeto distribuidos existentes tales como DCOM y CORBA.

2.2.2 Bloques Constructivos de Web Services

En el siguiente grafico se muestran los bloques constructivos principales necesarios para facilitar las comunicaciones remotas entre diferentes plataformas y lenguajes.

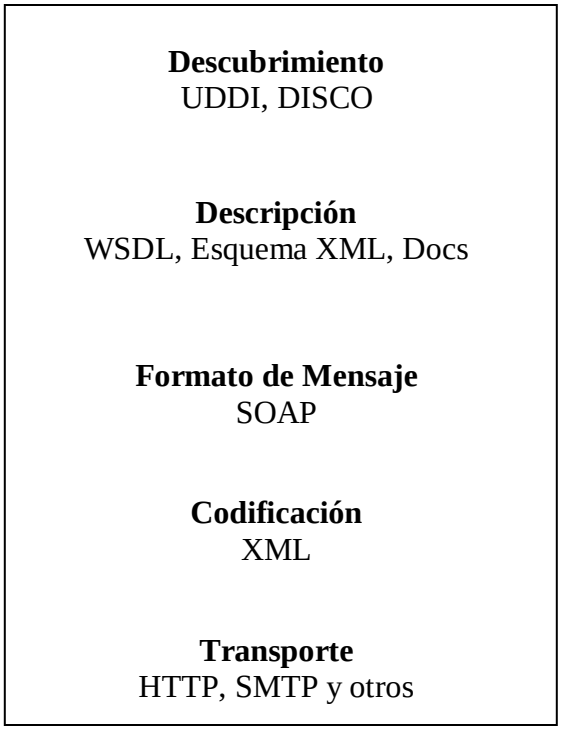


Fig. 2.3: (Bloque Constructivo del Web Service)

- **Descubrimiento:** La aplicación cliente que necesita acceder a la funcionalidad que expone un Web Service necesita una forma de resolver la ubicación de servicio remoto. Se logra mediante un proceso llamado, normalmente descubrimiento (discovery). El descubrimiento se puede proporcionar mediante un directorio centralizado así como por otros métodos. En DCOM, el servicio de descubrimiento lo proporciona el Administrador de control de servicios (SCM, Services Control Manager).

- **Descripción:** Una vez que se ha resuelto el extremo de un Web Service dado, el cliente necesita suficiente información para interactuar adecuadamente con el mismo. La descripción de un Web Service implica meta datos estructurados sobre la interfaz que intenta utilizar la aplicación cliente así como documentación escrita sobre el Web Service incluyendo ejemplo de uso. Un componente DCOM expone meta datos estructurados sobre sus interfaces mediante una biblioteca de tipo (typelib). Los meta datos dentro de una typelib de componente se guardan en un formato binario propietario a los que se accede mediante una interfaz de programación de aplicación (API) propietaria.

- **Formato del mensaje**: Para el intercambio de datos, el cliente y el servidor tienen que estar de acuerdo en un mecanismo común de codificación y formato de mensaje. El uso de un mecanismo estándar de codificar los datos asegura que los datos que codifica el cliente los interpretará correctamente el servidor. En DCOM los mensajes que se envían entre un cliente y un servidor tienen un formato definido por el protocolo DCOM Object RPC (ORPC).

- **Codificación:** Los datos que se trasmiten entre el cliente y el servidor necesitan codificarse en un cuerpo de mensaje. Dcom utiliza un esquema de codificación binaria para serializar los datos de los parámetros que se intercambian entre el cliente y el servidor.

- **Transporte:** Una vez se ha dado formato al mensaje y se han serializado los datos en el cuerpo del mensaje se debe transferir entre el cliente y el servidor utilizando algún

protocolo de transporte. DCOM dispone de varios protocolos propietarios como TCP, SPX, NetBEUI y NetBIOS sobre IPX.

2.2.3 Estándares

Un Servicio Web es un componente de software con las siguientes características:

- Es accesible a través de la interfase **SOAP** (Simple Object Access Protocol).
- Su interfase se describe en un documento **WSDL** (Web Service Description Language).
- Se realiza la Publicación de Servicios en un directorio **UDDI**.

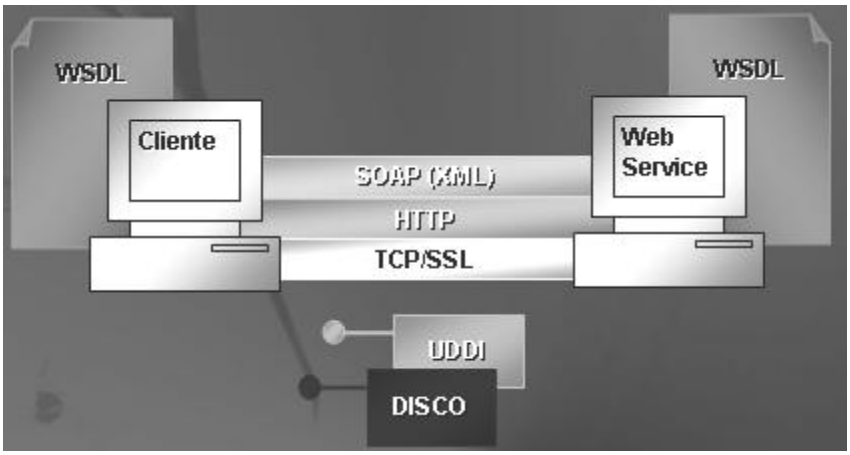


Fig. 2.4: (Estándares del Web Service)

La implementación de un Web Service genérico se comunica a través del Internet y a través de firewalls. La parte que inicializa el dialogo con un Web Service es llamada el Subscriptor;

mientras que la parte que acepta los requerimientos es llamada el Publicador. Visto desde afuera ambos, tanto el subscriptor como el publicador son implementados como "caja negra".

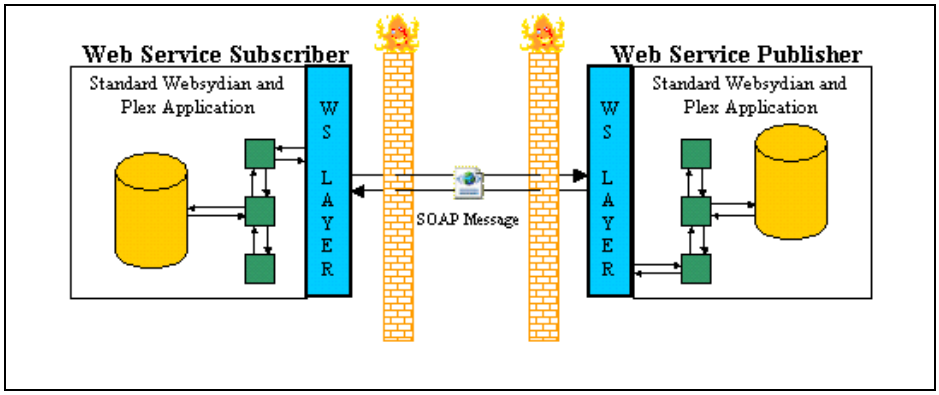


Fig. 2.5: (Comunicación del Web Service)

2.2.3.1 SOAP

SOAP (Simple Object Access Protocol) es un protocolo ligero para el intercambio de información en un entorno descentralizado y distribuido. SOAP utiliza tecnología XML para definir un sistema de mensajes extensible, que permite la construcción de mensajes que pueden ser intercambiados sobre otros protocolos (TCP/IP, HTTP, SMTP, etc). SOAP ha sido diseñado para que su funcionalidad sea independiente de cualquier modelo de programación e implementación específica.

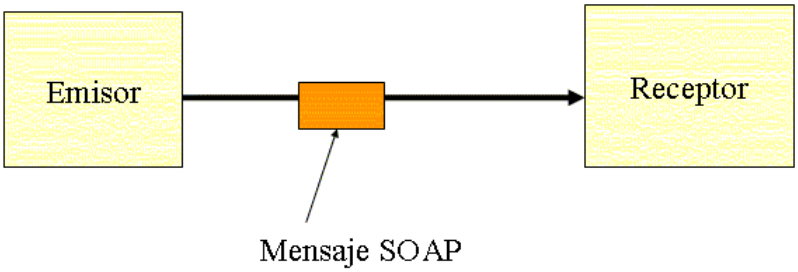


Fig. 2.6: (Mensaje SOAP)

No es lo mismo SOAP que RPCs + XML; más bien la programación es mas sencilla. SOAP es igual a paso de mensajes en XML ya que es una base de mensajería sobre la que se pueden crear modelos como petición/respuesta

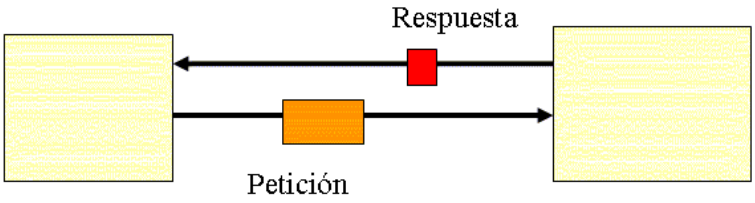


Fig. 2.7: (SOAP: Petición / Respuesta)

2.2.3.1.1 Mensaje SOAP

Un mensaje SOAP es fundamentalmente una transmisión en un solo sentido entre nodos SOAP, de un remitente SOAP a un destinatario SOAP, pero se espera que los mensajes SOAP sean combinados por las aplicaciones para implementar patrones de interacción más complejos desde la petición / respuesta a múltiples intercambios "conversacionales" de ida y vuelta.

Estos fundamentos comienzan con la exposición de la estructura de un mensaje SOAP y su intercambio en algunos escenarios de uso sencillos por ejemplo una aplicación de reservas de viajes. Se utilizarán varios aspectos del escenario de esta aplicación para explicar algunos conceptos de SOAP. En este escenario, la aplicación de reservas de viajes negocia la solicitud de reserva de un viaje planeado para un empleado de una empresa, con un servicio de reservas. La información intercambiada entre la aplicación de reservas de viajes y la aplicación del servicio de viajes se realiza en forma de mensajes SOAP.

El destinatario último de un mensaje SOAP enviado desde la aplicación de reservas de viajes es la aplicación del servicio de reservas, pero es posible que el mensaje SOAP pueda ser "encaminado" a través de uno ó más intermediarios SOAP que actúen de algún modo sobre el mensaje. Algunos ejemplos sencillos de tales intermediarios SOAP podrían ser aquellos que apuntan, auditan, o posiblemente enmiendan cada petición de viaje.

Para mayor entendimiento explicaremos un ejemplo de mensaje SOAP para una reserva de viajes, con bloques de encabezado y cuerpo.

Cuadro 2.2: (Ejemplo de Mensaje SOAP)

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reserva xmlns:m="http://empresaviajes.example.org/reserva"
        env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
   <m:referencia>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:referencia>
   <m:fechaYHora>2001-11-29T13:20:00.000-05:00</m:fechaYHora>
  </m:reserva>
  <n:pasajero xmlns:n="http://miempresa.example.com/empleados"
        env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
   <n:nombre>Åke Jógvan Øyvind</n:nombre>
  </n:pasajero>
 </env:Header>
 <env:Body>
  <p:itinerario
    xmlns:p="http://empresaviajes.example.org/reserva/viaje">
   <p:ida>
     <p:salida>Nueva York</p:salida>
     <p:llegada>Los Angeles</p:llegada>
     <p:fechaSalida>2001-12-14</p:fechaLlegada>
     <p:horaSalida>última hora de la tarde</p:horaSalida>
     <p:preferenciaAsiento>pasillo</p:preferenciaAsiento>
   </p:ida>
```

```
   <p:vuelta>
     <p:salida>Los Angeles</p:salida>
     <p:llegada>Nueva York</p:llegada>
     <p:fechaSalida>2001-12-20</p:fechaSalida>
     <p:horaSalida>media-mañana</p:horaSalida>
     <p:preferenciaAsiento/>
   </p:vuelta>
  </p:itinerario>
  <q:alojamiento
   xmlns:q="http://empresaviajes.example.org/reserva/hoteles">
   <q:preferencia>ninguna</q:preferencia>
  </q:alojamiento>
 </env:Body>
</env:Envelope>
```

El mensaje SOAP del Ejemplo contiene dos subelementos específicos de SOAP dentro del elemento global env:Envelope [env:Sobre], denominados env:Header [env:Encabezado] y env:Body [env:Cuerpo]. Los contenidos de estos elementos son definidos por la aplicación y no son parte de las especificaciones SOAP, aunque el último de ellos tiene algo que decir acerca de cómo deben ser utilizados tales elementos.

El elemento SOAP header [encabezado SOAP] es opcional, pero ha sido incluido en el ejemplo para explicar ciertas características de SOAP. Un encabezado SOAP es un mecanismo de extensión que proporciona un modo de pasar información en mensajes SOAP que no es parte de la generada por la aplicación. Tal información de "control" incluye, por ejemplo, el paso de directivas o de información contextual relativa al proceso del mensaje. Esto permite extender el mensaje SOAP de una manera específica a cada aplicación. Los elementos hijos inmediatos del elemento env:Header son denominados bloques de encabezado, y representan un agrupamiento lógico de datos que, como se verá posteriormente,

puede ser dirigido individualmente a nodos SOAP que podrían encontrarse en el camino de un mensaje desde un remitente hasta un destinatario final.

Los encabezados SOAP han sido diseñados en anticipación a varios usos de SOAP, muchos de los cuales implicarán la participación de otros nodos de proceso SOAP - llamados intermediarios SOAP - a lo largo del camino del mensaje desde el remitente SOAP inicial hasta el destinatario SOAP final. Esto permite a los intermediarios SOAP proporcionar servicios de valor añadido. Los encabezados, pueden ser inspeccionados, insertados, eliminados, o redirigidos por nodos SOAP encontrados a lo largo del camino de un mensaje SOAP. (Debe tenerse en cuenta, de todas formas, que las especificaciones SOAP no tratan con lo que son los contenidos de los elementos del encabezado en sí, o con la forma en la que los mensajes SOAP son enrutados entre los nodos, o con el modo en el que se determina la ruta y así sucesivamente. Éstos son parte de la aplicación global, y podrían ser objeto de otras especificaciones.)

El SOAP body [cuerpo SOAP] es el elemento obligatorio dentro del elemento SOAP env:Envelope, lo que implica que es aquí donde debe alojarse la información convenida en SOAP que se transporta de extremo a extremo.

Una representación gráfica del mensaje SOAP del ejemplo mencionado es como sigue.

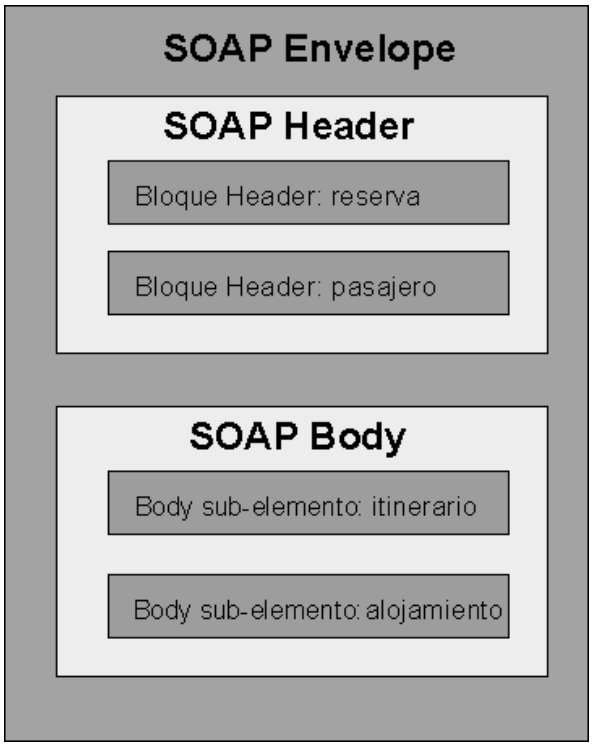


Fig. 2.8: (Estructura del Mensaje SOAP)

En el ejemplo, el encabezado contiene dos bloques de encabezado, cada uno de ellos se define en su propio espacio de nombres XML y representa algún aspecto perteneciente al proceso global del cuerpo del mensaje SOAP. Para esta aplicación de reservas, esa "meta" información perteneciente a la petición global son los bloques de encabezado reserva, que proporciona una referencia y una marca de tiempo para esta instancia de una reserva, y la identidad del viajero en el bloque pasajero.

Los bloques de encabezado reserva y pasajero deben ser procesados por el siguiente intermediario SOAP que se encuentre en el camino del mensaje o, si no existe tal intermediario, por el destinatario último del mensaje. El hecho de que es dirigido al siguiente nodo SOAP que se encuentre *en ruta* es indicado por la presencia del atributo env:role

[env:papel] con el valor "http://www.w3.org/2003/05/soap-envelope/role/next" (de ahora en adelante simplemente "next" ["siguiente"]), que es un role que todos los nodos SOAP deben estar dispuestos a jugar. La presencia de un atributo env:mustUnderstand [env:debeEntender] con el valor "true" ["cierto"] indica que el/los nodo(s) que procesan el encabezado deben procesar absolutamente esos bloques de encabezado de una forma consistente con sus especificaciones, o si no, no procesar el mensaje de ningún modo y lanzar un error. Nótese que cuando un bloque de encabezado es procesado es porque, o bien ha sido marcado con env:mustUnderstand="true" o por otra razón, el bloque debe ser procesado de acuerdo con las especificaciones para ese bloque. Tales especificaciones de los bloques de encabezado son definidas por la aplicación y no forman parte de SOAP.

La elección de los datos que serán empleados en un bloque de encabezado y los que van en el cuerpo SOAP son decisiones que se toman en el momento del diseño de la aplicación. El punto principal que hay que tener en cuenta es que los bloques de encabezado pueden ser dirigidos a varios nodos que podrían encontrarse a lo largo del camino de un mensaje que vaya desde un remitente a un destinatario final. Tales nodos intermediarios SOAP pueden proporcionar servicios de valor añadido basados en los datos de dichos encabezados. En el ejemplo, los datos del pasajero se ubican en un bloque de encabezado para ilustrar el uso de estos datos en un intermediario SOAP que realice procesamiento adicional.

El elemento env:Body y sus elementos hijos asociados, itinerario y alojamiento, están destinados al intercambio de información entre el remitente SOAP inicial y el nodo SOAP que asume el papel del destinatario SOAP final en el camino del mensaje, que es la aplicación del servicio de viajes. Por tanto, el env:Body y sus contenidos son dirigidos implícitamente y

esperan ser entendidos por el destinatario último. Los medios por los que un nodo SOAP asume tal papel no están definidos por la especificación SOAP, y es determinado como parte de la semántica y el flujo de mensajes de la aplicación global.

Observe que un intermediario SOAP puede decidir realizar jugar el papel del destinatario SOAP final para una determinada transferencia de mensaje, y entonces procesar el env:Body. Sin embargo, aunque este tipo de comportamiento no puede prevenirse, no es algo que debiera hacerse a la ligera ya que puede distorsionar las intenciones del remitente del mensaje, y tener efectos laterales no deseados (como no procesar los bloques de encabezado que podrían estar dirigidos a intermediarios más allá del camino del mensaje).

Un mensaje SOAP como el del ejemplo puede ser transferido por diferentes protocolos y utilizado en una variedad de patrones de intercambio de mensajes. Por ejemplo, para un acceso a través de la Web a la aplicación del servicio de viajes, podría ser ubicado en el cuerpo de una petición HTTP POST.

2.2.3.2 WSDL

El lenguaje de descripción de Web Services WSDL (Web Services Description Language) es un lenguaje XML que contiene información acerca de la interfaz, semántica, y administración de una llamada a un Web Service.

Una vez que se ha desarrollado un Web Service, se publica su descripción y se construye un link o apuntador en un depósito UDDI (Universal Description, Discovery and Integration) para que los usuarios potenciales lo puedan utilizar. Cuando alguien piensa en utilizar este

Web Service, solicitan el archivo WSDL para conocer la ubicación del servicio, llamado de funciones, y cómo acceder al Web Service. Luego utilizan la información en el archivo WSDL para construir una petición SOAP (Simple Object Access Protocol) y enviarla hacia el proveedor de servicio.

Una de las ideas centrales detrás de los Web Services es que las aplicaciones futuras estarán conformadas de una colección de servicios habilitados en la red. Mientras haya dos servicios equivalentes que se publiquen en la red de una forma estándar y neutra, en teoría una aplicación podría seleccionar uno de ellos en base a criterios establecidos de antemano como precio o rendimiento. Además, algunos servicios podrían permitir que fueran copiados entre máquinas, permitiendo así que una aplicación que corra en una máquina (o cluster de máquinas) mejore en rendimiento al copiar servicios útiles a unidades de disco locales.

Se podría hacer una analogía de esta situación con la del mercado de trabajo. Las compañías de contrataciones proveen un servicio de aparejamiento entre trabajadores y empleadores, utilizando los resúmenes personales o currículum vitae y las descripciones de los trabajos ofertados para facilitar el proceso de búsqueda. Si se encuentra una buena opción, las partes interesadas intentan negociar condiciones aceptables para ambas. Si se logra un acuerdo, el trabajador se traslada a la nueva empresa o usa las ventajas del Internet para trabajar a distancia.

2.2.3.2.1 Anatomía de un documento WSDL

Si se examina cada parte de un documento WSDL se encontrará:

<definitions>. El elemento <definitions> contiene la definición de uno o más servicios. En la mayoría de los casos, un archivo WSDL define un servicio únicamente. Seguido de la etiqueta de definición se encontrarán declaraciones de algunos atributos.

Dentro de la etiqueta <definitions> se encuentran tres secciones conceptuales:

- **<message>** y **<portType>**, describe *qué* operaciones provee el servicio.
- **<binding>**, describe cómo se invocan las operaciones.
- **<service>**, describe dónde se ubica el servicio.
- **<documentation>**, cualquier elemento WSDL puede contener información del servicio para el usuario.

2.2.3.3 UDDI

La especificación UDDI (Universal Description, Discovery, and Integration) ha sido desarrollada por IBM, Microsoft y Ariba y está soportada por más de 300 empresas, incluyendo Oracle, Sun Microsystems y Nortel Networks. La última versión, v2, extiende la funcionalidad UDDI para mejorar el soporte con objeto de desarrollar registros públicos y privados que gestionen los Web Services internos empleando las especificaciones UDDI.

La especificación UDDI, junto con Extensible Markup Language (XML), Simple Object Access Protocol (SOAP) y Web Services Description Language (WSDL), están ganando un amplio soporte en el marco de trabajo de los Web Services. De hecho, cerca de 7.000 empresas se han registrado en el directorio UDDI.

UDDI es una especificación para un registro distribuido de información sobre servicios Web, define una forma de publicar y descubrir información acerca de Web Services. Una nube de registros UDDI, o registro de negocios (registro único y publicación generalizada), proporciona información de cómo acceder los Web Services.

El modelo UDDI/SOAP, no es el único modelo para descubrimiento y mensajes en los Web Services. El modelo web XML ha sido también desarrollado para hacer lo mismo, así como también brindar una interfaz de software para negocios, funciones de seguridad robusta y otras, que permiten transacciones reales de e-comerce. El XML y UDDI/SOAP son tecnologías complementarias y al final se pueden utilizar ambos métodos a la vez.

Cómo su nombre lo indica, el estándar UDDI provee un mecanismo para que los negocios se "describan" a si mismos y los tipos de servicios que proporcionan y luego se pueden registrar y publicarse en un Registro UDDI. Tales negocios publicados pueden ser buscados, consultados o "descubiertos" por otros negocios utilizando mensajes con SOAP. Una vez descubiertos los negocios con quien se pueden asociar, los negocios pueden utilizar este mecanismo para "integrar" sus servicios en conjunción con sus socios y proveer los servicios a sus clientes.

2.2.3.3.1 Secciones Blanca, Amarilla y Verde.

Conceptualmente, la información proporcionada de un negocio en un registro UDDI consta de tres componentes:

- La **Sección Blanca** es muy similar a la información que aparece en el directorio telefónico, que incluye nombre, teléfono, y dirección.

- La **Sección Amarilla** es muy similar a su equivalente telefónico, e incluyen categorías de catalogación industrial tradicionales, ubicación geográfica, etc. Mediante el uso de códigos y claves predeterminadas, los negocios se pueden registrar y así facilitar a otros servicios la búsqueda usando estos índices de clasificación.
- La **Sección Verde** contiene la información técnica acerca de los servicios ofrecidos por los negocios. Se incluyen referencias de especificaciones de Web Services así como también información complementaria para los mecanismos diversos de búsqueda basados en URL.

2.2.3.3.2 Estructura Central del UDDI

La información que compone un registro UDDI consiste por el momento de cuatro tipos de estructuras de datos (cinco en la versión UDDI 2.0)

- **businessEntity:** Esta estructura captura la información sobre un negocio o entidad y que es utilizada por el negocio para publicar información descriptiva sobre si misma y los servicios que ofrece.
- **businessService:** Esta estructura representa los servicios o procesos de negocios que provee la estructura **businessEntity**.
- **bindingTemplate:** Esta estructura representa los datos importantes que describen las características técnicas de la implementación del servicio ofrecido.
- **tModel:** El papel principal de esta estructura es la de representar una especificación técnica.

2.2.3.3.3 Características del UDDI

El UDDI provee dos categorías de API:

El API de publicación y el API de consulta. El API de publicación, provee el mecanismo para que los proveedores de servicios se registren ellos mismos y sus servicios en el Registro UDDI.

El API de consulta permite a los subscriptores de servicios buscar los servicios disponibles.
El API de consulta provee dos tipos de llamados, un mecanismo de búsqueda y un mecanismo de obtención, cuando ya se tiene disponible toda la información referente a la disponibilidad de un servicio.

Al registrar en UDDI a su empresa, los tModels personalizados y los servicios que ofrece, facilita a los desarrolladores la búsqueda de los Web Services. El registro UDDI permite exponer algo más que extremos de Web Services e información sobre empresas. Los usuarios de UDDI pueden utilizar la interfaz para buscar documentación y ejemplos del Web Services.

2.3 Advantage Plex

Advantage Plex es un ambiente de desarrollo de trabajo en grupo basado en modelos que utiliza tecnología de patrones para diseñar y construir aplicaciones críticas de negocios, a partir de modelos diseñados por desarrolladores, Advantage Plex utiliza generadores para entregar eficiente código nativo para el AS/400, Windows NT, Java.

Advantage Plex se preocupa de crear procesos distribuidos, diseño y creación de base de datos-DB2/400, SQL Server, Oracle 8. Advantage Plex integra sus existentes aplicaciones y permite liberarlas para Internet.

Es una herramienta de desarrollo altamente innovadora que permite acelerar el proceso de diseño, desarrollo e implementación de las aplicaciones multiplataforma para ambientes:

- Windows NT (WinC)
- Java
- AS/400 iSeries

Es así como hace posible adoptar nueva tecnología con un riesgo mínimo y un costo reducido.

2.3.1 Soluciones Advantage

Advantage proporciona un entorno sólido de desarrollo de aplicaciones, una plataforma de integración eBusiness de extremo a extremo y un completo conjunto de soluciones de bases de datos, que ofrecen una gestión de la información eBusiness. La familia de productos Advantage, ofrece soluciones que permiten a las empresas desarrollar y distribuir rápidamente

aplicaciones de nivel superior compatibles con las iniciativas eBusiness. Advantage proporciona tecnología de potencial industrial y todos los servicios necesarios para la integración rápida de aplicaciones, bases de datos y sistemas de socios de negocios. Las empresas pueden crear y depurar rápidamente información para los mercados eBusiness y almacenar datos, sin comprometer la facilidad de uso, las transformaciones complejas y la gestión potente de los metadatos. Desde la automatización del diseño de un almacén de datos óptimo hasta la oferta de fácil acceso a una amplia gama de recursos de información.

Advantage Plex para los sistemas distribuidos es una herramienta de desarrollo altamente productiva que utiliza patrones, modelos y generadores del software para permitir a organizaciones diseñar y entregar las aplicaciones de negocio sofisticados para ambientes Windows, J2EE y los iSeries 400. Los patrones del software permiten que los desarrolladores empleen tecnologías del Internet que emergen usando un conjunto de habilidades, técnicas y un ambiente.

2.3.2 Características Generales

- **El Negocio debe Adaptarse a las Nuevas Tecnologías**

  El Internet redefine tendencias del mercado en el desarrollo del uso de nuevas tecnologías basadas en Web, requieren que las organizaciones definan una estrategia de negocio clara que abarque varias áreas, ayudando a asegurar la prosperidad del negocio. Estas tecnologías conducen a un servicio mejorado para el cliente, un costo de distribución más bajo, un control de inventario más apretado y productividad creciente.

- **Permite Rápida Adaptación e Integración**

  Plex para los sistemas distribuidos de Computer Associates International, Inc. (CA) tiene como ventaja ser una herramienta de desarrollo basada en modelos que permite la adopción de la nueva tecnología con riesgo mínimo y reduce significativamente los costos.

  Los patrones de software permiten que las aplicaciones distribuidas sean fácilmente extendidas hacia tecnologías dinámicas de la Web tales como aplicaciones inalámbricas o aplicaciones transaccionales usando XML.

- **Productividad Creciente**

  Los patrones predefinidos usados o modificados para requisitos particulares que satisfacen necesidades exactas del negocio, proporcionan una construcción de aplicaciones basadas en modelos específicos que pueden ser implementados y reutilizados a través de aplicaciones acelerando el diseño y la exactitud de todo el sistema.

- **Comunicación Realzada**

  Permite el desarrollo de aplicaciones que se centran en requisitos del negocio más que especificaciones técnicas. La comunicación entre los responsables del negocio y los departamentos técnicos se realza a través de las TI.

- **Agilidad Creciente a las Demandas del Mercado**

  Permite que las organizaciones integren fácilmente sus datos y aplicaciones existentes de negocio con tecnologías del Internet que emergen para satisfacer objetivos de negocio actuales y futuros.

2.3.3 Características Distintivas

- **Basado en Modelos**

  Modelos que almacenan toda la información de diseño de la aplicación en un punto central. Los Modelos garantizan exactitud, consistencia, agilidad de desarrollo y mantenimiento.

- **Patrones**

  Advantage Plex permite utilizar y hacer Patrones para acelerar el diseño y construcción de aplicaciones. Con Advantage Plex vienen patrones y estos pueden ser personalizados o adquiridos desde grupo de Empresas desarrolladoras.

- **Repositorio de Grupos de Trabajo**

  La estructura del Repositorio de Grupo de trabajo mantiene a los desarrolladores trabajando sincronizados. Esto permite a grupos de desarrolladores trabajar eficientemente. Maneja Seguridad, Versiones, Estándares de Diseños y resolución de conflictos.

- **Diagramadores, Editores**

  Fácil de usar, ambiente estilo Visual Basic con facilidades para manejar todas las fases del ciclo de vida de una aplicación.

- **Configuración Incorporada**

  Cada objeto en un Modelo puede ser configurado por Versión, Nivel, Variante e Idioma. Permitiendo configurar:

  - Objetos diferentes entre versiones
  - Información de Plataforma
  - Idiomas

- **Generación en Código Nativo**

  Los generadores transforman las especificaciones del Modelo en Código Nativo. Los generadores separan el desarrollo de cambios tecnológicos y permite para múltiple plataformas de desarrollo desde un mismo diseño.

- **Generadores**

  Advantage Plex Crea código nativo, Multi-Capa para las siguientes plataformas:

  - **Java** servers con Win32 C++ o Java clients, JDBC access
  - **AS/400** Servers con Win32 C++ or Java clients, AS/400 5250, con SQL, JDBC y DDS access para DB2/400
  - **Windows NT** completo BackOffice-compliant Windows NT servers con Win32 C++ clients. Soporta para Oracle 8 y SQL Server
  - **HTML, Domino Patterns** para Lotus Domino y/o HTML Browser.

- **Soporta Múltiples Lenguajes**

  Plex soporta 25 lenguajes diferentes, incluyendo Japonés, Francés, Alemán, Italiano y Español.

2.3.4 Ventajas

- Incrementa la productividad del equipo de desarrollo.
- Herencia de patrones predefinidos reutilizables.
- Reduce el tiempo de aprendizaje de los desarrolladores.
- Aplicaciones multiplataforma permite que el desarrollo se oriente a las necesidades del negocio y no las especificaciones técnicas.
- Advantage Plex, permite desarrollar desde diversos backgrounds orígenes para trabajar en un ambiente común y entregar aplicaciones usando una variedad de tecnología.
- Advantage Plex genera completa conformidad con aplicaciones Microsoft Backoffice.

2.3.5 Desarrollo Basado en Modelos

Las aplicaciones construidas con Advantage Plex se diseñan a nivel del negocio para asegurar exactitud y consistencia. Desde un solo diseño, las aplicaciones pueden ser explotadas en una variedad de plataformas y de ambientes.

El mantenimiento de la aplicación se realiza en el nivel del negocio, no en el código. Esto se asegura de que la TI y la administración del negocio estén sincronizados, y de que los cambios a las necesidades del negocio estén reflejados en las aplicaciones más rápidamente.

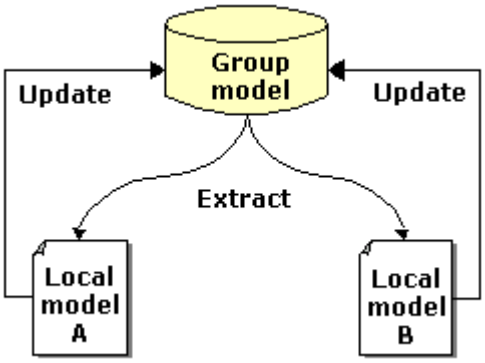


Fig. 2.9: (Desarrollo de modelos)

2.3.6 Patrones

Los patrones son los "bloques construidos reutilizables" cuyas características se pueden heredar inmediatamente dentro de aplicaciones ya sea para interfaz de usuario como para un sistema de reglas de negocio fundamentales. Los estudios independientes han demostrado que el uso de patrones es hasta 53 veces más productivos que otros medios de desarrollo.

La ventaja de Plex es que viene con centenares de bibliotecas patrón. Los patrones pueden ser propios de Plex, o pueden ser comprados a terceros (Websydian).

2.3.6.1 Tipos De Patrones

- **Patrones técnicos:** Suministra interfaces a otras tecnologías como: Windows, API, Packaged software
- **Primitivos y bases:** Los Patrones son pequeños, medianos y grandes bloques comunes para todas las aplicaciones como: UI Basics, One-to-many, Referenced entity, Attribute extensions, Tab dialog.

- **Business fundamentals:** Son Patrones a nivel de aplicaciones que pueden ser aplicados sobre la mayoría de aplicaciones de negocios
- **Cross-Industry**: Son Patrones que pueden que cubren múltiples industrias o representa aplicaciones centrales como: Inventarios, Productos, A/P, Customer service.
- **Industry:** Nivel de Patrones son usados para crear verticales soluciones de aplicaciones como: Financieros, Bancos, salud etc.

2.3.7 Ambiente De Trabajo

Advantage Plex puede ser configurado de varias maneras para ser utilizado en una red de área local como se ilustra en la siguiente figura:

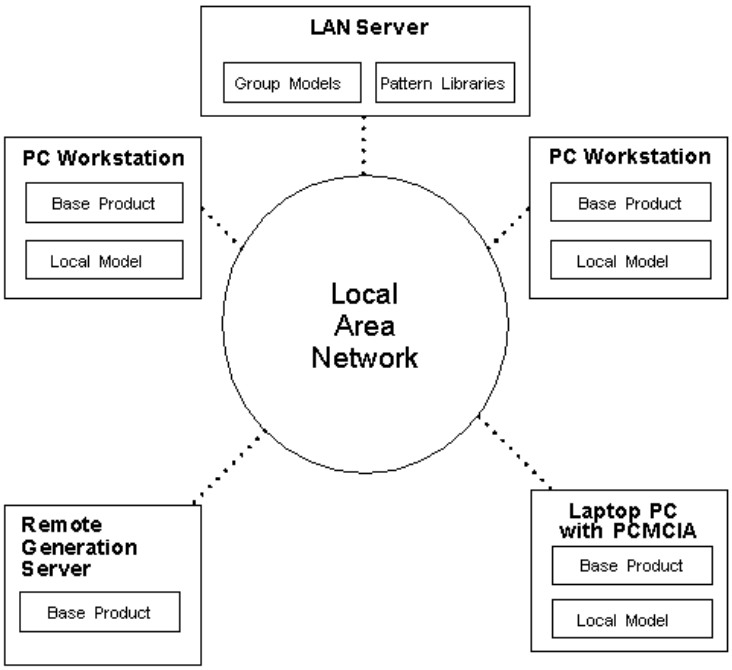


Fig. 2.10: (Ambiente de trabajo del Plex)

Sus modelos de grupo y bibliotecas del modelo residen en el servidor LAN. Los modelos locales residen en las estaciones de trabajo de cada desarrollador. Ellos también pueden residir en una computadora portátil con una conexión a la red. Cuando se esté trabajando en un modelo de grupo, solo necesita estar conectado a la red.

2.4 WEBSYDIAN

El objetivo básico de la arquitectura de Websydian es facilitar la comunicación entre aplicaciones de Advantage Plex y los Navegadores Web en un método directo.

El Software de los Servidores Web (Microsoft Internet Information Server o Apache Web Server) sirve como gateway entre el navegador Web y las aplicaciones Web. Éste se comunica con el navegador usando HTTP (HyperText Transfer Protocol), y las aplicaciones Web usan una interfase estándar, como por ejemplo CGI (Common Gateway Interface) o ASP (Active Server Pages).

Los patrones de Websydian son básicamente dos cosas:

- Este implementa la interfase entre el software del Servidor Web y la aplicación Web.
- Proporciona las abstracciones requeridas para permitir que los diseñadores de Advantage Plex lleven a cabo las Aplicaciones Web básicamente de la misma manera que ellos llevan a cabo las aplicaciones no Web.

A diferencia del cliente de Advantage Plex, Websydian genera un sistema run-time que une el Servidor Web a las funciones del cliente. Las funciones heredadas del Websydian nos ayudan a generar Páginas Web de las que se envían al navegador en respuesta a eventos recibidos.

Hay algunos puntos adicionales que vale la pena mencionar en relación a esta herramienta:

- El cliente en la arquitectura de Websydian es un cliente como cualquier otro de la aplicación Advantage de Plex. La única diferencia es que el cliente de Websydian no tiene un GUI atado a él, pero consigue su entrada del usuario como requisito HTTP del Web o Navegador Inalámbrico.

- Websydian tiene el cuidado de todo relacionado a recuperar e interpretar la entrada del Navegador de Web y transferirlos a los campos de variables regulares de Advantage Plex.

- Websydian pone cuidado en la codificación en la contestación de la aplicación como una Página Web (como entienden los Navegadores Web).

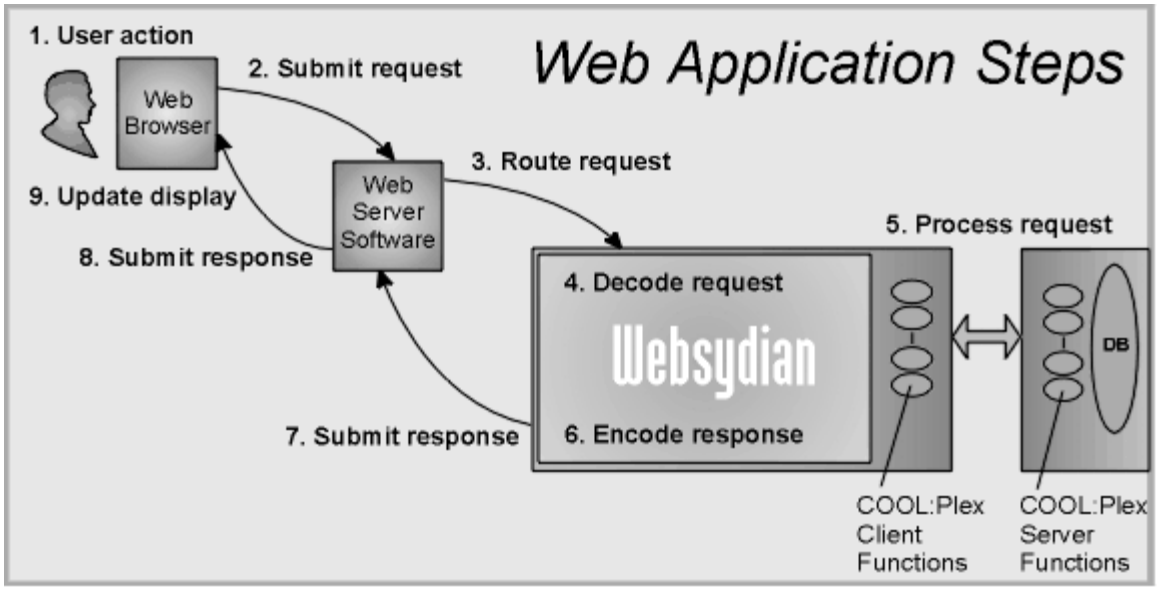


Fig. 2.11: (Pasos de la Aplicación Web sobre Websydian)

Las Aplicaciones Web, como las aplicaciones GUI, son eventos manejados por el usuario. Cuando un evento (es decir un requisito de un Navegador Web) se activa, la aplicación

simplemente realiza un adecuado proceso para proporcionar una contestación, y en ese momento se detiene.

2.4.1 Círculo del Websydian

En la figura siguiente se observa el proceso del Websydian desde la caja marcada como "Web Browser", que representa al usuario del explorador web. Todas las otras cajas de la figura representan las funciones derivadas desde las funciones en las librerías del Websydian. Se puede identificar los controles de flujo entre cada caja de una aplicación en proceso.

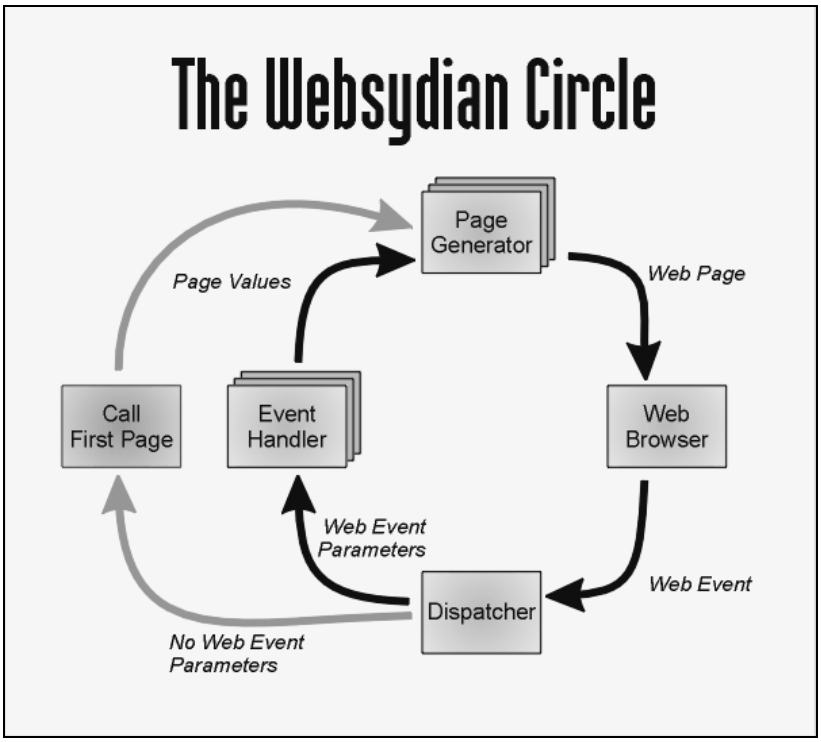


Fig. 2.12: (Círculo del Websydian)

2.4.1.1 El Navegador Web (Web Browser)

El navegador web es donde el usuario se localiza. Cuando el usuario sigue un link o pulsa un botón en la página web, un requisito web se envía a la aplicación. La contestación a una petición web siempre es otra página web, la cual proporciona una respuesta apropiada a cada demanda recibida.

Una aplicación web puede tener muchos usuarios al mismo tiempo, y las demandas de varios usuarios llegan en ningún orden en particular. Cuando requiriera, las aplicaciones web deben guardar la huella de sesiones del usuario explícitamente agregando la lógica de sesión de manejo.

2.4.1.2 Funciones Principales del Websydian

Websydian se ocupa de la tarea de procesar las demandas y controlar el flujo de eventos con un juego de funciones que forman un círculo unido como se ilustra en la figura anterior.

- **Función Distribuidor**

  Los trabajos de función del Distribuidor como el Single-Point-Of-Entry de la Aplicación de Web, recupera toda la entrada recibida del navegador web. Basado en esa entrada, determina si hay que transferir el control a una función del Evento Handler o a la Llamada de la función First Page (cuando un usuario está pidiendo la primera página web). Además, la función del Distribuidor es responsable para realizar los chequeos de seguridad.

Hay siempre sólo una función del Distribuidor en una aplicación de Websydian.

- **Función de la Primera Llamada a la Página**

  La función Call First Page realiza dos tareas importantes, contiene el proceso requerido para crear una sesión del usuario, y llamar a la función de Generador de Página de entrada a la aplicación de Websydian. La función se llama automáticamente por la función del Distribuidor cuando éste detecta que un nuevo usuario web está autentificado en la aplicación de Websydian.

- **Función Manejador de Eventos (Handler)**

  Hay una función Handler para cada evento web que puede ocurrir en la aplicación. El papel de este evento Handler es ejecutar el proceso asociado con el evento correspondiente (por ejemplo poniendo al día la base de datos) y decidir qué página web debe ser la contestación al evento llamando por la Página Generadora que produce esa página. En el curso de su proceso, las funciones de Manejadores de Eventos necesitan a menudo acceder a la base de datos.

  Normalmente, existen muchos Manejadores de Eventos en una aplicación de Websydian.

- **Función Generador de Página**

  Hay una función de Generador de Página para cada página que la aplicación de Websydian puede producir en la respuesta a un evento web. La página web producida

a una demanda específica se crea insertando valores en una Plantilla del Documento. En el curso de su proceso, las funciones de Generador de Página necesitan a menudo acceder a la base de datos.

Normalmente, existen muchas funciones de páginas generadoras en una aplicación de Websydian, uno por cada página HTML en la aplicación.

2.4.2 Websydian y la Arquitectura CWA

CWA (Classic Websydian Architecture), es la clásica arquitectura que ha utilizado el Websydian a comparación del DWA (Distribuido). La primera interfase soportada como servidor web en Websydian fueron los CGI sobre la plataforma Windows y las plataformas iSeries.

Tabla 2.2: (Pros y Contras de la Arquitectura CWA de Websydian)

Pros	Contras
Simple estructura y desarrollo	Ya que cada requisito es manejado por un proceso CGI separado, éste pone la carga pesada al servidor web.
El interfase CGI es soportado en forma nativa por todos los servidores web.	No son muy escalables
Tecnología verificada	Relativamente responde a tiempos largos
Falla cuando manipula un requisito que no es influencia de otro requisito tal como respuesta en un proceso separado.	El balanceo de carga no es posible en Websydian

Los Web Services trabajan nativamente sobre una arquitectura distribuida por su forma estructurada, por lo que la arquitectura CWA sobre WinC-RPG es el complemento ideal para el desarrollo del mismo.

2.5 Metodología y Modelamiento UML

En base a los modelos que se trabaja en el Websydian y Plex, este sistema se desarrollará con la metodología Orientado a Objetos, basado en el Modelamiento UML.

2.5.1 Definición

UML (Unified Modeling Language) Lenguaje Unificado de Modelamiento

"UML es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir."

UML es también un lenguaje de modelamiento visual que permite una abstracción del sistema y sus componentes.

2.5.2 Objetivos del UML

- Debe ser un lenguaje universal, como cualquier lenguaje de propósito general.
- Imponer un estándar mundial.

2.5.3 Casos de Uso

El diagrama de casos de uso representa la forma en como un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan (operaciones o casos de uso).

Un diagrama de casos de uso consta de los siguientes elementos:

- Actor.
- Casos de Uso.
- Relaciones de Uso, Herencia y Comunicación.

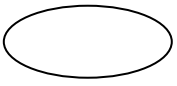
2.5.3.1 Elementos

2.5.3.1.1 Actores

- Principales: personas que usan el sistema.
- Secundarios: personas que mantienen o administran el sistema.
- Material externo: dispositivos materiales imprescindibles que forman parte del ámbito de la aplicación y deben ser utilizados.
- Otros sistemas: sistemas con los que el sistema interactúa.

La misma persona física puede interpretar varios papeles como actores distintos, el nombre del actor describe el papel desempeñado.

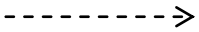
Los Casos de Uso se determinan observando y precisando, actor por actor, las secuencias de interacción, los escenarios, desde el punto de vista del usuario. Los casos de uso intervienen durante todo el ciclo de vida. Siendo que el proceso de desarrollo estará dirigido por los casos de uso.

- Caso de Uso: 

Es una función específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.

- Relaciones:

Es el tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra función (caso de uso). Dicha relación se denota con una flecha simple.

- Dependencia o Instanciación: 

Es una forma muy particular de relación entre clases, en la cual una clase depende de otra, es decir, sé instancia (se crea). Dicha relación se denota con una flecha punteada.

- Generalización

Este tipo de relación es uno de los más utilizados, cumple una doble función dependiendo de su estereotipo, que puede ser de Uso (<<uses>>) o de Herencia (<<extends>>).

Este tipo de relación esta orientado exclusivamente para casos de uso (y no para actores);

extends: se recomienda utilizar cuando un caso de uso es similar a otro (características).

uses: se recomienda utilizar cuando se tiene un conjunto de características que son similares en más de un caso de uso y no se desea mantener copiada la descripción de la característica.

2.5.4 Diagramas de Interacción

Existen dos tipos de diagramas de interacción: el Diagrama de Colaboración y el Diagrama de Secuencia.

El Diagrama de Secuencia es más adecuado para observar la perspectiva cronológica de las interacciones, muestra la secuencia explícita de mensajes y son mejores para especificaciones de tiempo real y para escenarios complejos. El Diagrama de Colaboración ofrece una mejor visión espacial mostrando los enlaces de comunicación entre objetos, muestra las relaciones entre objetos y son mejores para comprender todos los efectos que tiene un objeto y para el diseño de procedimientos.

El diagrama de Colaboración puede obtenerse automáticamente a partir del correspondiente diagrama de Secuencia (o viceversa).

2.5.4.1 Diagramas de Secuencia

- Muestra la secuencia de mensajes entre objetos durante un escenario concreto.
- Cada objeto viene dado por una barra vertical.
- El tiempo transcurre de arriba abajo.
- Cuando existe demora entre el envío y la atención se puede indicar usando una línea oblicua.

2.5.4.1.1 Línea de vida de un objeto

Un objeto se representa como una línea vertical punteada con un rectángulo de encabezado. El rectángulo de encabezado contiene el nombre del objeto y el de su clase, en un formato nombreObjeto: nombreClase.

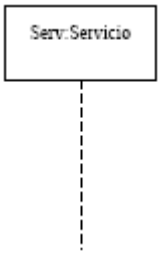


Fig. 2.13: (Símbolo de la clase o entidad)

Con rectángulos a través de la línea principal que denotan la ejecución de métodos (véase activación).

Fig. 2.14: (Símbolo de la línea de ejecución)

2.5.4.1.2 Activación

Muestra el periodo de tiempo en el cual el objeto se encuentra desarrollando alguna operación, bien sea por sí mismo o por medio de delegación a alguno de sus atributos. Se denota como un rectángulo delgado sobre la línea de vida del objeto.

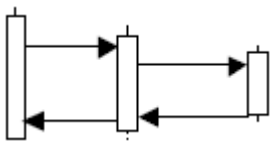


Fig. 2.15: (Símbolo del período del tiempo)

2.5.4.1.3 Mensaje

El envío de mensajes entre objetos se denota mediante una línea sólida dirigida, desde el objeto que emite el mensaje hacia el objeto que lo ejecuta. Terminando este mensaje se da a la destrucción de la operación del objeto denotado por una "X" al finalizar la línea punteada.

Fig. 2.16: (Muestra de un mensaje entre clases)

Ejemplo de una bifurcación de mensajes con validación de datos.

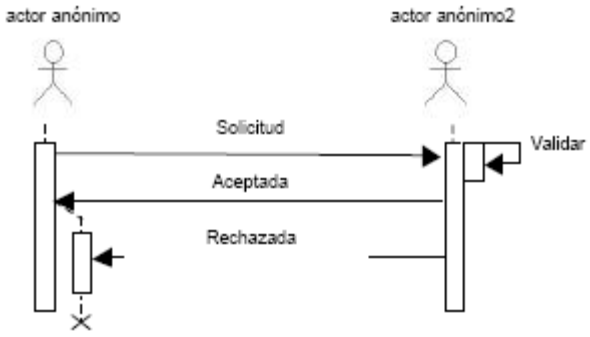


Fig. 2.17: (Ejemplo de un Diagrama de Secuencia)

CAPÍTULO III

ESPECIFICACIÓN DE REQUERIMIENTOS

3.1 Objetivo de la E.R.S

Esta sección tiene como propósito identificar de forma clara las especificaciones necesarias para realizar el Web Services Piloto para Diners Club del Ecuador; de tal forma que cualquier desarrollador del área de tecnología de Diners pueda interpretarlo ya que a pesar de ser un sistema piloto, ésta es una buena base para futuros desarrollos.

3.2 Alcance

Esta aplicación "Web Services Piloto" tiene como función principal interpretar los parámetros de entrada en formato XML vía Web, transformarlos a datos que puedan ser procesados por la base de datos DB2 y devolverlos en formato XML como respuesta a la petición realizada.

Esto permitirá el intercambio de información en línea y en formato XML entre plataformas diferentes; además será el precedente de que se puede realizar cualquier otro desarrollo bajo los beneficios que un Web Services provee.

3.3 Descripción General

3.3.1 Diagrama de contextos

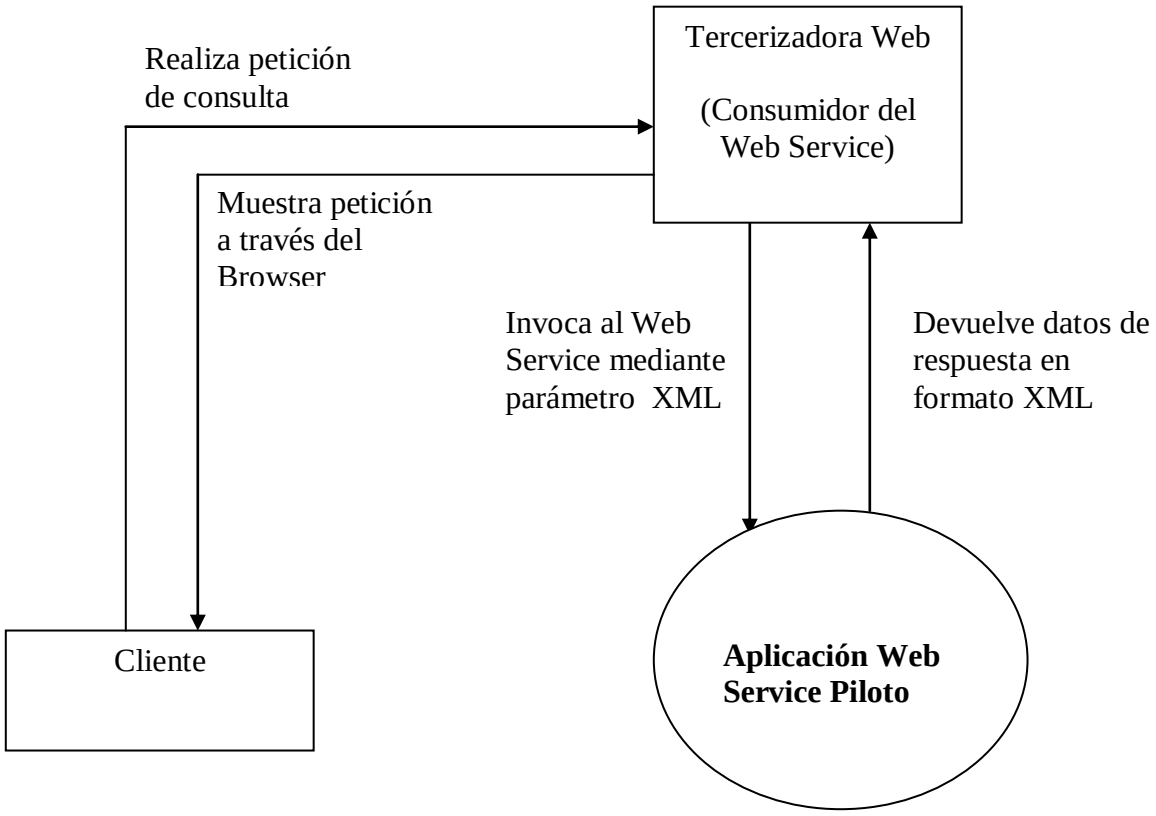


Fig. 3.1: (Diagrama del contextos)

3.3.2 Funciones del Producto

Tabla 3.1: (Funciones del producto)

LISTA DE ACONTECIMIENTOS	FUNCIONES
* El Web Services Piloto recibe de entrada un mensaje XML en el que se detalla el número de tarjeta del cliente	* Recibir mensaje XML de entrada desde el Browser. .
* El sistema Piloto realiza un proceso de transformación de los datos de entrada de formato XML a datos que puedan ser procesados normalmente por cualquier proceso de base de datos.	* Transformar datos de entrada de formato XML a formato que pueda ser interpretado por el proceso de base de datos.
* El Web Services Piloto una vez con los datos transformados recupera la información correspondiente a los movimientos del número de tarjeta indicado.	* Recuperar información de los movimientos realizados por el tarjetahabiente.
* El Web Services Piloto transforma el resultado correspondiente a todos los movimientos realizados por el tarjetahabiente en el mes actual a formato XML	* Transformar los datos obtenidos por el proceso de base de datos a formato XML.
* El Web Services Piloto envía de salida un mensaje XML en el que se detalla cada uno de los movimientos del tarjetahabiente.	* Enviar mensaje XML de salida al Browser

3.3.3 Aspectos de Rendimiento

- El rendimiento está dado por el número de movimientos hechos por el tarjetahabiente.
- Por la rapidez de procesamiento de información en el servidor de base de datos.
- Por la velocidad de transformación de datos de un formato normal a un formato XML o viceversa que lo realiza la máquina cliente donde está ubicado el Web Services (Sistema Piloto).

3.3.4 Restricciones técnicas y de gestión

- Personal no capacitado en el manejo de la nueva herramienta Websydian en Diners.
- Soporte técnico para el desarrollo en Websydian inexistente en SudAmérica.
- Soporte técnico solamente se realiza en el idioma Inglés y con diferencia de 6 horas en el horario; dado que la casa de software productora de Websydian se encuentra en Dinamarca.

3.3.5 Rendimiento

El rendimiento está dado por la velocidad de respuesta a la invocación del Web Services y a la velocidad de procesamiento de recuperación de información en el servidor de base de datos.

3.3.6 Obedencia a los estándares

En esta ocasión por tratarse de un Web Services Piloto y su poca complejidad los esquemas del formato XML se han definido solo por parte del propietario del Web Service.

Pero los esquemas pueden ser definidos y formalizados de acuerdo a las interfaces XML de cualquiera de las partes involucradas ya sea del que publica el Web Services o del que consume el Web Service o mutuo acuerdo.

3.3.7 Limitaciones de Hadware

Si se desea realizar esta aplicación Web Services o cualquier otra aplicación Web Services en plataforma Java es necesaria la adquisición de un servidor Websydian valorado en 24000 dólares norteamericanos.

3.3.8 Interfaces

- **Interfaces de Software**

  No existe interfaz de software para el desarrollo de esta aplicación Web Services Piloto; sin embargo por motivos de demostración del consumo del Web Services realizado se ha creado un cliente Web, tomando como base la interfaz que Diners tiene ya con su tercerizadora.

- **Interfaces de Comunicación**

  La interfaz de comunicación se la realiza a través de documentos XML mediante el protocolo SOAP.

3.3.9 Seguridad

La seguridad será administrada por parte de la empresa tercerizadora mediante el uso de perfiles de usuario con sus respectivas contraseñas.

3.3.10 Recursos

- **Equipos Hadware**
    - Un PC de mínimo 400 Mhz.
    - Servidor de base de datos DB2 que reside en el AS-400.

- **Software**
    - Licencia Advantage Plex 5.1
    - Licencia Websydian 5.5
    - Microsoft Parsing MSXML 4.0
    - Servidor Web Apache en modo consola donde residirá el Web Services.
    - Microsoft Visual Studio 6.0 como compilador C++
    - Consola de acceso al servidor AS-400.

CAPITULO IV

ANÁLISIS Y DISEÑO

4.1 Modelamiento UML

En esta sección describiremos paso a paso el Análisis y Diseño de los principales componentes del sistema del Web Services utilizando la metodología de UML Orientado a Objetos.

UML o lenguaje de Modelamiento Unificado es el conjunto de especificaciones gráficas que sirven para modelar y documentar cada una de las partes que comprende el desarrollo de toda la aplicación.

Las partes utilizadas en este prototipo fueron las de Casos de Uso y Diagramas de Interacción.

Los diagramas de Casos de Uso representan la forma de como cada uno de los "Agentes" humanos o informáticos relacionados con un sistema interaccionan con las funciones de éste. A estos "Agentes" se les denomina Actores.

Los diagramas de Interacciones permiten mostrar gráficamente los mensajes que se intercambian los diferentes actores y el orden en que lo hacen hasta terminar el proceso.

4.1.1 Definición de Actores

La definición del Web Service para la consulta del tarjeta ambiente Diners Club son:

Tarjeta Habiente: Actor principal: Cliente dueño de la tarjeta Diners Club

Rol: Consultar el estado de cuenta Diners

Web Site Diners: Actor secundario: Sitio Web Diners.

Rol: Permite el acceso del cliente a la consulta de su estado de cuenta.

Web Service: Actor secundario: Publicador del servicio web

Rol: Dar la información requerida por el cliente.

4.1.2 Diagrama de Casos de Uso

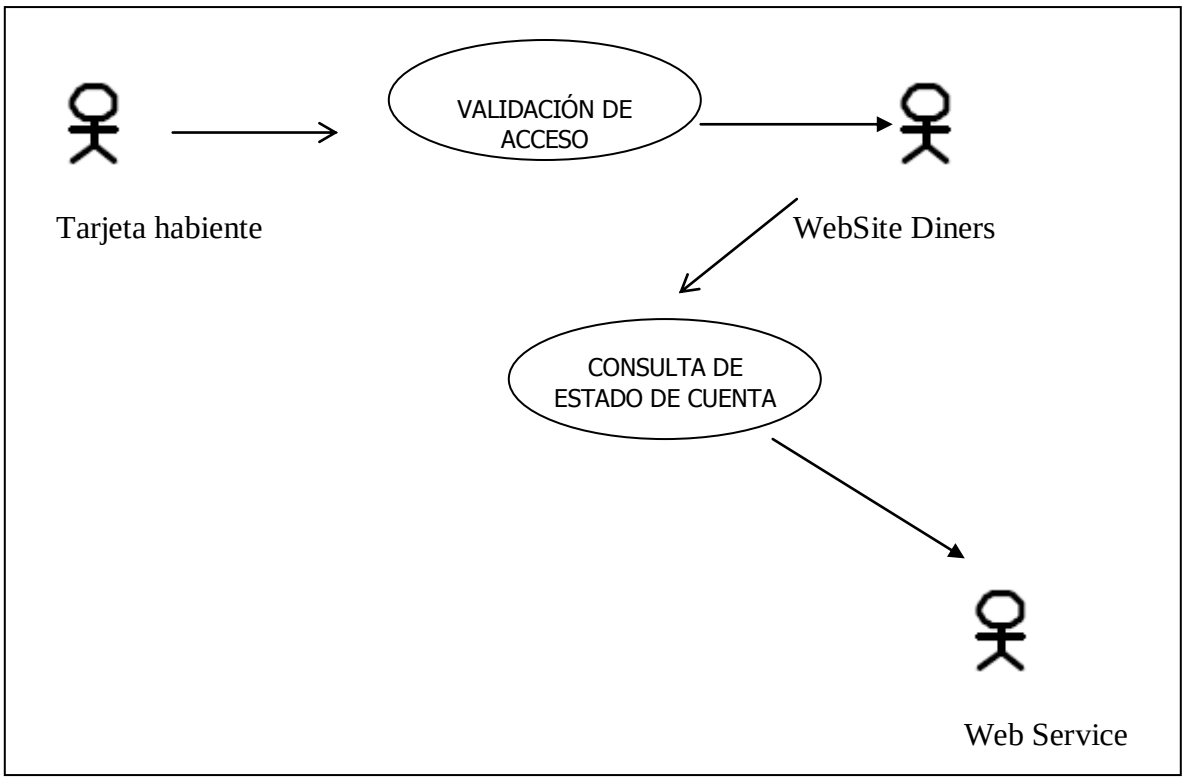


Fig. 4.1: (Diagrama del Caso de Uso de la "Consulta del Estado de Cuenta Diners")

4.1.3 Plantillas de los Casos de Uso

Tabla 4.1: (Descripción del Caso de Uso: Cliente)

Nombre del caso de uso	Validación de Acceso
Actor Principal:	Cliente
Actores involucrados	Cliente: Ingresar Clave para acceder al WebSite Diners. WebSite Diners: Validar acceso de cliente.
Objetivos	Ingresar al WebSite.
Precondición	El cliente debe estar registrado en el sistema con el perfil adecuado.
Secuencia Normal: (Escenario exitoso)	1.- El cliente ingresa la cédula y la clave de su tarjeta. 2.- El WebSite Diners valida parámetros ingresados por el cliente, emitiendo cliente aceptado o cliente rechazado.
Excepciones:	Cliente no registrado.
Frecuencia Esperada:	Cada vez que el cliente solicite.
Comentarios:	Es el primer escenario a desarrollarse en el proceso de consulta del estado de cuenta. Sin este escenario no es posible desarrollar los escenarios subsecuentes.

Tabla 4.2: (Descripción del Caso de Uso: Tarjeta Habiente)

Nombre del caso de uso	Consulta de Estado de Cuenta
Actor Principal:	Tarjeta Habiente.
Actores involucrados	Tarjeta Habiente: Consultar su estado de cuenta. Web Service: Envía el estado de cuenta.
Objetivos	Obtener el estado de cuenta del tarjeta habiente.
Precondición	Ninguna.
Secuencia Normal: (Escenario exitoso)	1.-El tarjeta habiente solicita el estado de cuenta. 2.-El WS envía el estado de cuenta pedido por el TH.
Excepciones:	Ninguna.
Frecuencia Esperada:	Cada vez que el tarjeta habiente peticione.
Comentarios:	El tarjeta habiente recibe el estado de cuenta emitido y el proceso concluye.

4.1.4 Diagrama de Actividades

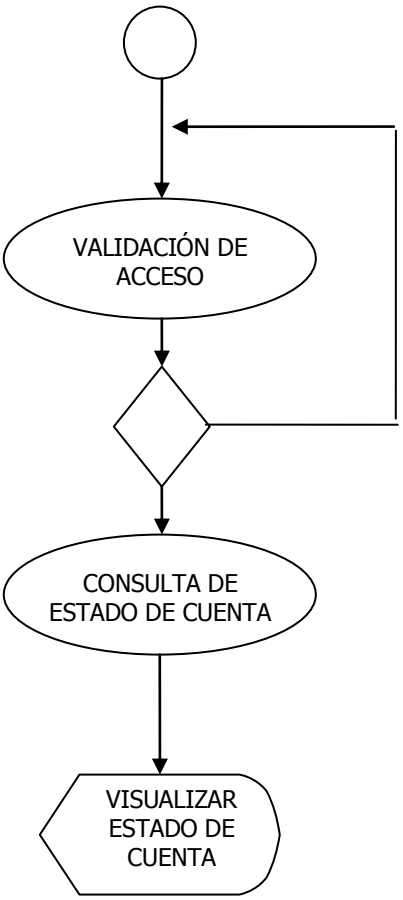


Fig. 4.2: (Diagrama de Actividades de la "Consulta del Estado de Cuenta Diners")

4.1.5 Diagrama de Interacción (Secuencias)

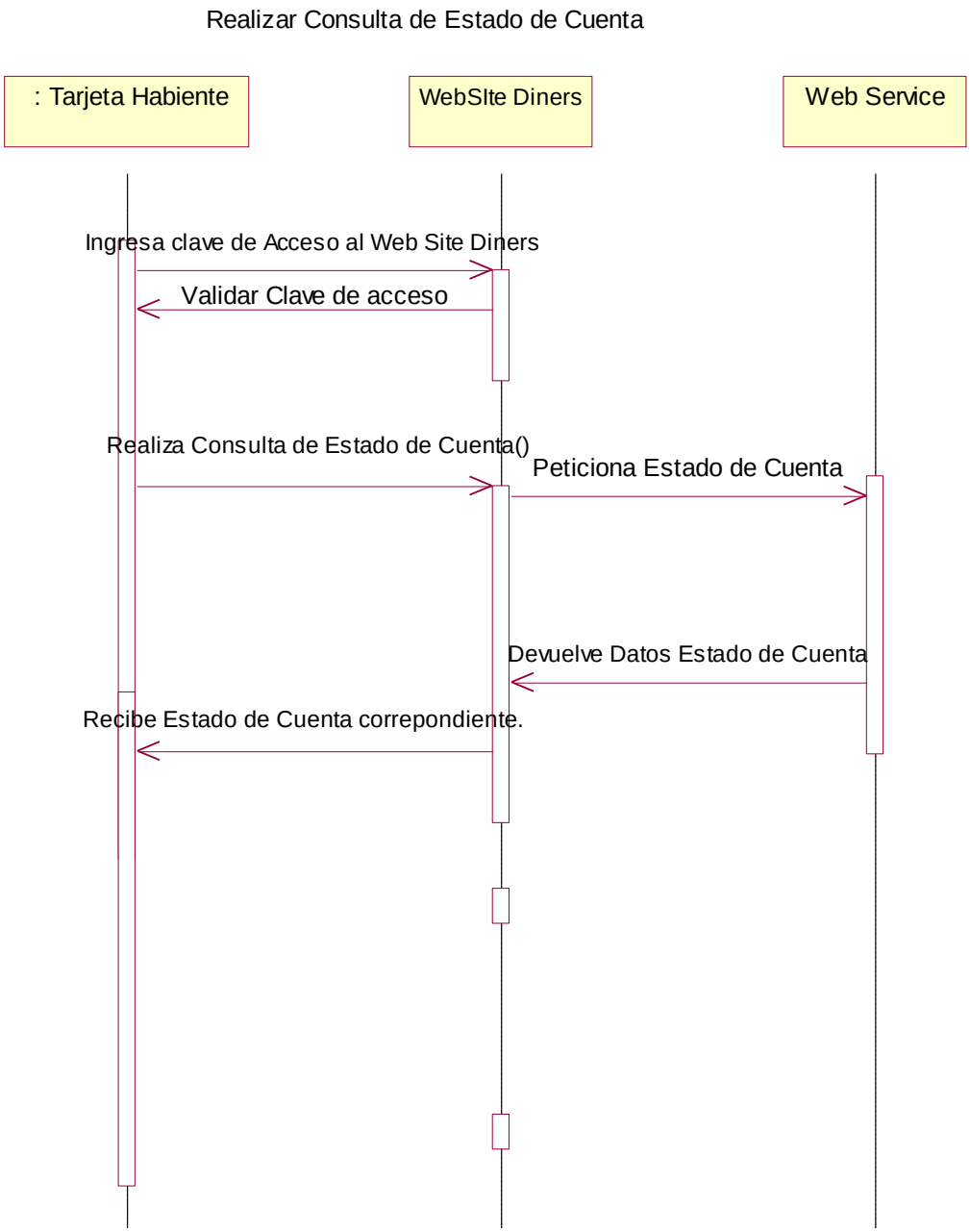


Fig. 4.3: (Diagrama de Secuencias de la "Consulta del Estado de Cuenta Diners")

4.1.6 Modelo Plex

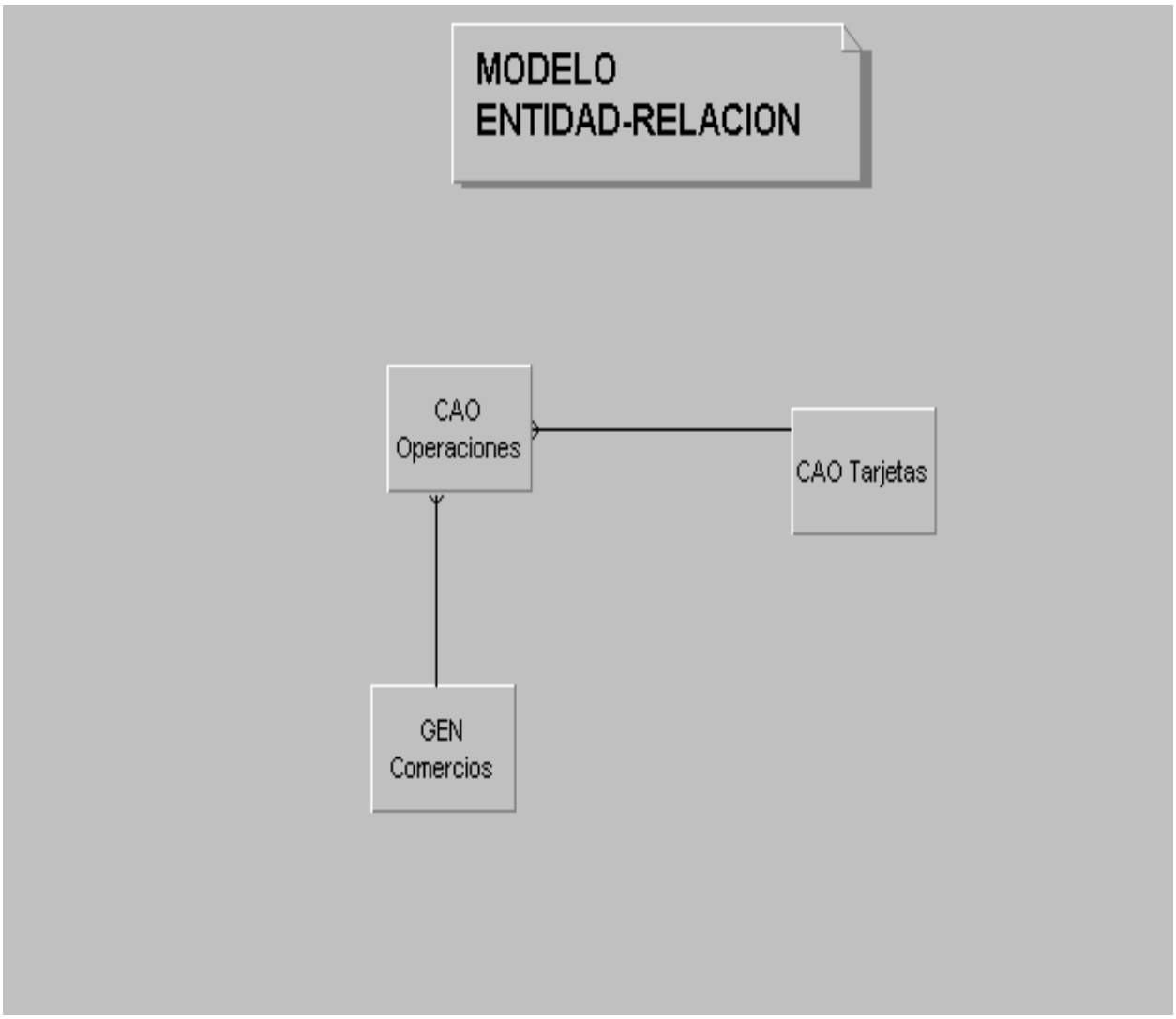


Fig. 4.4: (Modelo Plex)

4.1.7 Diagrama Físico

Para una mejor apreciación de la implementación del Web Service, se ha definido 4 máquinas para la puesta en marcha del sistema piloto como se muestra en el siguiente gráfico.

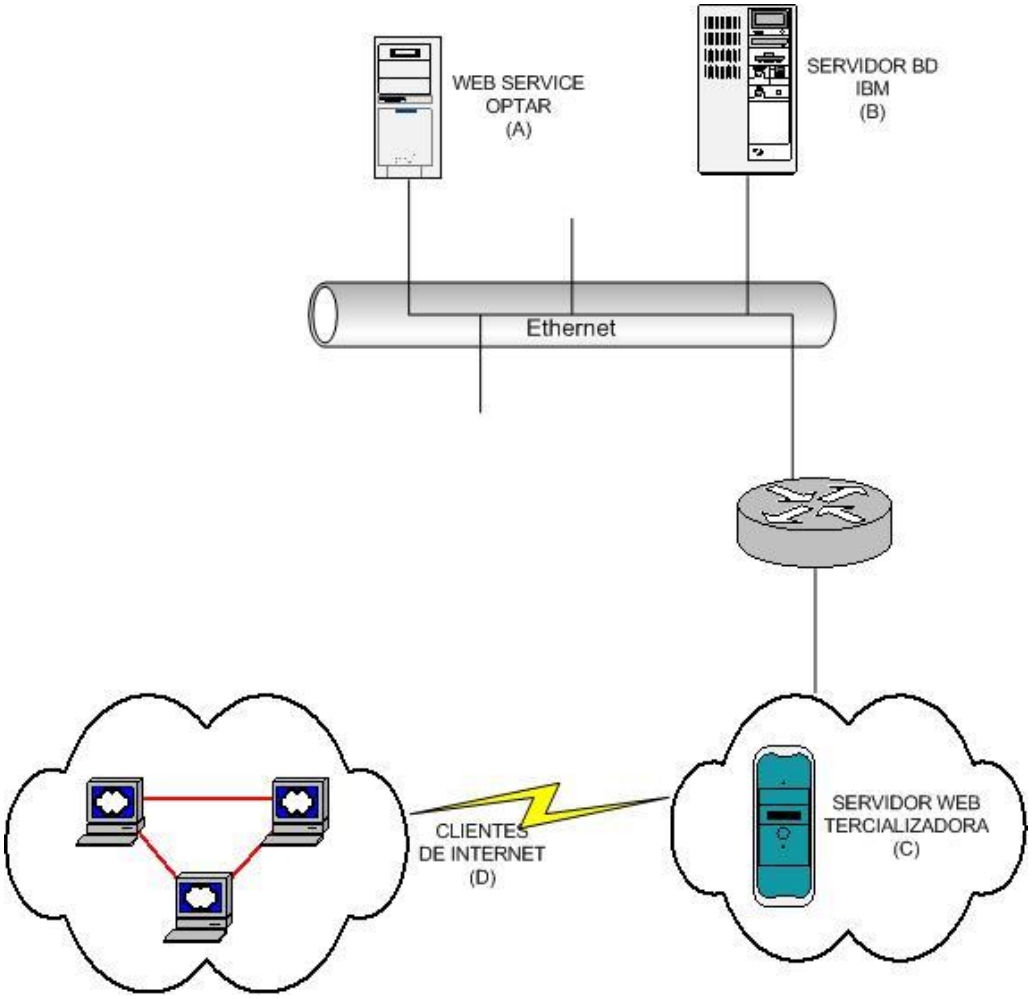


Fig. 4.5: (Diagrama físico del funcionamiento del Web Service)

En la máquina A, se encuentra el Web Service y todas las librerías necesarias del Plex y Websydian para su ejecución. Además de estar alojado el Web Service, se encuentra la Base de Datos con toda la información de los estados de cuenta del tarjeta habiente de Diners Club.

Para la ejecución normal del Web Service, se necesita tener instalado como servidor web el Apache en modo consola.

La base de datos que reside en el servidor de IBM (AS400), se encuentra localizado en la misma red del residente del Web Service, denominado máquina B.

En la máquina C, se encuentra el invocador del Web Service desarrollado en .Net y como servidor web el IIS (Internet Information Server). Además se encuentra alojado la Base de Datos SQL Server para el almacenamiento temporal de cada cuenta de estado por tarjeta y para la validación de ingreso al Web Site.

En la máquina D, simplemente es un computador común con el Internet Explorer como navegador, el cual realiza la tarea de consultor externo de la página Web residente en la máquina C.

Cuando el consultor navega e ingresa con su usuario y clave al Web Site de la tercerizadora de Diners, invocará la petición del estado de cuenta al Web Service residente en la máquina A, y a través del protocolo Soap como transporte enviará y regresará en formato XML la respuesta del mismo, el cual se visualizará de forma transparente para el usuario final (máquina D).

4.2 Análisis de la Configuración Plex

4.2.1 Lenguaje

Después de realizar la investigación necesaria de la herramienta Plex se concluyó que es más flexible y menos costoso trabajar sobre un modelo basado en WinC-RPG , del cual se ha visto las siguientes características:

- Por tener un servidor nativo sobre lenguaje RPG
- La conexión WinC-RPG es mucho más rápida que Java-Java ó Java-RPG.
- La implementación WinC-RPG no necesita un servidor específico para Websydian cuando se trabaja sobre lenguaje Java.
- Las opciones que nos provee Websydian sobre la plataforma WinC-RPG son más diversas.

4.2.2 Servidores

El Servidor es un AS400 con sistema operativo OS400 y una base de datos DB2 mediante lenguaje RPG. El Servidor Web es un As400 con sistema operativo OS400 con Websphere.

Por lo tanto, la interacción entre estos dos servidores son compatibles entre sí, sin la necesidad de realizar algún cambio en ellos.

Para nuestro modelo realizado en Plex se deberá tomar todos los puntos de configuración hacia estos servidores.

4.3 Diseño de la Arquitectura WINC/RPG

4.3.1 Ambiente AS-400

Advantage Plex maneja varias configuraciones multicapa dentro del AS-400, como son:

- WinC – RPG vía TCP-IP
- Java – RPG vía TCP-IP
- Browser (Websydian) – RPG

La configuración WINC - RPG trabaja con multiplataformas en la Web, el cual nos permite utilizar el puerto de comunicación número 51000 del equipo AS-400 que se ha creado dentro del servidor.

4.3.2 Librerías Plex

Las librerías del producto Advantage Plex que deben instalarse en el equipo AS-400, se encuentran en el CD de instalación del producto, estas son:

- Librería PLEX510 que contiene la configuración remote y objetos runtime del AS-400.
- Librería del Tutorial Advantage Plex: YTUTORIAL
- Librería de Referencia al tutorial Advantage Plex: YTUTREFER

4.3.3 Ambiente PC

La configuración se realiza en el entorno de la herramienta, y en archivos de configuración

4.3.3.1 Entorno de la Herramienta

- Levantar Plex y abrir el modelo local que está configurando o crear uno a partir de un modelo grupal cumpliendo con los estándares de alojamiento y nomenclatura.

- Abrir la ventana de Generación y Compilación, seleccionando del menú principal la opción Tools/Generate and Build.

- En el menú principal escoja la opción Build/Generate and Build Options.

- Definir el Sistema AS-400 al que se compilarán los objetos servidores
    - Seleccione la opción System Definitions.
    - Presione Add para añadir el sistema AS-400.

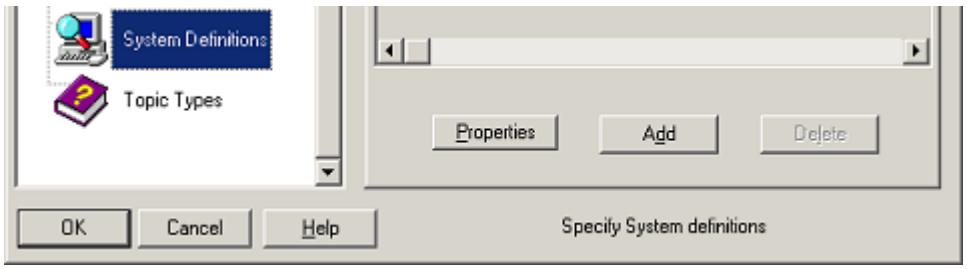


Fig. 4.6: (Configuración Plex: Definiciones del Sistema)

    - Ingrese el nombre del sistema, percatándose de escoger como tipo de sistema AS400.

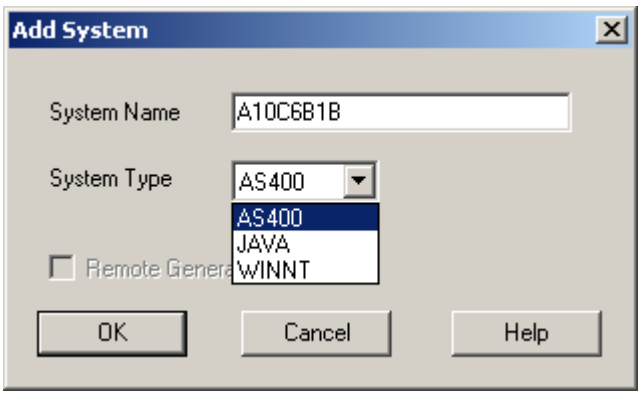


Fig. 4.7: (Configuración Plex: Tipo del Sistema)

- Presione OK para salir y grabar.
- Remarque el sistema añadido (clic sobre el sistema) y presione Properties.
- Seleccione la opción AS/400 Build para la configuración de Job Description y Cola de Trabajo del sistema.

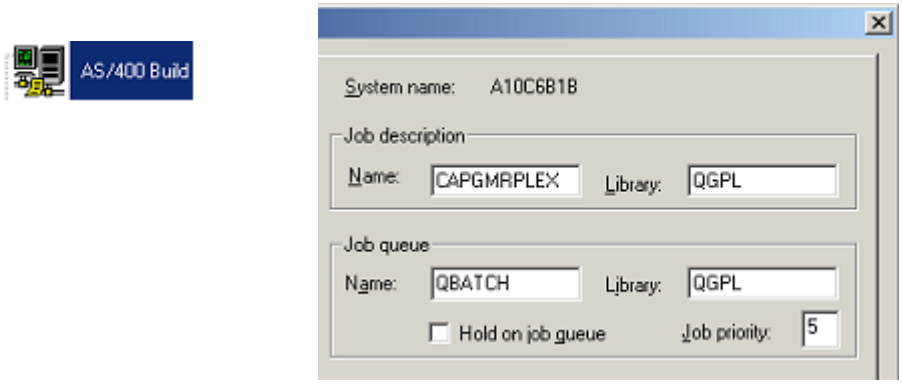


Fig. 4.8: (Configuración Plex: Configuración AS/400 - 1)

- Seleccionar la opción AS/400 Configuration para establecer la Job Description donde se encuentra instalado Plex en el AS-400, el protocolo y puerto de comunicación.

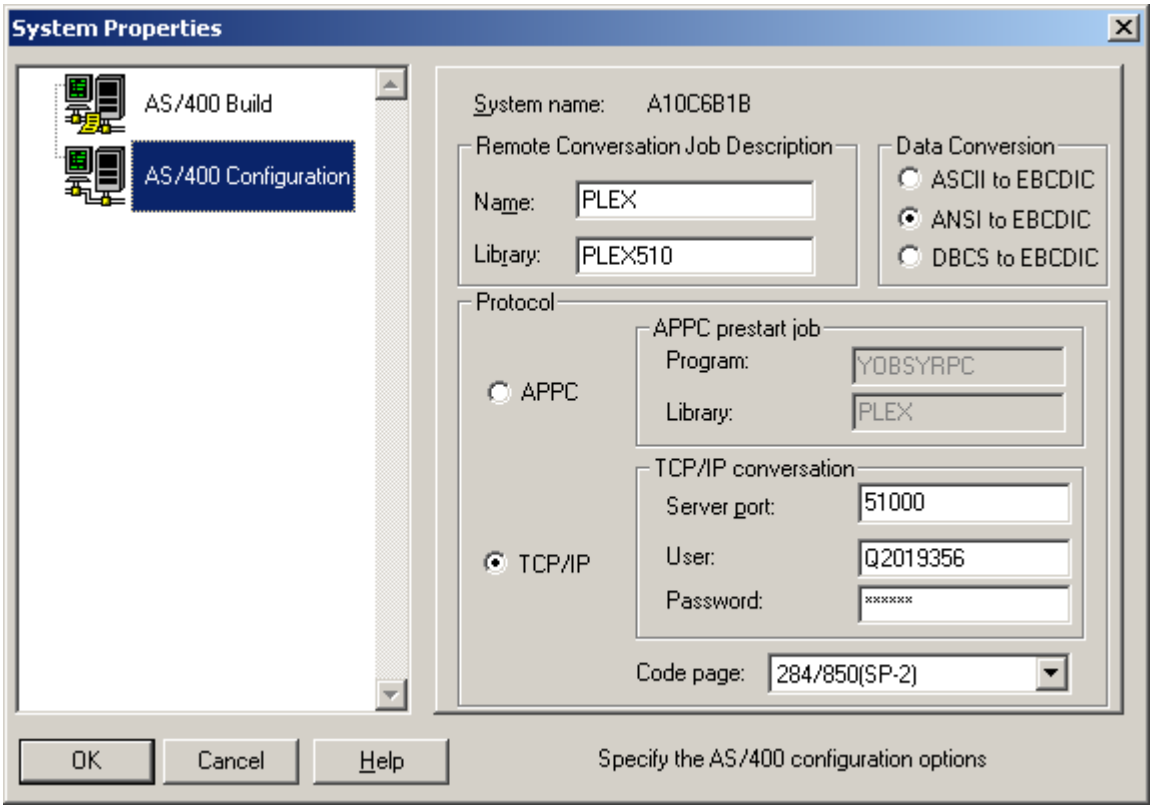


Fig. 4.9: (Configuración Plex: Configuración AS/400 - 2)

- Presione OK para salir y grabar.
- Mantener activa la pantalla de configuración Generate and Build Options.

- Definir los sistemas de generación y compilación de los objetos, dependiendo del lenguaje
- Para el lenguaje WinC se adopta el que por defecto indica Advantage Plex, es decir el sistema local.
- Para los objetos a compilarse en RPG, debe asignarse el sistema AS-400 definido en el paso anterior, para lo que debe proceder de la siguiente manera:
  - Escoger la opción Generate & Build Systems

- Remarcar el lenguaje RPG400... (Click sobre la línea específica).

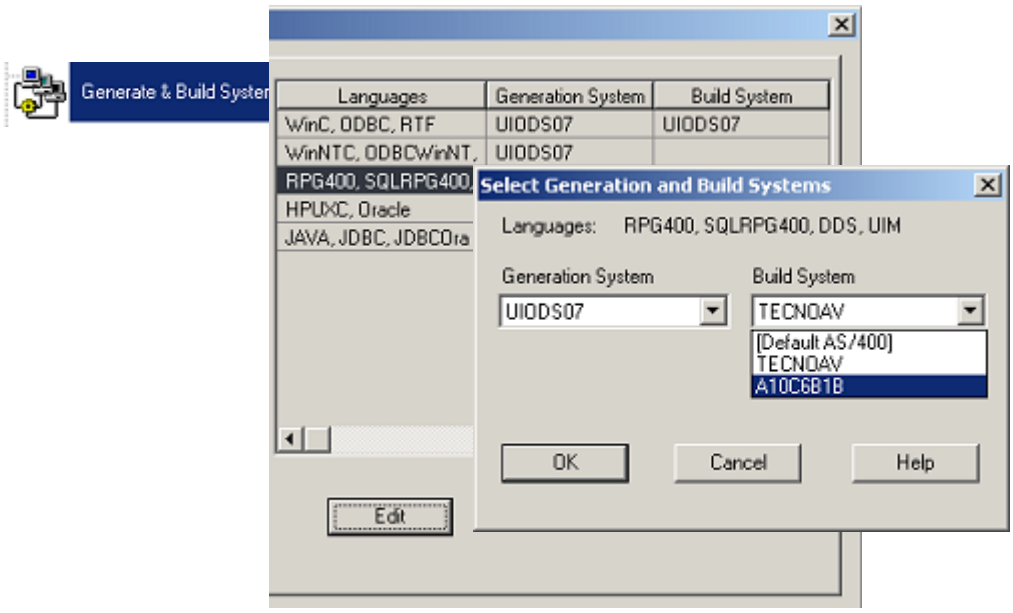


Fig. 4.10: (Configuración Plex: Configuración de la Generación del Sistema)

- Presione Edit, en la pantalla de diálogo, escoja el sistema A10C6B1B de entre los sistemas listados en el combo box.
- Presione OK para grabar y salir.

- Cierre la pantalla Generate and Build Options presionando OK para grabar y salir

- Elegir variantes y lenguajes: En el menú principal, elija la opción File/Configuration. En la ventana de diálogo, modificar las variantes remarcadas, con los valores como se indica en el gráfico.

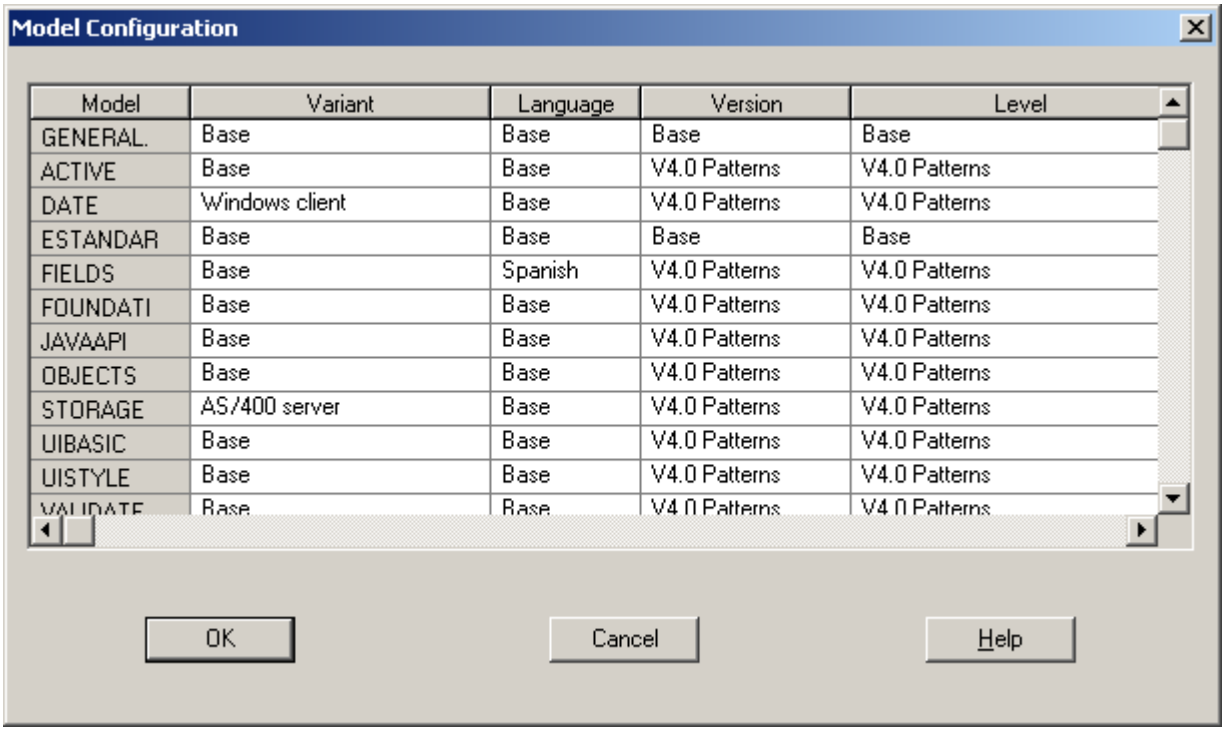


Fig. 4.11: (Configuración Plex: Configuración del Modelo)

4.3.3.2 Archivos de Configuración

- Editar el archivo HOSTS ubicado en el path C:\WINNT\system32\drivers\etc para así garantizar el acceso al sistema AS-400 definido en el entorno PLEX, aún cuando el PC no cumpla con la propiedad de resolución de nombres. Una vez realizada la configuración dentro del entrono de PLEX, se debe modificar: El archivo OB510RC.INI ubicado en la carpeta BIN en el directorio de instalación de PLEX, La sección a modificar es la denominada [REMOTE], como se ilustra en el gráfico siguiente:

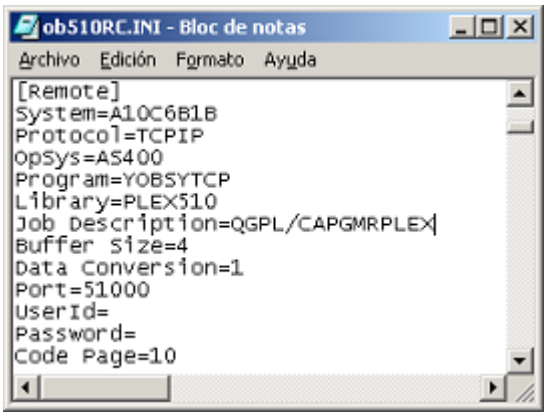


Fig. 4.12: (Configuración Plex: Configuración del archivo OB510RC.ini)

- Editar el archivo HOSTS ubicado en el path C:\WINNT\system32\drivers\etc para así garantizar el acceso al sistema AS-400 definido en el entorno PLEX, aún cuando el PC no cumpla con la propiedad de resolución de nombres.

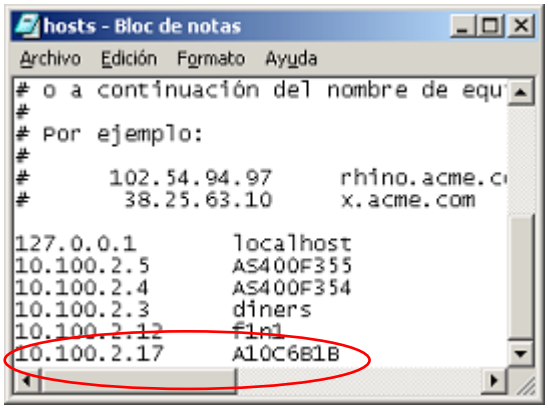


Fig. 4.13: (Configuración Plex: Configuración del archivo hosts)

4.3.3.3 Verificación de la Configuración del Puerto de Comunicación

Con el propósito de comprobar el estado de la comunicación con el equipo AS-400 vía TCP-IP, trabajando con su modelo local:

- Abrir la ventana de Generación y Compilación presionando el botón New Gen and Build (CTRL + G)

Fig. 4.14: (Configuración Plex: Configuración del puerto de comunicación - 1)

- De la barra de herramientas, AS-400 Build presione el botón AS-400 Build Status.

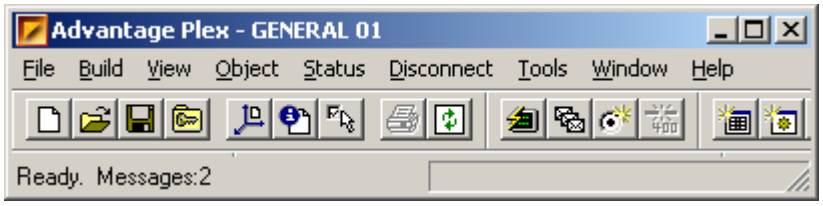


Fig. 4.15: (Configuración Plex: Configuración del puerto de comunicación - 2)

- Si la configuración no fue exitosa, Advantage Plex emite los errores producidos, a través del Log de Mensajes.

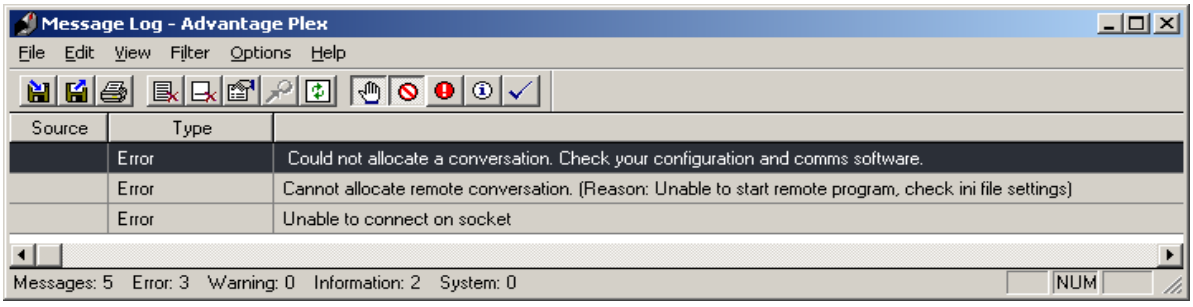


Fig. 4.16: (Configuración Plex: Configuración del puerto de comunicación - 3)

- De lo contrario se visualiza la pantalla de diálogo para los trabajos de compilación remota al AS-400.

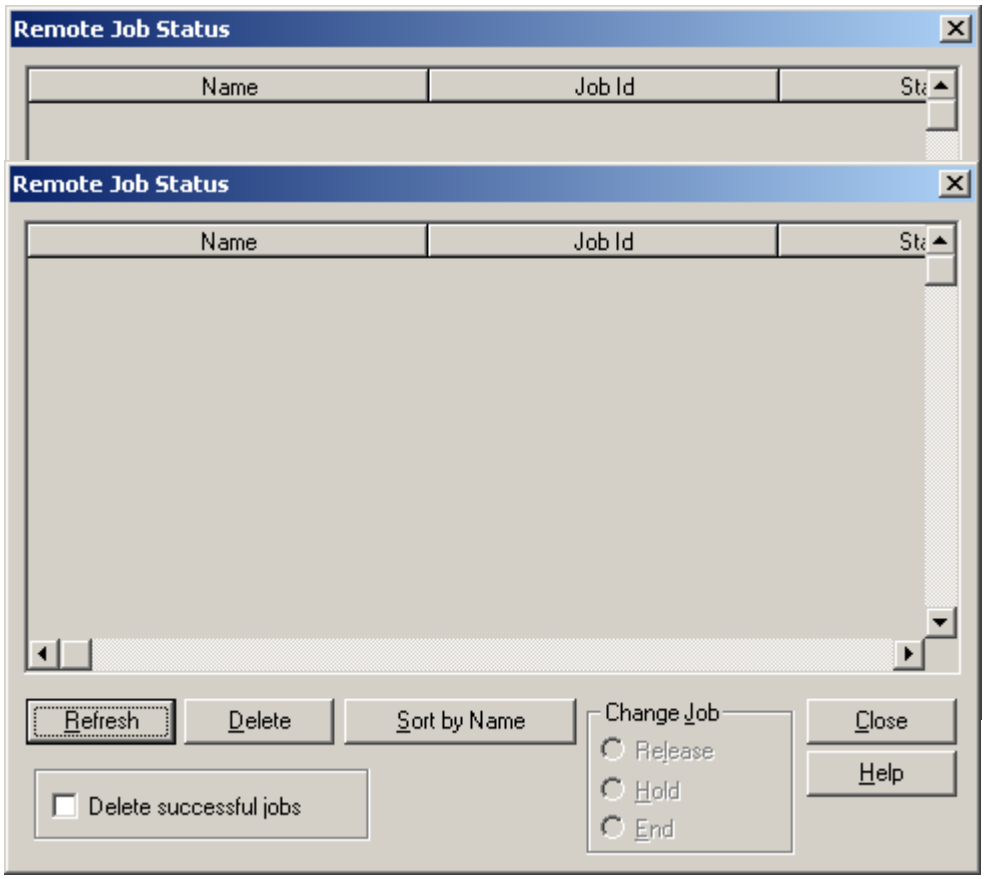


Fig. 4.17: (Configuración Plex: Configuración del puerto de comunicación - 4)

Hasta este momento, está completamente habilitado para trabajar con la herramienta Advantage Plex en la arquitectura WinC – RPG.

4.4 Diseño del Web Service Piloto

4.4.1 TransacXML

En este apartado observaremos la implementación de la importación y exportación de un archivo XML, paso previo a la implementación del Web Services.

Dentro de la funcionalidad específica en la exportación de un XML, un documento XML es creado y alimentado con datos desde una simple base de datos. El proceso de la exportación se encuentra ilustrado en la figura 4.18.

Dentro de la funcionalidad específica en la importación de un XML, una simple base de datos es alimentado con datos desde un documento XML. Éste Proceso es una prueba equivalente al proceso de importación que se ilustra en la figura 4.18.

El TransacXML ofrecerá completo soporte para el Web Service, el cual es relativamente nuevo en la manera de implementar intercambios de información de sitios independientes de lenguajes de plataformas.

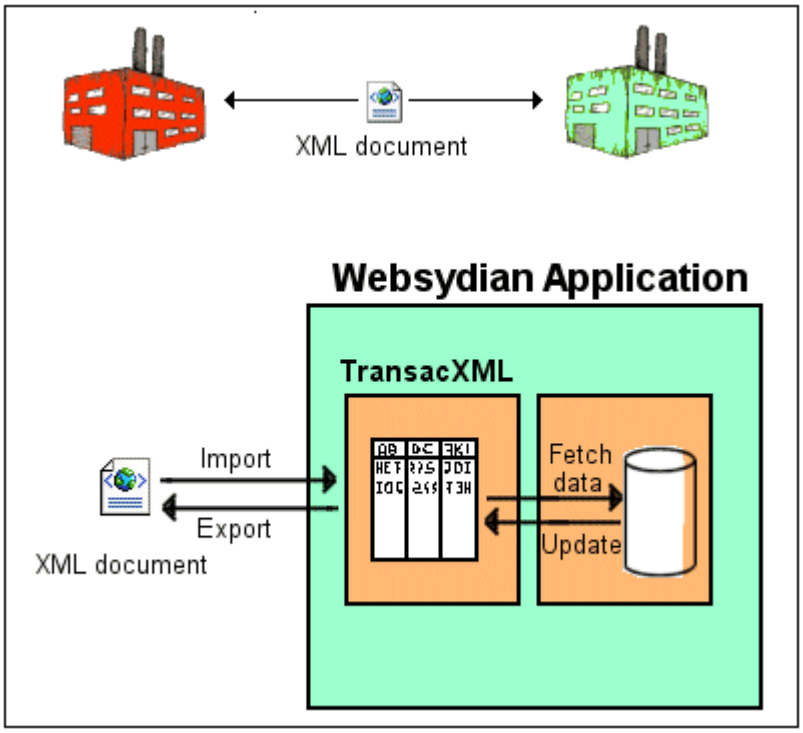


Fig. 4.18: (TransacXML: Ilustración de la importación y exportación de un XML)

El módulo del TransacXML habilita al desarrollador del Websydian a acceder datos dentro de un documento XML usando **well-known** funciones de Base de Datos relacionales tal como ***SingleFetch*** y ***ProcessGroup***.

4.4.1.1 Contexto de la Aplicación

En este apartado se implementa la funcionalidad para usar en la parte del desarrollo del Web Service. El TransacXML permitirá al desarrollador intercambiar datos entre documentos XML y la aplicación. Se importará y exportará documentos XML, demostrados en simples ejemplos.

En esta aplicación de TransacXML una entidad de la base de datos contendrá la información de los estado de cuenta poseídos por Optar (tarjeta Diners).

4.4.1.2 Configuración del Modelo PLex

Asegúrese de añadir las siguientes librerías del Websydian:

- WSYDom
- WSYXml
- WsySoap

Especificar la variante **WSYDOM** para la variante **MSXML** (sobre la plataforma Windows)

Asegúrese de generar y construir los objetos bajo las áreas de:

- WSYDOM/DomObjectsToGenerateAndBuild
- SDSTRING/SDStringObjsToGenerateAndBuild
- WSYXML/XMLObjextToGenerateAndBuild

4.4.1.2.1 Dependencias del Analizador Gramatical XML

Con el propósito de trabajar a través de la versión de Windows, se necesita obtener la última versión del Microsoft XML Parser 4.0

4.4.1.2.2 Estructura de la Base de Datos

Una simple base de datos es definida, conteniendo información acerca del saldo de la tarjeta habiente Diners por Optar. La base de datos es definida por las tripletas definidas posteriormente. La entidad Data almacena toda la base de datos relacionando la funcionalidad:

Tabla 4.3: (TransacXML: Tripleta de la base de datos)

Source Object	Verb	Target Object
Data	includes ENT	Optar
Data.Optar	includes ENT	Item
Data.Optar.Item	is a	RelationalTable EditDialog
	known by FLD	NumOper
	has FLD	Tarjeta FechaOper

Añadir las siguientes tripletas para especificar cada campo:

Tabla 4.4: (TransacXML: Tripleta por campos específicos)

Source Object	Verb	Target Object
ItemID	is a FLD	Identifier
	length NBR	40
Tarjeta	is a FLD	ShortDescription
	length NBR	40
NumOper	is a FLD	Int
FechaOper	is a FLD	Date

Especificar la siguiente tripleta para definir la Base de Datos del Broker de todos los Ítems o campos.

Tabla 4.5: (TransacXML: Base de Datos del Broker para todos los ítems o campos)

Source Object	Verb	Target Object
Data	includes ENT	Broker
Data.Broker	includes ENT	All Items
Data.Broker.All Items	is a	RelationalTable EditDialog
	known by FLD	NumOper

	has FLD	Tarjeta FechaOper

Especificar la siguiente tripleta para definir la base de datos del Broker Returned Items

Tabla 4.6: (TransacXML: Base de Datos del Broker para campos retornados)

Source Object	Verb	Target Object
Data.Broker	includes ENT	Returned Items
Data.Broker.Returned Items	is a	RelationalTable EditDialog
	owned by ENT	Data.Broker.All Items
	has FLD	Tarjeta FechaOper Comercio Monto Corriente Diferido

4.4.1.2.3 Modelamiento de los Documentos XML en el Modelo Plex

Para esta aplicación se implementará los documentos ItemList y ReturnOffer.

El modelamiento de un documento XML dentro del Plex puede ser realizado desde un número diferente de recursos, diagramas, DTD o documentos XML existentes. Éste último es el que se utilizará para la aplicación.

El documento XML ItemList representa los parámetros de entrada al proceso. Mientras que el documento XML ReturnOffer es la respuesta a la pregunta Cuál es mi estado de cuenta de este mes?.

A continuación se muestra el documento XML para ItemList y ReturnOffer:

Tabla 4.7: (TransacXML: Documento XML para ItemList)

XML Document for Item List	**DTD for ItemList**
<?xml version='1.0'?> <ItemList> <Item NumOper="1"> <Tarjeta>1</Tarjeta> </Item> </ItemList>	<!ELEMENT ItemList (Item*)> <!ELEMENT Item (Tarjeta)> <!ATTLIST Item NumOper CDATA> <!ELEMENT Tarjeta (#PCDATA)>

Tabla 4.8: (TransacXML: Documento XML para ReturnOffer)

XML Document for ReturnOffer	**DTD for ReturnOffer**
<?xml version='1.0'?> <ReturnOffer> <Item NumOper="1"> <Tarjeta>1</Tarjeta> <FechaOper>09/09/2004</FechaOper> <Cliente>1</Cliente> <Comercio>1</Comercio> <Monto>1</Monto> <Corriente>1</Corriente> <Diferido>1</ Diferido > </Item> </ReturnOffer>	<!ELEMENT ReturnOffer (Item*)> <!ELEMENT Item (Tarjeta,FechaOper,Cliente,Comercio,Monto,Corriente, Diferido)> <!ATTLIST Item NumOper CDATA> <!ELEMENT Tarjeta (#PCDATA) <!ELEMENT Comercio (#PCDATA) <!ELEMENT Monto (#PCDATA) <!ELEMENT Corriente (#PCDATA) <!ELEMENT Diferido (#PCDATA)

Con el propósito de usar los patrones del TransacXML, el documento XML debe ser representado en el modelo Plex como entidad y campos de objetos (de esta forma éste posiblemente accederá al contenido del documento XML como si fuera ellos los representados en tablas relacionales).

Añadir las siguientes tripletas para describir los dos documentos XML usados para esta aplicación, primero el documento XML ItemList (La entidad CommonXML abarca los diferentes documentos XML).

Tabla 4.9: (TransacXML: Tripleta para la función ItemList)

Source Object	Verb	Target Object
CommonXml	Includes ENT	ItemList
CommonXml.ItemList	is a ENT	XmlElement
	Includes ENT	Item
CommonXml.ItemList.Item	is a ENT	XmlRepeatingElement
CommonXML.ItemList.Item.Fields	field FLD	NumOper Tarjeta
CommonXml.ItemList.Item	Has	Fields.NumOper Fields.Tarjeta
CommonXML.ItemList.Item.Fields.NumOper	is a FLD	AttributeField
CommonXML.ItemList.Item.Fields.Tarjeta	is a FLD	ElementField

Además el documento XML ReturnOffer.

Tabla 4.10: (TransacXML: Tripleta para la función ReturnOffer)

Source Object	Verb	Target Object
CommonXML	includes ENT	ReturnOffer
CommonXml.ReturnOffer	is a ENT	XmlElement
	includes ENT	Item
CommonXml.ReturnOffer.Item	is a ENT	XmlRepeatingElement
CommonXML.ReturnOffer. Item.Fields	field FLD	NumOper Tarjeta FechaOper Monto Comercio Diferido
CommonXML.ReturnOffer. Item.Fields.NumOper	is a FLD	AttributeField
CommonXML.ReturnOffer.	is a FLD	ElementField

Item.Fields.Tarjeta		
CommonXML.ReturnOffer. Item.Fields.FechaOper	is a FLD	ElementField
CommonXML.ReturnOffer. Item.Fields.Monto	is a FLD	ElementField
CommonXML.ReturnOffer. Item.Fields.Corriente	is a FLD	ElementField
CommonXML.ReturnOffer. Item.Fields.Diferido	is a FLD	ElementField
CommonXml.ReturnOffer.Item	has	Fields.NumOper Fields.Tarjeta Fields.FechaOper Fields.Monto Fields.Corriente Fields.Diferido

4.4.1.2.4 Paquetes de Librerías y Funciones

Asegurarse que las funciones provistas por el Websydian abarquen las siguientes áreas que serán generadas y construidas.

- WSYDOM/DomObjectsToGenerateAndBuild
- SDSTRING/SDStringObjsToGenerateAndBuild
- WSYXML/XMLObjectsToGenerateAndBuild

Generar y construir lo siguiente:

Todo lo que abarque a **Data**

Todo lo que abarca a **Common XML**

Nota*:* Usando la variante MSXML, asegúrese que el WSYDXML11.DLL haya sido copiado al directorio Release.

4.4.1.2.5 Especificación de la Funcionalidad de Exportación

Por medio de la aplicación de base de datos y la estructura del documento XML especificado en el modelo Plex, ahora se debe especificar la funcionalidad que exportará los datos desde la base de datos a un documento XML.

4.4.1.2.6 Exportación de los Datos dentro del Documento ReturnOffer

En este caso, existiendo un documento XML conteniendo el parámetro de la Tarjeta llamada "Ítem" es procesada, y un nuevo documento XML es creado con información extendida acerca de todas las transacciones que se ha realizado por dicho número de tarjeta en la entrada del documento.

Especificar tripletas para definir el ExportToReturnOffer.

Tabla 4.11: (TransacXML: Tripleta para la función ExportToReturnOffer)

Source Object	Verb	Target Object
ExportToReturnOffer	is a FNC	FunctionShell DomServerExternal
	input FLD ...for VAR	ObjectStoreReference InputDoc
	input FLD ..for VAR	ObjectDocument InputDoc
	input FLD ...for VAR	ObjectStoreReference OutputDoc
	input FLD ...for VAR	ObjectDocument OutputDoc
	local FLD	ItemID Int ExceptionCode
	local view VW ..for VAR	CommonXML.ReturnOffer.item.Data WSYXML/OutputDocument
	local FLD ..for VAR	ObjectStoreReference WSYXML/OutputDocument

	local FLD ...for VAR	ObjectElement WSYXML/OutputDocument
	local FLD ...for VAR	ObjectDocument WSYXML/OutputDocument
	local FLD ..for VAR	ObjectStoreReference InputDocument
	local FLD ...for VAR	ObjectElement InputDocument
	local FLD ...for VAR	ObjectDocument InputDocument
	file name NME	ExportRO
	impl name NME	ExportRO

Abrir el diagrama de acción para el ExportToReturnOffer e insertar el siguiente código dentro del específico Edit Points y subrutinas.

Cuadro 4.1: (TransacXML: Diagrama de acción para el ExportToReturnOffer)

Pre Point Subroutines

Sub Initialize XML input document
Sub Initialize XML output document
Sub Loop through requested items
Sub Error check

Pre Point Execute

Go Sub Initialize XML input document
Go Sub Initialize XML output document
Go Sub Loop through requested items
Go Sub Terminate

Sub Initialize XML input document

Set InputDocument = InputDoc
Sub Initialize XML output document
Set OutputDocument = OutputDoc
Call CommonXML.ReturnOffer.InsertRow
// Map OutputDocument<ObjectStoreReference>
// Map OutputDocument<ObjectDocument>
// Map OutputDocument<ObjectDocument>
Go Sub Error check

```
Set OutputDocument<ObjectElement> =
CommonXML.ReturnOffer.InsertRow/Output<ObjectElement>

Sub Loop through requested ítems

Call CommonXML.ItemList.Item.GetFirstOccurrence
  // Map InputDocument<ObjectStoreReference>
  // Map InputDocument<ObjectDocument>
  // Map <ParentElement.NULL>
Set InputDocument<ObjectElement> =
CommonXML.ItemList.Item.GetFirstOccurrence/Output<ObjectNode>
Go Sub Error check

While InputDocument<ObjectElement> != <ObjectElement.NULL>
  Call CommonXML.ItemList.Item.SingleFetch
    // Map InputDocument<ObjectStoreReference>
    // Map InputDocument<ObjectElement>
  Set InputDocument<ObjectElement> =
CommonXML.ItemList.Item.SingleFetch/Output<NextOccurrence>
  Go Sub Error check

  Cast Local<NumOper>, Data<CommonXML.ItemList.Item.Fields.NumOPer>
  Call Data.Optar.Item.Fetch.SingleFetch
    // Map Local<NumOper>
  Go Sub Error check

  Cast OutputDocument<CommonXML.ReturnOffer.Item.Fields.Tarjeta>,
FetchedData<Tarjeta>
  Cast OutputDocument<CommonXML.ReturnOffer.Item.Fields.NumOper>,
FetchedData<NumOper>
  Cast OutputDocument<CommonXML.ReturnOffer.Item.Fields.FechaOper>,
FetchedData<FechaOper>
Cast OutputDocument<CommonXML.ReturnOffer.Item.Fields.Monto>,
FetchedData<Monto>
  Cast OutputDocument<CommonXML.ReturnOffer.Item.Fields.Comercio>,
FetchedData<Comercio>
  Cast OutputDocument<CommonXML.ReturnOffer.Item.Fields.Diferido>,
FetchedData<Diferido>

  Call CommonXML.ReturnOffer.Item.InsertRow
    // Map OutputDocument<ObjectStoreReference>
    // Map OutputDocument<ObjectElement>
    // Map OutputDocument<ObjectDocument>
    // Map OutputDocument<CommonXML.ReturnOffer.Item.Fields.Tarjeta>
 // Map OutputDocument<CommonXML.ReturnOffer.Item.Fields.NumOper>
 // Map OutputDocument<CommonXML.ReturnOffer.Item.Fields.FechaOper>
 // Map OutputDocument<CommonXML.ReturnOffer.Item.Fields.Monto>
```

```
// Map OutputDocument<CommonXML.ReturnOffer.Item.Fields.Comercio>
// Map OutputDocument<CommonXML.ReturnOffer.Item.Fields.Diferido>
 Go Sub Error check

Sub Error check
If Environment<*Returned status> IS <State: OBJECTS/*Returned
status.*Abnormal>
 Set Environment<*Returning status> = <*Returning status.*Error>
 Go Sub Terminate
```

Crear una simple función que abarque probar la función del ExportToReturnOffer.

Tabla 4.12: (TransacXML: Tripleta para probar la función ExportToReturnOffer)

Source Object	Verb	Target Object
TestExportToReturnOffer	is a FNC	WSYXML/ImportXMLDocument WSYXML/ExportXMLDocument
	impl name NME	TestExRO
	file name NME	TestExRO
	message MSG	InputDocument OutputDocument

- Especificar el contenido del mensaje en **InputDocument** para c:\temp\Request.xml
- Especificar el contenido del mensaje en **OutputDocument** para c:\temp\Response.xml

Abrir el diagrama de acción para **TestExportToReturnOffer** e insertar el siguiente código dentro del específico Edit Points y Sub Rutinas.

Cuadro 4.2: (TransacXML: Diagrama de acción para el ExportToReturnOffer)

```
Edit Point Process XML document

Call ExportToReturnOffer
  // Map Local/InputDocument<ObjectStoreReference>
  // Map Local/InputDocument<ObjectDocument>
  // Map Local/OutputDocument<ObjectStoreReference>
  // Map Local/OutputDocument<ObjectDocument>
```

Format Message Message: TestExportToReturnOffer.OutputDocument, Local/OutputDocument<FileName>

Pre Point Start load input document

Format Message Message: TestExportToReturnOffer.InputDocument, Local/InputDocument<FileName>

Post Point Handle error message

Dialog Message Message: OBJECTS/Message
// Map Environment<*Message text>

4.4.1.2.7 Generación y Construcción de las Funciones

Se procede a generar y construir las funciones realizadas:

- ExportToReturnOffer
- TestExportToReturnOffer

Crear un archivo EXE para la función **TestExportToReturnOffer**

Como resultado de correr el TestExport, el archivo llamado Response.xml ahora existe en c:\temp, como lo siguiente:

Cuadro 4.3: (TransacXML: Documento Response.xml)

```
<?xml version="1.0" encoding="utf-8" ?>
- <WSDiners:ReturnOffer xmlns:WSDiners="http://localhost/webtutor/WSDiners">
- <WSDiners:Item NumOper="1">
  <WSDiners:Tarjeta>1</WSDiners:Tarjeta>
  <WSDiners:FechaOp>2004-10-12</WSDiners:FechaOp>
  <WSDiners:Cliente>1</WSDiners:Cliente>
  <WSDiners:Comercio>1</WSDiners:Comercio>
```

```
        <WSDiners:Monto>1</WSDiners:Monto>
        <WSDiners:Corriente>1</WSDiners:Corriente>
        <WSDiners:Diferido>1</WSDiners:Diferido>
          </WSDiners:Item>
      - <WSDiners:Item NumOper="2">
        <WSDiners:Tarjeta>2</WSDiners:Tarjeta>
        <WSDiners:FechaOp>2012-10-12</WSDiners:FechaOp>
        <WSDiners:Cliente>2</WSDiners:Cliente>
        <WSDiners:Comercio>2</WSDiners:Comercio>
        <WSDiners:Monto>2</WSDiners:Monto>
        <WSDiners:Corriente>2</WSDiners:Corriente>
        <WSDiners:Diferido>2</WSDiners:Diferido>
          </WSDiners:Item>
  </WSDiners:ReturnOffer>
```

4.4.2 WsySoap

WsySoap es el módulo de implementación del Web Services luego de haber descrito el módulo del TransacXML que es indispensable para dicha elaboración. Este módulo es dividido en dos secciones, el primero cubre la implementación de la aplicación del Publicador, y el segundo la implementación de la aplicación del Subscriptor.

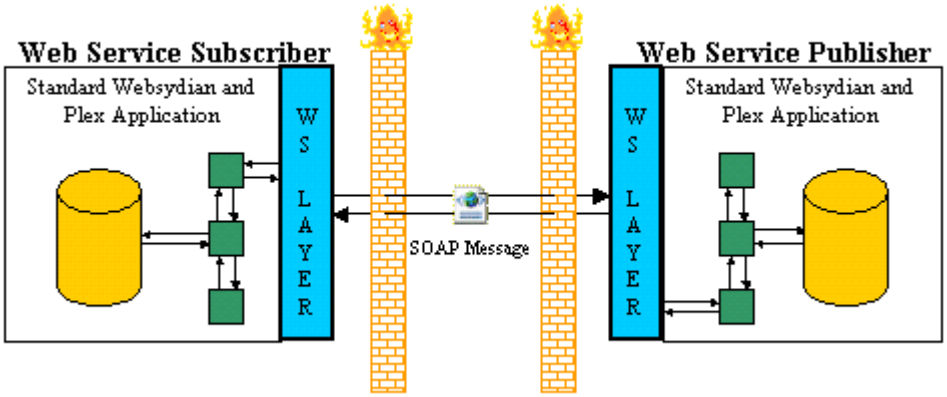


Fig. 4.19: (Mensaje Soap en el Web Service)

4.4.2.1 Contexto de la Aplicación

El Publicador del Web Service es nuestro caso es OPTAR, el cual realiza todas la transacciones de las tarjetas de crédito de Diners Club, mientras que el Subscriptor del Web Service es el intermediario (broker) que se encarga de mostrar toda la información de los estados de cuenta Diners a través de su Web Site. El Broker procesa la información retornada y presenta los resultados (estado de cuenta) para el usuario del servicio Broker.

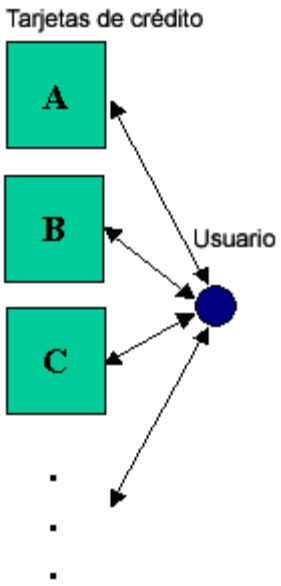
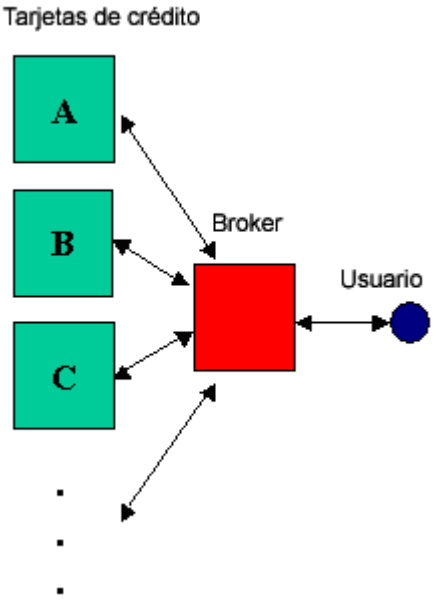
Método Estándar de la Web	Uso del Broker con Web Services
	
El usuario va a los sitios web de cada uno de las tarjetas de crédito o bancos	El usuario va al sitio web del Broker que colecciona toda la información de las tarjetas de crédito.

Fig. 4.20: (Uso del Broker con Web Service)

La metodología estándar en la Web es un método de interacción entre el usuario y los dueños de las tarjetas de crédito o bancos.

En el método del Web Service interactúan los usuarios y las tarjetas de crédito, es una mezcla del browser basado en comunicación (entre el usuario y el Broker) y los Web Service (entre el Broker y las tarjetas de crédito). En otras palabras una capa de la comunicación del Web Service es insertada entre el usuario y las tarjetas de crédito, pero sin que el usuario requiera hacer cambios en sus costumbres o hábitos y sin conocer a los Web Services.

En este sistema, la comunicación entre el Broker y una única tarjeta de crédito es implementado. El Broker y Diners Club interactúan usando Web Services, donde Diners o la tarjeta de crédito actúa como el Publicador, mientras que el Broker actúa como el Subscriptor para el Web Service.

Una función del Web Service será implementado para la tarjeta de crédito, el cual retornará toda la información del estado de cuenta de la tarjeta habiente Diners.

El Broker llamará a la función Web Service ReturnAll. Éste retornará la información del estado de cuenta que será importada dentro de la base de datos del Broker.

El Broker y Diners intercambiarán datos usando documentos XML y Web Services. La operabilidad de la importación y exportación usada para la implementación el Web Service está descrita en la implementación del TransacXML que se especificó en el punto anterior.

4.4.2.1.1 Diferencias entre la implementación del TransacXML y la implementación del Web Services

En la figura 4.21 se muestra la exportación desde la Base de Datos de Diners a un documento XML con la importación subsecuente dentro de la Base de Datos del Broker es mostrada como éste actúa en la implementación del TransacXML. Como el documento XML resulta desde el primer paso es accesible para la importación en el segundo paso que no está descrito; éste podría almacenar en un medio removible, email o copiado vía Internet. Los agentes invocarán dentro de la operación de copiar datos desde Diners al Broker que son los responsables de transferir el documento XML.

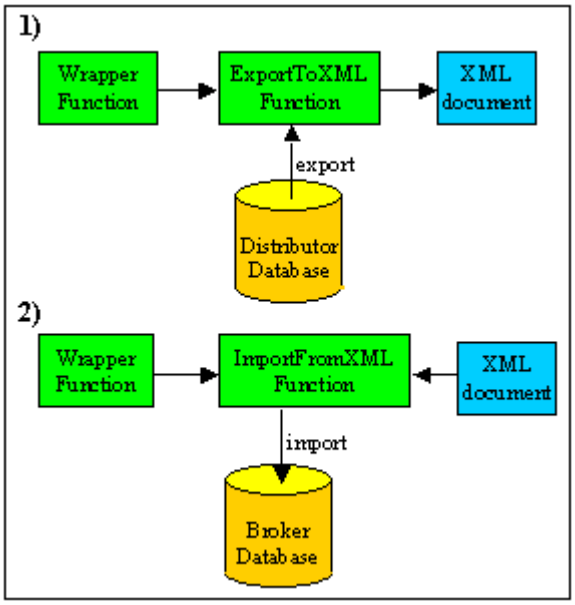


Fig. 4.21: (Transferencia del XML en Web Services)

Usando la simple funcionalidad del TransacXML, intercambian los datos entre el sistema disparado, que es el procedimiento del segundo paso, donde el documento XML comienza a ser usado como transporte de la información que debería de algún modo ser transferido desde un sistema al otro.

En la siguiente figura 4.22, se muestra como se exportará desde la Base de Datos de Diners a la Base de Datos del Broker usando Web Services. Avisa que el Web Service a través de las funciones SOAP ambos maneja la transferencia del documento XML y la invocación de la funcionalidad remota. En esta aplicación es el Broker quien solicita a Diners exportar datos retornar el documento XML a través del Web Services.

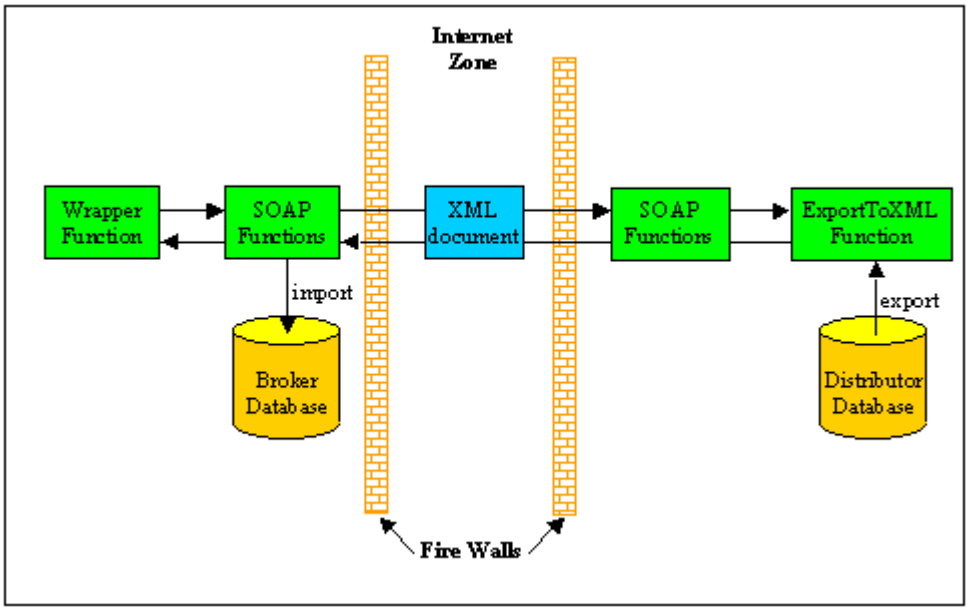


Fig. 4.22: (Transferencia del XML en Web Services)

El Broker inicializa el proceso y Diners responde con un documento XML conteniendo la información solicitada (encapsulado dentro de la envoltura SOAP).

4.4.2.2 Elaboración

4.4.2.2.1 Edición de Namespace a los Documentos XML

Cuando se realiza los Web Services es recomendado añadir un namespace a los documentos XML usados para el Web Service, éste habilita el desarrollo para crear archivos WSDL bien formados.

Adicionar la siguiente tripleta al modelo

Tabla 4.13: (WSYSOAP: Tripleta Namespace)

Source Object	Verb	Target Object	Comments
CommonXML.ReturnOffer	is a ENT	NameSpaceAware	
CommonXML.ItemList	is a ENT	NameSpaceAware	

4.4.2.2.2 Implementación de la funcionalidad del Publicador (Diners)

El Publicador implementa la función del ReturnAll.

Especificar la entidad SOAP del WS

La entidad base para el Publicador es la entidad HttpSoap

Tabla 4.14: (WSYSOAP: Tripleta Entidad Soap)

Source Object	Verb	Target Object	Comments
Optar	is a ENT	HttpSoap	

Las funciones núcleo en la entidad HttpSoap son el SoapGenerator y el SoapProcessor. El SoapGenerator es usado por el Subscriptor para comunicarse con el Web Service, mientras que el SoapProcessor es usado por el Publicador para implementar el Web Service.

Optar (Diners) actúa solo como Publicador y no hace uso de la función SoapGenerator, el cual no debería ser generado.

Tabla 4.15: (WSYSOAP: Tripleta SoapProcessor)

Source Object	Verb	Target Object	Comments
Optar.Services.SoapProcessor	impl name NME	SOAPDisA	
	file name NME	SOAPDisA	
Optar.Services.SoapGenerator	implement SYS	NO	

Para probar la invocación del Web Service, primero debemos hacerlo de forma manual, para esto debemos crear el directorio c:\temp\SoapProcessor (éste es donde la función del SoapProcessor almacenará los archivos temporales)

Crear el ejecutable de la función del SoapProcessor.

Abrir el archivo SoapDisA.ini que se creó al momento de generar el .exe, y añadir las siguientes líneas:

```
[TransacXML]
MAX_CONTENT_LENGTH=10240
TEMPORARY_FILES=c:\temp\SoapProcessor\
```

El parámetro del MAX_CONTENT_LENGTH es usado para limitar el tamaño aceptado del requisito del Web Service, y el parámetro TEMPORARY_FILES es usado para especificar donde el SoapProcessor debería colocar los archivos temporales.

Especificar la función XMLHandler para cada función del Web Service

Optar exporta desde la base de datos dentro de un documento XML así como cuando responde al ReturnAll del Web Service.

Una función XmlHandler es usada para implementar la función del Web Service. Optar implementa un XmlHandler para la función del Web Service ReturnAll.

Especificar tripletas para definir el XmlHandler de la función ReturnOffer:

Tabla 4.16: (WSYSOAP: Tripleta XMLHandler en la función ReturnOffer)

Source Object	Verb	Target Object
Optar.Services.XMLHandlers.ReturnOffer	is a FNC	Optar.Abstract.XMLHandler DomServerExternal
	impl name	CalculateOffer
	file name	CalcOff
	implement SYS	Yes

Abrir el diagrama de acción para el ReturnOffer y especificar el siguiente código en el Edit Point XmlHandler functionality

Cuadro 4.4: (WsySoap: Diagrama de acción para ReturnOffer)

Edit Point XmlHandler functionality

Call ExportToReturnOffer
// Map OutputDocument<BodyOutObjectStoreReference>
// Map OutputDocument<BodyOutDocument>
// Map InputDocument<BodyInObjectStoreReference>
// Map InputDocument<BodyInDocument>

Especificar la función XmlHandler en el SoapProcessor

Tabla 4.17: (WsySoap: Tripleta XMLHandler en el SoapProcessor)

Source Object	Verb	Target Object
Optar.Services.SoapProcessor	comprises FNC	Optar.Services.XMLHandlers.ReturnOffer

4.4.2.2.3 Implementación de la funcionalidad del Broker (Subscriptor)

El Broker importa desde los datos del XML retornado dentro de la base e datos cuando se llama al Web Service ReturnAll.

Especificar la entidad SOAP del Web Service

La entidad base para el Subscriptor es la entidad SOAP

Especificar la herencia del Broker:

Tabla 4.18: (WSYSOAP: Tripleta del Broker)

Source Object	Verb	Target Object
Broker	is a ENT	HttpSoap
Broker.Services.SoapGenerator	impl name NME	SGBroker
	file name NME	SGBroker
Broker.Services.SoapProcessor	implement SYS	No

El Broker actúa solo como Subscriptor y no se usa la función del SoapProcessor.

Especificar la función que llamará el Web Service para importar los ítems específicos de la base de datos Optar.

El Broker peticiona a Optar a que retorne un listado específico con el parámetro basado dentro del documento XML. El Web Service necesita de un archivo XML como entrada.

Especificar la función WSReturnOfferExchange:

Tabla 4.19: (WSYSOAP: Tripleta de la función WSReturnOfferExchange)

Source Object	Verb	Target Object	Comments
WSReturnOfferExchange	is a FNC	ExportXMLDocument ImportXMLDocument	
	input FLD ...for VAR	WSYHTTP/InternetServerName Connection	
	input FLD ...for VAR	WSYHTTP/InternetServerPort Connection	
	input FLD ...for VAR	WSYHTTP/InternetApplicationName Connection	
	input FLD ...for VAR	WSYHTTP/InternetUrl Connection	
	input FLD ...for VAR	SOAPAction Connection	
	file name NME	WSOffer	
	impl name NME	WSOffer	
	name NME	WSOffer	

Abrir el diagrama de acción para el WsReturnOfferExchange y añadir el siguiente código en la especificación del edit points:

Cuadro 4.5: (WsySoap: Diagrama de acción para WsReturnOfferExchange)

Edit Point Process XML document

```
Call Broker.Services.SoapGenerator
  // Map Local/InputDocument<ObjectDocument>
  // Map Local/InputDocument<ObjectStoreReference>
  // Map Local/OutputDocument<ObjectDocument>
  // Map Local/OutputDocument<ObjectStoreReference>
  // Map Connection<SOAPAction>
  // Map Connection<InternetServerName>
  // Map Connection<InternetServerPort>
  // Map Connection<InternetApplicationName>
  // Map Connection<InternetUrl>
  // Map <CharSet.utf-8>
  // Map <UseSSL.*No>
If NOT Environment<*Returned status> IS <State: OBJECTS/*Returned
status.*Abnormal>
  Call ImportReturnOfferFromXML
    // Map Local/OutputDocument<ObjectStoreReference>
    // Map Local/OutputDocument<ObjectDocument>
  Go Sub Check error
else
  Go Sub Check error
```

Pre Point Handle error message

```
Dialog Message Message: OBJECTS/Message
  // Map Environment<*Message text>
```

4.4.2.2.4 Prueba de la Funcionalidad del Web Service

Para probar las funciones, es necesario configurar el Servidor Web, y generar y construir las funciones usadas.

Crear las funciones que invocan

Primero es necesario crear la función prueba que invoque al WSItemListExchange y WSReturnOfferExchange respectivamente.

Especificar la función TestWSReturnOffer:

Tabla 4.20: (WSYSOAP: Tripleta de la función WSReturnOfferExchange)

Source Object	Verb	Target Object	Comments
TestWSReturnOffer	is a FNC	FunctionShell ClientExternal	
	message MSG	InternetUrl	
		SOAPAction	
		InputDocument	
	local FLD	SOAPAction WSYHTTP/InternetUrl BodyFile	

- Especificar el contenido del mensaje InternetUrl para el SoapDisA.exe
- Especificar el contenido del mensaje SOAPAction para el CalculateOffer
- Especificar el contenido del mensaje InputDocument para c:\temp\Request.xml
- Insertar en el diagrama de acción el siguiente código dentro de la función TestWSReturnOffer:

Cuadro 4.6: (WsySoap: Diagrama de acción para TestWsReturnOffer)

Edit Point Execute

```
Format Message Message: TestWSReturnOffer.InternetUrl, Local<InternetUrl>
Format Message Message: TestWSReturnOffer.SoapAction,
Local<SOAPAction>
Format Message Message: TestWSReturnOffer.InputDocument,
Local<BodyFile>
Call WSReturnOfferExchange
// Map <FileName.*Blank>
  // Map <version.*Blank>
  // Map <Encoding.*Blank>
  // Map Local<BodyFile>
  // Map <InternetServerName.localhost>
  // Map <InternetServerPort.HTTP>
```

```
  // Map <InternetApplicationName.*Blank>
  // Map Local<InternetUrl>
  // Map Local<SOAPAction>
Go Sub Terminate
```

Generar y Construir

Generar y construir cada función implementada en esta aplicación:

- WSReturnOfferExchange
- TestWSReturnOffer
- Broker
- Optar

***Crear el ejecutable para la función de invocación* TestWSReturnOffer**

Añadir la siguiente información dentro de los archivos ini para las funciones

```
[TransacXML]
MAX_CONTENT_LENGTH=10240
TEMPORARY_FILES=c:\temp\SoapGenerator\
```

Especificar de la Configuración del Servidor Web Apache

Para nuestro caso se utilizó el Servidor Web Apache por la interacción de un Alias Script que nos permite invocar al SoapDisA.exe como tipo CGI.

Dentro del archivo de configuración del Apache colocamos lo siguiente:

Cuadro 4.7: (WsySoap: Configuración servidor apache)

```
ScriptAlias /webtutor/ "D:/WSTESIS/GEN/WINC/Release/"
<Directory "D:/WSTESIS/GEN/WINC/Release">
 AllowOverride None
   Options None
    Order allow,deny
    Allow from all
</Directory>
ScriptAlias /cgi-bin/ "D:/WSTESIS/GEN/WINC/Release/"
```

Como podemos apreciar en el código anterior creamos un alias llamado webtutor que direccionamos hacia la localidad de los archivos ejecutables de nuestro proyecto que en este caso es: D:/WSTESIS/GEN/WINC/Release/"

4.4.2.2.5 Ejecución Manual del Web Service

Como el Web Service necesita un XML de entrada con el parámetro del número de la tarjeta de crédito, colocamos:

Cuadro 4.8: (WsySoap: Documento Request.xml)

```
<?xml version="1.0" encoding="utf-8" ?>
- <WSDiners:ItemList xmlns:WSDiners="http://localhost/webtutor/WSDiners">
  - <WSDiners:Item>
      <WSDiners:Tarjeta>36504823210047</WSDiners:Tarjeta>
    </WSDiners:Item>
</WSDiners:ItemList>
```

Éste archivo XML llamado Request.xml se conforma de dos parámetros que ingresarán al Web Service al ser peticionado, que es el número de la tarjeta de crédito.

Como resultado al correr el TestWSReturnOffer la base de datos Data.Broker.Returned Items del Broker ahora contendrán los ítems pedidos especificados dentro del documento del XML, pero con más detalles, como son: número de operación, tipo de comercio, fecha de transacción, monto, saldo.

El archivo XML de respuesta Response.xml contendrá algo como:

Cuadro 4.9: (WsySoap: Documento Response.xml)

```
<?xml version="1.0" encoding="utf-8" ?>
- <WSDiners:ReturnOffer
      xmlns:WSDiners="http://localhost/webtutor/WSDiners">
  - <WSDiners:Item NumOper="1">
      <WSDiners:Tarjeta>36504823210047</WSDiners:Tarjeta>
      <WSDiners:FechaOp>2004-10-12</WSDiners:FechaOp>
      <WSDiners:Cliente>Christian B</WSDiners:Cliente>
      <WSDiners:Comercio>Gasolinera Shell</WSDiners:Comercio>
      <WSDiners:Monto>12.33</WSDiners:Monto>
      <WSDiners:Corriente>12.33</WSDiners:Corriente>
      <WSDiners:Diferido>0</WSDiners:Diferido>
    </WSDiners:Item>
  - <WSDiners:Item NumOper="2">
      <WSDiners:Tarjeta>36504823210047</WSDiners:Tarjeta>
      <WSDiners:FechaOp>2004-10-12</WSDiners:FechaOp>
      <WSDiners:Cliente> Christian B </WSDiners:Cliente>
      <WSDiners:Comercio>De Prati</WSDiners:Comercio>
      <WSDiners:Monto>58.00</WSDiners:Monto>
      <WSDiners:Corriente>18.00</WSDiners:Corriente>
      <WSDiners:Diferido>40.00</WSDiners:Diferido>
    </WSDiners:Item>
  </WSDiners:ReturnOffer>
```

4.4.2.2.6 Aplicación de los Namespace

El namespace XML es un mecanismo para evitar que los nombres choquen en documentos XML que usan elementos y atributos definidos por fuentes diferentes. Un namespace es así una colección de elementos y atributos, el namespace es identificado por una referencia URL, y cada elemento o atributo en el namespace es identificado por un namespace y el nombre del elemento/atributo.

En los documentos XML un prefijo es usado en combinación con el nombre local y el prefijo es entonces mapeado para un namespace. Así el prefijo es ahora más que un alias para el namespace. Ejemplo:

<p:Order xmlns:p="http://www.websydian.com/namepsaces/order">...</p:Order>

En el elemento Order pertenece al namespace http://www.websydian.com/namespaces/order entonces en ese momento será reflejado en el documento XML. El nombre del prefijo puede ser cualquier otro nombre y no es usado para identificar al elemento. Son únicos el nombre local Order y el nombre del namespace.

Cuando el transacXML genera un documento XML basado en el modelo anterior, la salida será como:

Cuadro 4.10: (WsySoap: Antes del namespace en Request.xml)

```
<WSDiners:Item NumOper="1">
    <WSDiners:Tarjeta>36504823210047</WSDiners:Tarjeta>
    <WSDiners:FechaOp>2004-10-12</WSDiners:FechaOp>
</WSDiners:Item>
```

Para definir los namespace definimos lo siguiente:

Tabla 4.21: (WsySoap: Definición de los Namespace)

ItemList	is a ENT	NamespaceAware

Y entonces la salida se obtiene algo como:

Cuadro 4.11: (WsySoap: Después del Namespace en Response.xml)

```
- <WSDiners:ItemList
      xmlns:WSDiners="http://localhost/webtutor/WSDiners">
- <WSDiners:Item Numoper="1">
  <WSDiners:Tarjeta>36504823210047</WSDiners:Tarjeta>
  <WSDiners:FechaOper>09/09/2004</WSDiners:FechaOper>
  </WSDiners:Item>
  </WSDiners:ItemList>
```

4.4.3 Invocación del Web Service Piloto

Para la invocación del Web Service que fue realizado en Advantage Plex y Websydian, se ha realizado el cliente o el programa que hará uso del Web Service en ASP .Net de Microsoft. Con esto se probará que para invocar un Web Service no importa en que tipo de lenguaje esté desarrollado.

4.4.3.1 Creación del WSDL

Para la invocación del Web Service se necesita un documento WSDL (Web Service Description Language), el cual es un lenguaje para componer descripciones formales de los Web Services.

En el documento WSDL se incluye información que hace posible a un número de herramientas de programación generar la estructura y llamadas requeridas para comunicarse con el Web Service, éste incluye:

- La URL y otra información usada para contactarse con el Web Service.
- Información declarada que el protocolo SOAP que debería usar cuando se contacte con el Web Service.
- Definiciones para las operaciones que el Web Service ofrece (SOAPActions)
- Representaciones formales de la estructura del documento usado como entrada y salida para las operaciones ofrecidas.

Para la generación del wsdl haremos uso del Plex, para esto creamos el CreateWSDL.

4.4.3.1.1 Documentos de Entrada para XmlHandler

El CreateWSDL permite la creación de archivos WSDL basado en la selección de múltiples entradas de documentos para el XMLHandler.

Se recomienda que todos los documentos con el cuál son usados como entrada y/o salida para un XmlHandler, sean definidos como pertenecientes a un namespace, aún cuando la función CreateWSDL sea capaz de la creación valida de archivos WSDL para documentos, de la cual no le pertenece a ningún namespace.

4.4.3.1.2 Implementación del CreateWSDL

En este punto crearemos el wsdl para el Web Service piloto de Diners.

Heredar desde XmlElementWithServiceFunctions

Todas las entidades XMLElement, de los cuales son parte de los documentos usados por el Web Service, deben heredar desde XmlElementWithServiceFunctions

Tabla 4.22: (CreateWSDL: Tripleta para heredar de XmlElementWithServiceFunctions)

CommonXml.ItemList	is a ENT	XmlElementWithServiceFunctions
CommonXml.ItemList.Item	is a ENT	XmlElementWithServiceFunctions
CommonXML.ReturnOffer	is a ENT	XmlElementWithServiceFunctions
CommonXML.ReturnOffer.Item	is a ENT	XmlElementWithServiceFunctions

De éste añada una función ***CreateSchema*** para cada entidad XMLElement.

Permitir a los XMLHandlers Contener Documentos de Entrada y Salida

La generación del WSDL necesita información referente a cuáles documentos las funciones XmlHandler usa como entrada y/o salida.

La conexión XmlHandler y los documentos de entrada y/o salida usa permisos basados en el ***CreateSchema*** (use la tripleta FNC comprises FNC)**.**

En este caso el XmlHandler ReturnOffer recibe un documento ItemList como entrada y retorna un documento ReturnOffer como salida. El resultado de la tripleta es la siguiente:

Tabla 4.23: (CreateWSDL: Tripleta para el definir documentos de E/S del ReturnOffer)

Optar.Services.XMLHandlers.ReturnOffer	comprises	CommonXML.ItemList.CreateSchema
Optar.Services.XMLHandlers.ReturnOffer	comprises	CommonXML.ReturnOffer.CreateSchema

Si el documento es de entrada o salida, es especificado durante la ejecución de la función CreateWSDL.

Generar y Construir las Funciones de las Librerías

Generar y construir las funciones contenidas en las librerías contenidas en este tema:

- WSYDOM/DomObjectsToGenerateAndBuild
- SDSTRING/SDStringObjsToGenerateAndBuild
- WSYXML/SchemaObjectsToGenerateAndBuild
- WSYSOAP/WSDLObjectsToGenerateAndBuild

Generar y Construir las Funciones CreateSchema y GetName

Estas funciones abarcan las entidades del XMLElement, las cuales son llamadas durante la ejecución de la función del CreateWSDL.

Las siguientes funciones deben ser generadas y construidas:

- CommonXML.ItemList.CreateSchema
- CommonXML.ItemList.Item.CreateSchema
- CommonXML.ReturnOffer.CreateSchema
- CommonXML.ReturnOffer.Item.CreateSchema
- CommonXML.ItemList.GetName
- CommonXML.ItemList.Item.GetName
- CommonXML.ReturnOffer.GetName
- CommonXML.ReturnOffer.Item.GetName

Generar y Construir la Función CreateWSDL

La función CreateWSDL y la función que abarca al HandleXMLHandlers, son las funciones principales para la creación WSDL.

Las funciones son:

- DistributorA.Services.SoapProcessor.CreateWSDL
- DistributorA.Services.SoapProcessor.CreateWSDL.HandleXMLHandlers

Ejecutar la Función CreateWSDL

Al ejecutar la función CreateWSDL, se mostrará la siguiente pantalla como:

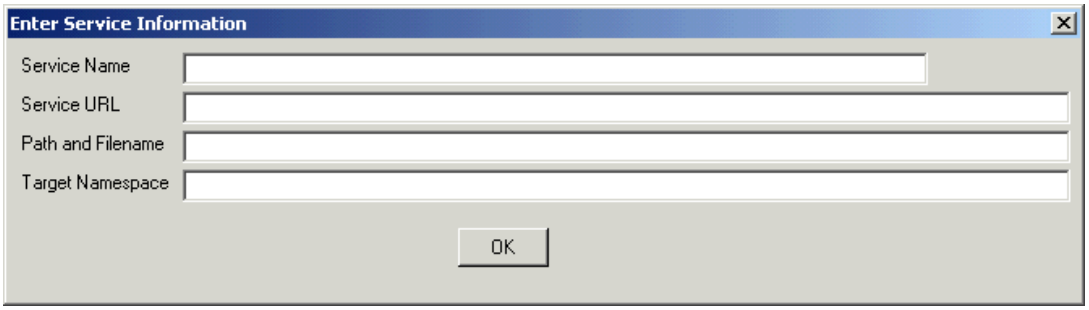


Fig. 4.23: (CreateWSDL: Ejecución)

- **Nombre del Servicio:** Es el nombre público del Web Service. De acuerdo a la especificación el nombre debe ser un NCNAME. Esto significa que no todos los caracteres son permitidos en el nombre del Web Service, como por ejemplo espacios y caracteres especiales.
- **Servicio URL:** La URL usado para contactar el Web Service.
- **Path y Nombre del Archivo:** Ingrese el path completo para el archivo donde el documento WSDL debería ser salvado. El directorio debe existir, mientras que el archivo se crea o se sobrescribe si éste existiera. Es recomendado usar el la extensión .wsdl para el documento, el cual representa un archivo WSDL para un número de aplicaciones.
- **Destino del Namespace:** Es el destino Namespace para el documento WSDL. Por defecto es el Servicio URL con el cual se hará bien en la mayoría de los casos, pero se puede reeditar si se lo desea.

El panel con la configuración del Web Service sería algo como:

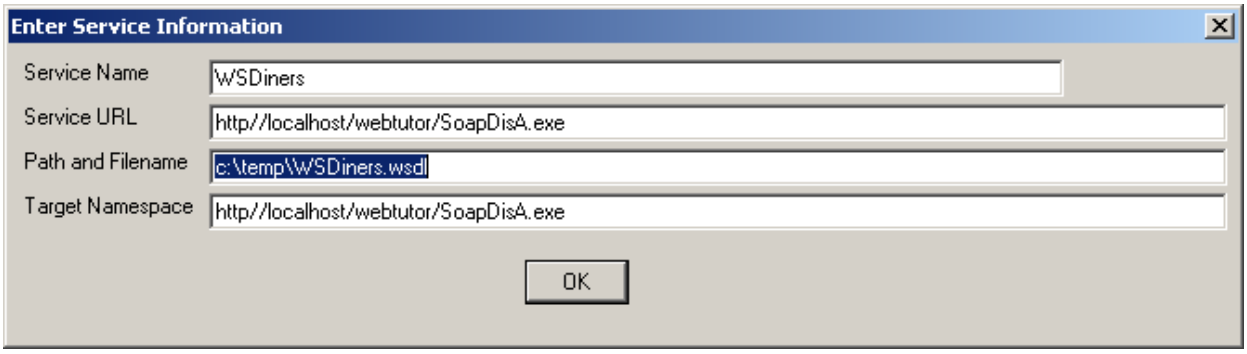


Fig. 4.24: (CreateWSDL: Configuración)

La información ingresada en el panel anterior es la información común para todas las funciones XmlHandler (SOAPActions). Después presione OK, el manejo de las funciones XmlHandler se iniciará. Para cada XmlHandler el siguiente panel se mostrará:

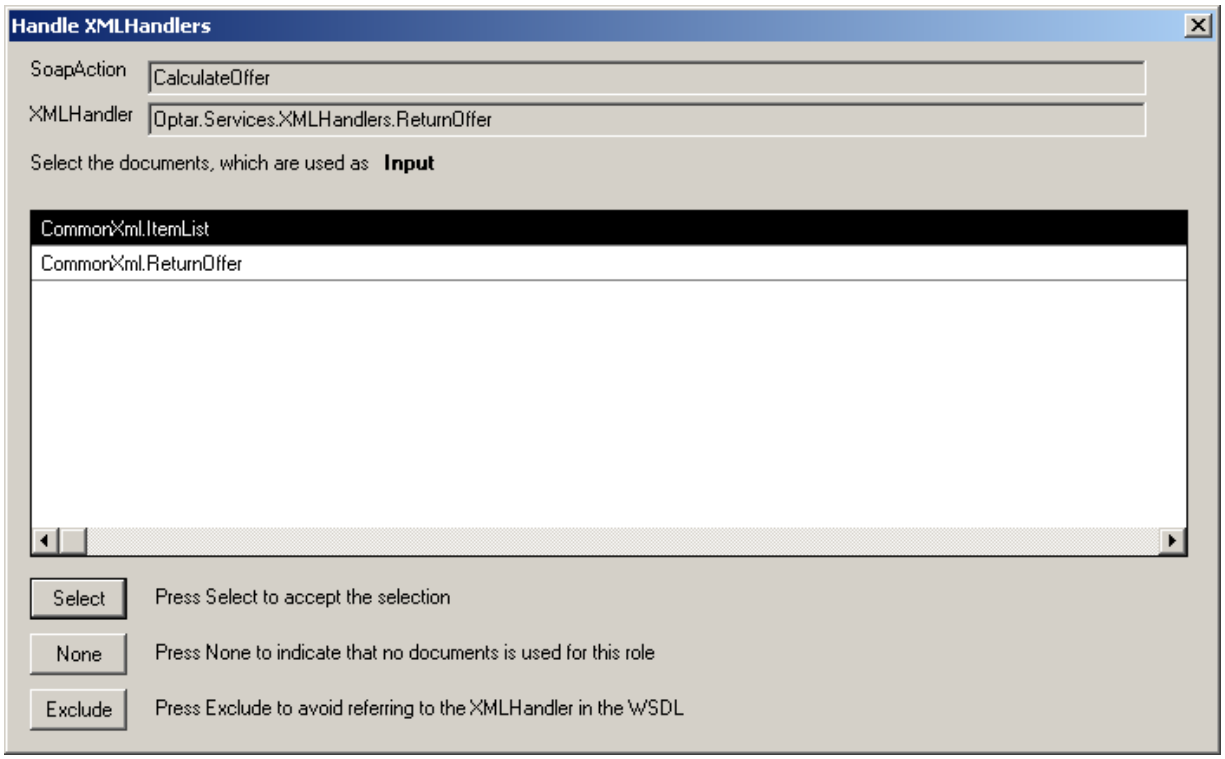


Fig. 4.25: (CreateWSDL: Asignación de XMLHandler de entrada)

Arriba del panel se muestra el SOAPAction y el nombre que abarca al XmlHandler. En el grid se muestra todos los documentos, los cuales son contenidos por el XmlHandler.

Este panel se muestra 2 veces por cada XmlHandler, uno donde el documento de entrada es seleccionada, y otro donde el documento de salida es seleccionada. El rol (ingreso o salida) a ser seleccionado se muestra en la parte de arriba del grid.

Entonces el segundo panel (documento de salida) quedaría como:

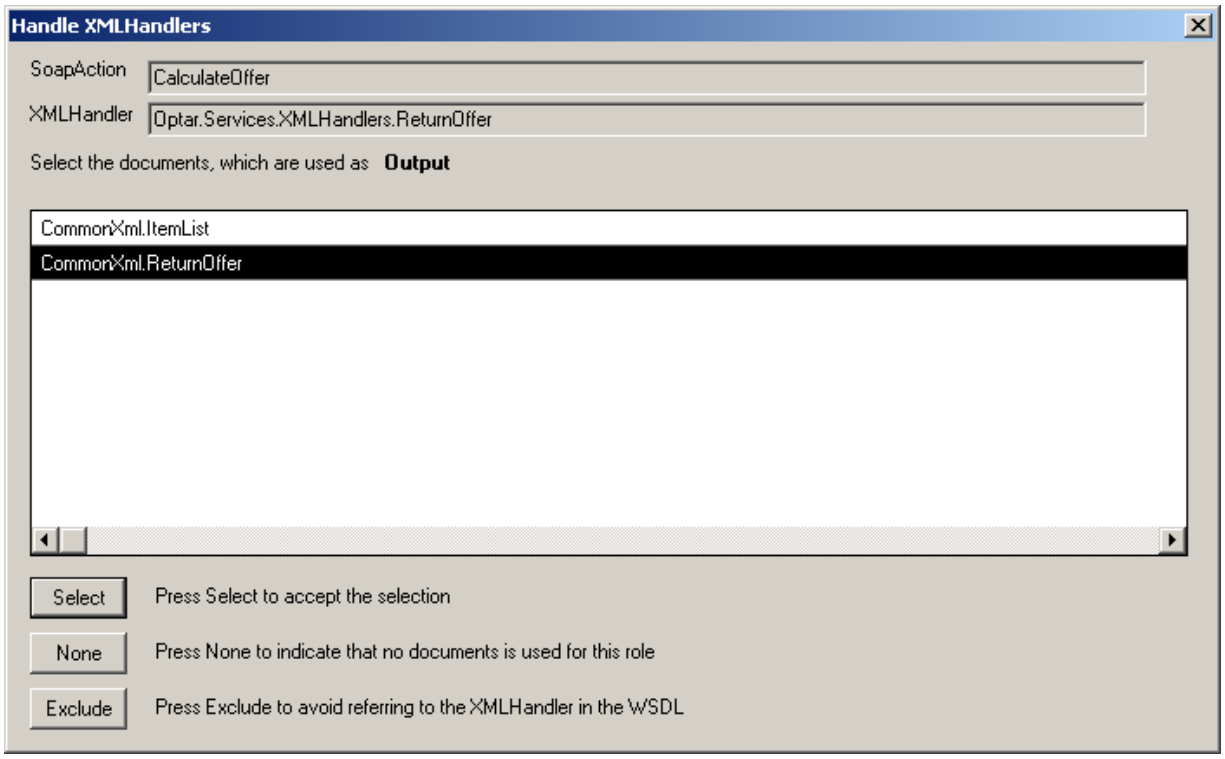


Fig. 4.26: (CreateWSDL: Asignación de XMLHandler de salida)

El documento CommonXML.ItemList debería seleccionarse como documento de entrada y el CommonXML.ReturnOffer como documento de salida.

Después, la función terminará y el archivo WSDL será creado y salvado en el directorio especificado.

El archivo WSDiners.wsdl contiene la descripción de la interfase del Web Service. El archivo WSDiners1.xsd también se creará conjuntamente con el WSDL, el cual contiene las

definiciones de esquema para los documentos de entrada y salida, dichos archivos quedarían de la siguiente forma:

Cuadro 4.12: (WSDL: Documento WSDiners.wsdl)

```
<?xml version="1.0" encoding="utf-8" ?>
- <wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns=""
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://192.100.7.14/Webtutor/SoapDisA.exe"
    targetNamespace="http://192.100.7.14/Webtutor/SoapDisA.exe"
    xmlns:STNS="http://192.100.7.14/Webtutor/SoapDisA.exeSTNS"
    xmlns:p1="http://localhost/webtutor/WSDiners">
- <wsdl:types>
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://192.100.7.14/Webtutor/SoapDisA.exeSTNS"
    elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://localhost/webtutor/WSDiners"
    schemaLocation="WsDiners1.xsd" />
    </xs:schema>
    </wsdl:types>
- <wsdl:message name="CalculateOfferInput">
  <wsdl:part name="body" element="p1:ItemList" />
    </wsdl:message>
- <wsdl:message name="CalculateOfferOutput">
  <wsdl:part name="body" element="p1:ReturnOffer" />
    </wsdl:message>
- <wsdl:portType name="WsDinersPortType">
- <wsdl:operation name="CalculateOffer">
  <wsdl:input message="tns:CalculateOfferInput" />
  <wsdl:output message="tns:CalculateOfferOutput" />
    </wsdl:operation>
    </wsdl:portType>
- <wsdl:binding name="WsDinersSoapBinding" type="tns:WsDinersPortType">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="CalculateOffer">
  <soap:operation soapAction="CalculateOffer" />
- <wsdl:input>
  <soap:body use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
    </wsdl:output>
    </wsdl:operation>
    </wsdl:binding>
```

```
- <wsdl:service name="WsDiners">
- <wsdl:port name="WsDinersPort" binding="tns:WsDinersSoapBinding">
  <soap:address location="http://192.100.7.14/Webtutor/SoapDisA.exe" />
    </wsdl:port>
    </wsdl:service>
    </wsdl:definitions>
```

Y el archivo WSDiners.xsd queda como:

Cuadro 4.13: (WSDL: Documento WSDiners1.xsd)

```
   <?xml version="1.0" encoding="utf-8" ?>
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://localhost/webtutor/WSDiners"
      elementFormDefault="qualified" attributeFormDefault="unqualified">
- <xs:element name="ItemList">
- <xs:complexType>
- <xs:sequence>
- <xs:element name="Item" minOccurs="1" maxOccurs="unbounded">
- <xs:complexType>
- <xs:sequence>
  <xs:element name="Tarjeta" type="xs:string" minOccurs="1" maxOccurs="1" />
    </xs:sequence>
    </xs:complexType>
    </xs:element>
    </xs:sequence>
    </xs:complexType>
    </xs:element>
- <xs:element name="ReturnOffer">
- <xs:complexType>
- <xs:sequence>
- <xs:element name="Item" minOccurs="1" maxOccurs="unbounded">
- <xs:complexType>
- <xs:sequence>
  <xs:element name="Tarjeta" type="xs:string" minOccurs="1" maxOccurs="1" />
  <xs:element name="FechaOp" type="xs:string" minOccurs="1" maxOccurs="1" />
  <xs:element name="Cliente" type="xs:string" minOccurs="1" maxOccurs="1" />
  <xs:element name="Comercio" type="xs:string" minOccurs="1" maxOccurs="1" />
  <xs:element name="Monto" type="xs:string" minOccurs="1" maxOccurs="1" />
  <xs:element name="Corriente" type="xs:string" minOccurs="1" maxOccurs="1" />
  <xs:element name="Diferido" type="xs:string" minOccurs="1" maxOccurs="1" />
```

```
    </xs:sequence>
  <xs:attribute name="NumOper" type="xs:string" use="required" />
    </xs:complexType>
    </xs:element>
    </xs:sequence>
    </xs:complexType>
    </xs:element>
    </xs:schema>
```

Los documentos pertenece al mismo nombre del namespace, lo cual significa que solo un único esquema de archivo es creado. Estos dos archivos deben estar localizado en la misma carpeta del archivo WSDL.

4.4.3.2 Desarrollo del Cliente en ASP .Net

Luego de haber creado el archivo WSDL, donde se encuentra todas las características necesarias para poder invocarlo, procedemos en este caso a crear el cliente o consumidor del Web Services en ASP.Net

4.4.3.2.1 Procedimiento

Crear un proyecto en ASP .Net.

Creamos un proyecto de tipo Visual .Net, donde se encuentra todo lo relacionado a páginas ASP.

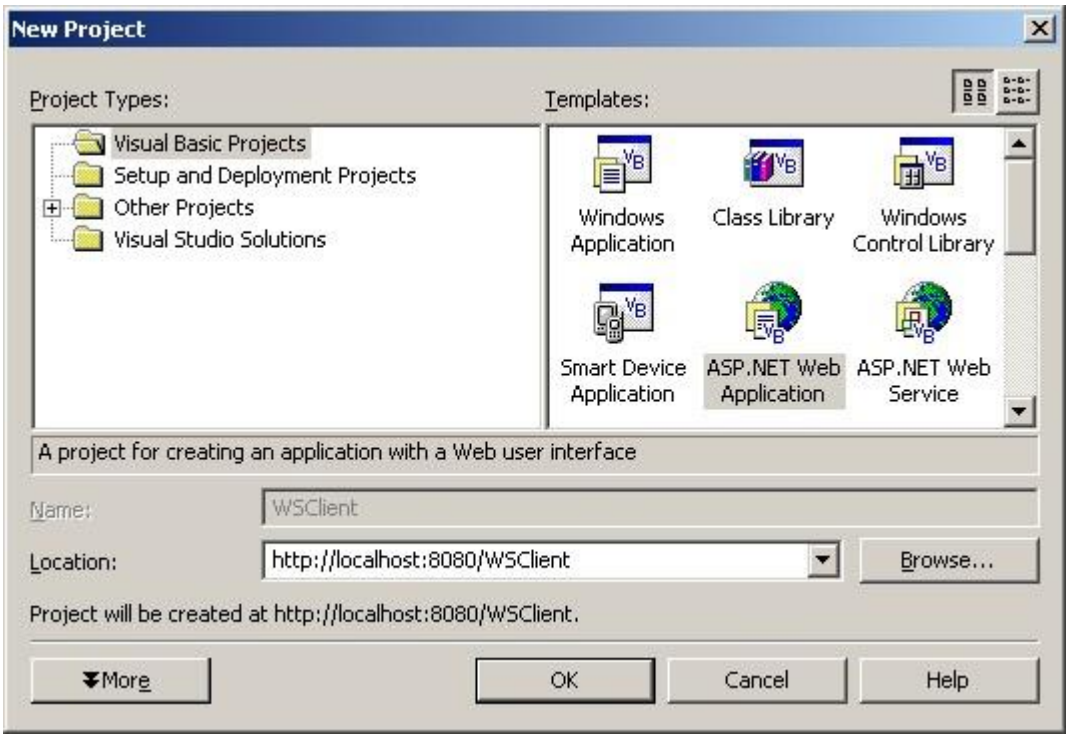


Fig. 4.27: (WSClient: Creación del cliente en ASP.Net)

Insertar un Web Reference

Añadimos un Web Reference, el cual se encarga de escoger el archivo WSDL desde una URL o dirección IP, e insertarlo en el proyecto.

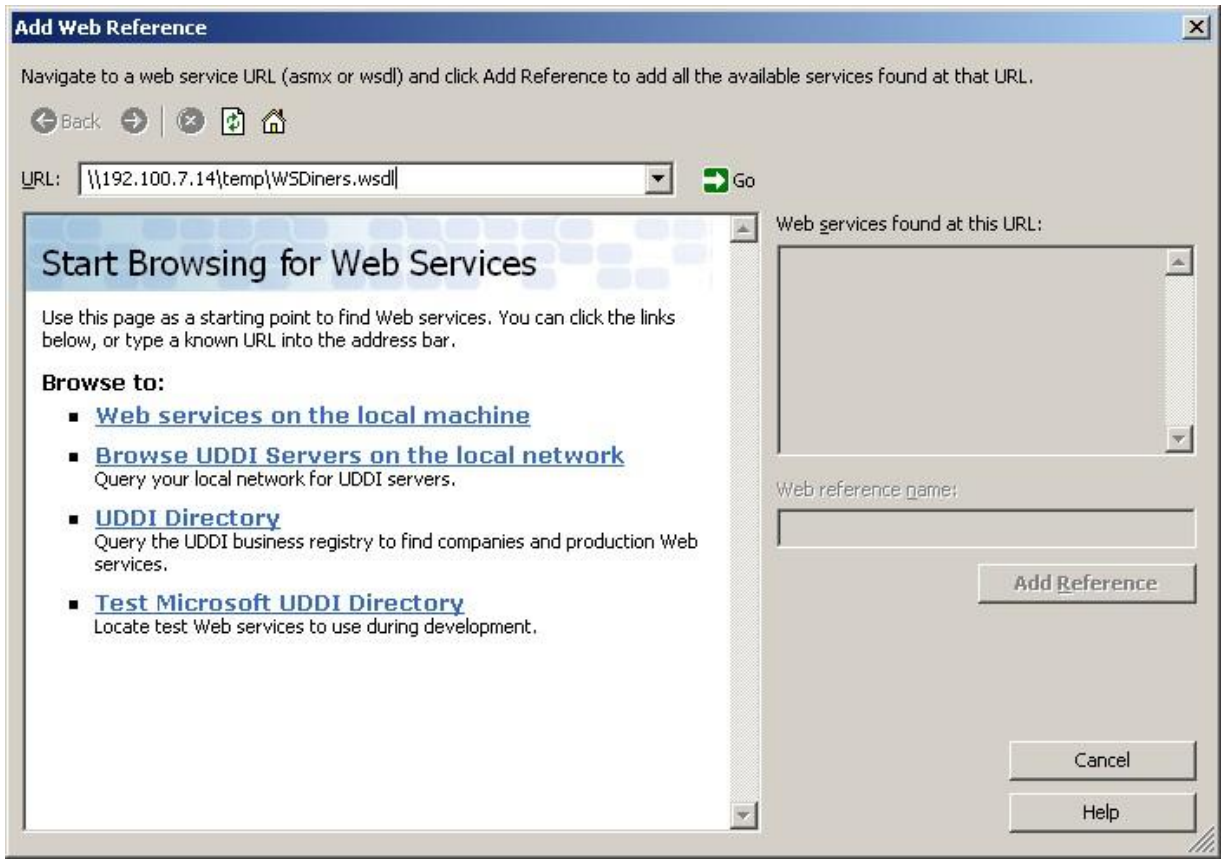


Fig. 4.28: (WSClient: Inserción del Web Reference)

Una vez añadido al proyecto, se crearán automáticamente las clases Proxy, con las respectivas funciones u operaciones del Web Service que se encuentra especificado en el documento WSDL.

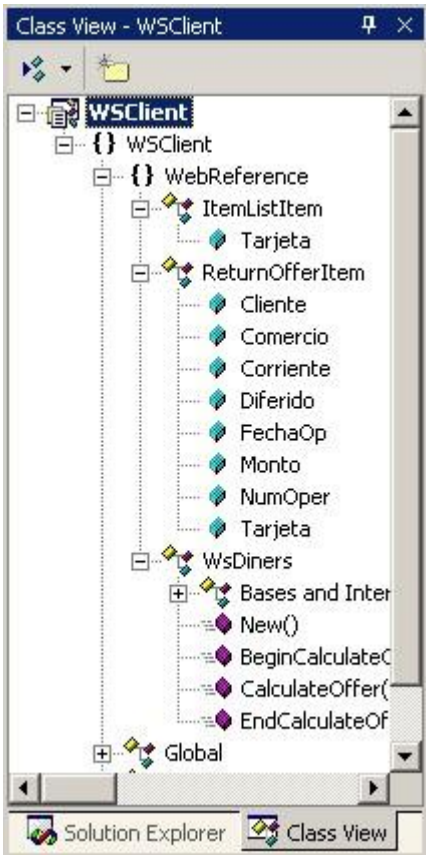


Fig. 4.29: (WSClient: Clases Proxy)

Añadir un Web Form

En este Web Form, a más de insertar el diseño de las páginas web con sus respectivas tablas, links, botones, etc, colocamos el código de Visual Basic .Net de la siguiente manera:

Cuadro 4.14: (WSClient: Código VB.Net para invocar al Web Service)

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load

    'Put user code to initialize the page here
    Dim i As Integer
    Dim nregistros As Integer
    Dim totalConsumo As Double
```

```
Dim totalDiferido As Double

Dim myWS As New WSClient.WebReference.WSDiners
Dim items(0) As WSClient.WebReference.ItemListItem
Dim item As New WSClient.WebReference.ItemListItem
Dim result As New WSClient.WebReference.ReturnOfferItem

item.Tarjeta = Request.QueryString.GetValues("tarjeta")(0)

items(0) = New WSClient.WebReference.ItemListItem
items(0) = item

Dim results() As
WSClient.WebReference.ReturnOfferItem=myWS.CalculateOffer(items)

nregistros = results.Length
If nregistros > 0 Then
   result.Tarjeta = results(0).Tarjeta
   result.Cliente = results(0).Cliente
   Label2.Text = result.Cliente
   Label4.Text = result.Tarjeta

   totalConsumo = 0
   totalDiferido = 0
   i = 0
   Do While i <= nregistros - 1
      result.Comercio = results(i).Comercio
      result.Corriente = results(i).Corriente
      result.Diferido = results(i).Diferido
      result.FechaOp = results(i).FechaOp
      result.Monto = results(i).Monto
      result.NumOper = results(i).NumOper

      totalConsumo = totalConsumo + result.Monto
      totalDiferido = totalDiferido + result.Diferido

      Label3.Text = Label3.Text & "<tr><td</td>" & _
            "<td>" & result.FechaOp & "</td>" & _
            "<td>" & result.NumOper & "</td>" & _
            "<td>" & result.Comercio & "</td>" & _
            "<td>" & FormatNumber(result.Monto, 2) & "</td>" & _
            "<td>" & FormatNumber(result.Diferido, 2) & "</td>" & _
            "</tr>"
      i = i + 1
   Loop
   Label6.Text = FormatNumber(totalConsumo, 2)
   Label9.Text = FormatNumber(totalConsumo, 2)
```

```
        Label7.Text = FormatNumber(totalDiferido, 2)
        Label8.Text = FormatNumber(totalDiferido, 2)
    Else
        Label5.Text = "<p><span align='center'>NO EXISTEN RESULTADOS</span></p>"
    End If
End Sub
```

- Como podemos apreciar en el código, en el evento Load del Web Form se encuentra las declaraciones de objetos y array del ItemListItem:

  Dim items(0) As WSClient.WebReference.ItemListItem
  Dim item As New WSClient.WebReference.ItemListItem

- Y para el resultado:

  Dim result As New WSClient.WebReference.ReturnOfferItem

- Obtenemos el número de la tarjeta que envían como parámetro:

  item.Tarjeta = Request.QueryString.GetValues("tarjeta")(0)

- Asignamos un objeto del ItemListItem al array cero de ítems y luego asignamos a item:

  items(0) = New WSClient.WebReference.ItemListItem
  items(0) = item

- Seguidamente asignamos a results la invocación del Web Service con la operación CalculateOffer:

  Dim results() As WSClient.WebReference.ReturnOfferItem = myWS.CalculateOffer(items)

- Y obtenemos del array results() de la siguiente manera:

  result.Tarjeta = results(0).Tarjeta

```
result.Cliente = results(0).Cliente


result.Comercio = results(i).Comercio
result.Corriente = results(i).Corriente
result.Diferido = results(i).Diferido
result.FechaOp = results(i).FechaOp
result.Monto = results(i).Monto
result.NumOper = results(i).NumOper
```

En el siguiente diagrama podemos ver de forma gráfica el funcionamiento de la invocación al Web Services WsDiners que para nuestro ejemplo, reside en la dirección http://192.100.7.14 mediante el protocolo de comunicación SOAP, haciendo referencia a la operación CalculateOffer que es el que mediante el mensaje de Entrada con las Inputs requeridas procesa y recupera la información accesando a la base de datos y la devuelve como mensaje de salida con las Outputs definidas.

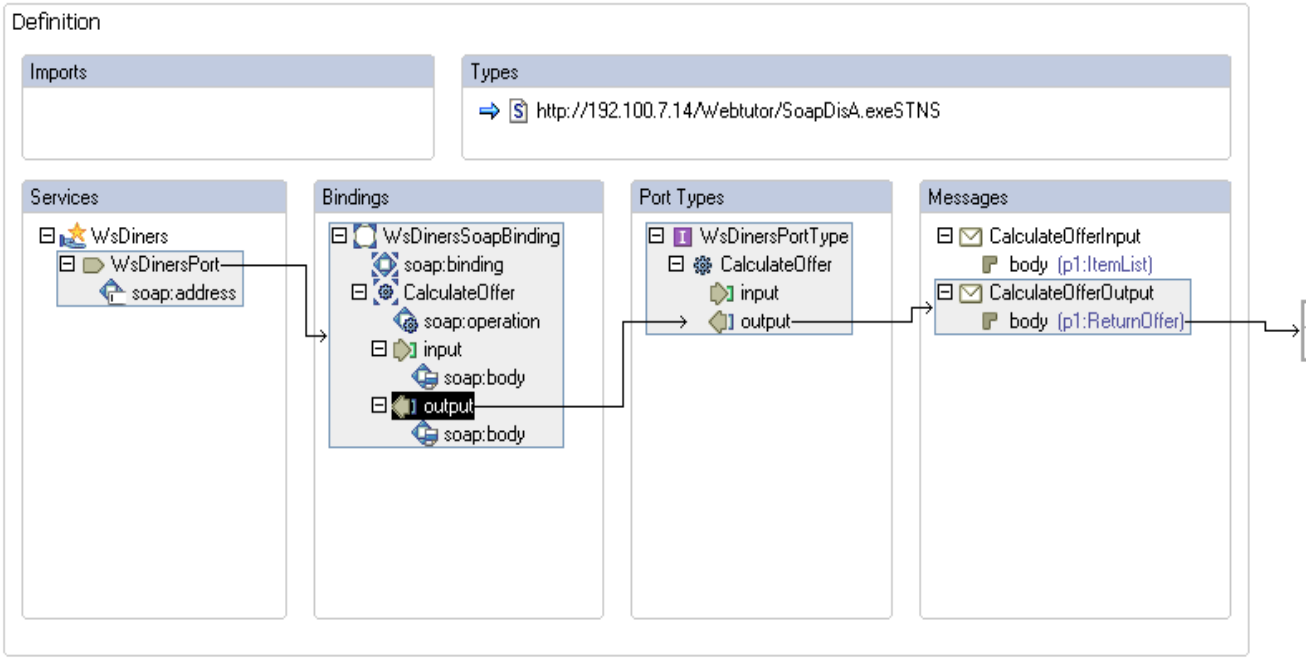


Fig. 4.30: (Diagrama Lógico del funcionamiento del Web Service)

4.4.3.2.2 Pruebas del Cliente Web

El diseño de la página web Cliente, para nuestro tema se lo desarrolló en DreamWeaver 2004 para luego implementarlo en el proyecto del ASP.Net anteriormente visto.

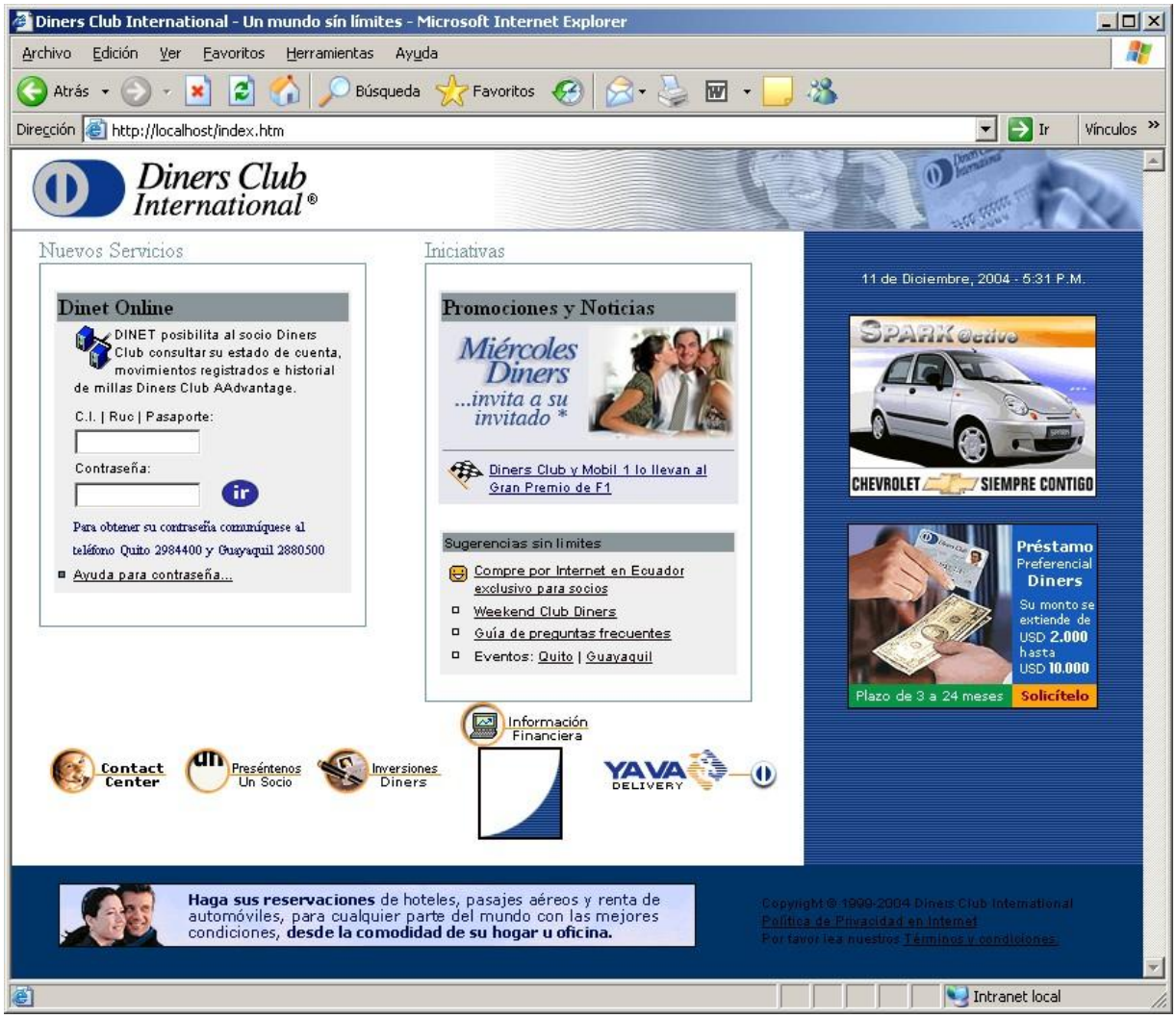


Fig. 4.31: (Cliente Web: Página inicial)

Para acceder a la autentificación del usuario, se ingresan la cédula de identidad o Ruc de la persona natural o jurídica como se indica en el gráfico siguiente.

Fig. 4.32: (Cliente Web: Ingreso con usuario y clave)

Una vez accedido a la página web del estado de cuenta Diners, se pulsa el botón de ingreso para obtener el detalle del estado de cuenta al mes actual.

Fig. 4.33: (Cliente Web: Datos de la tarjeta de crédito Diners)

La llamada o invocación al Web Service se lo realiza a través del botón antes visto, el cual llama a la página resumen.aspx que tiene el código que se mostró en el Cuadro 4.14. Inmediatamente se lista el detalle del consumo del mes realizado por dicho individuo.

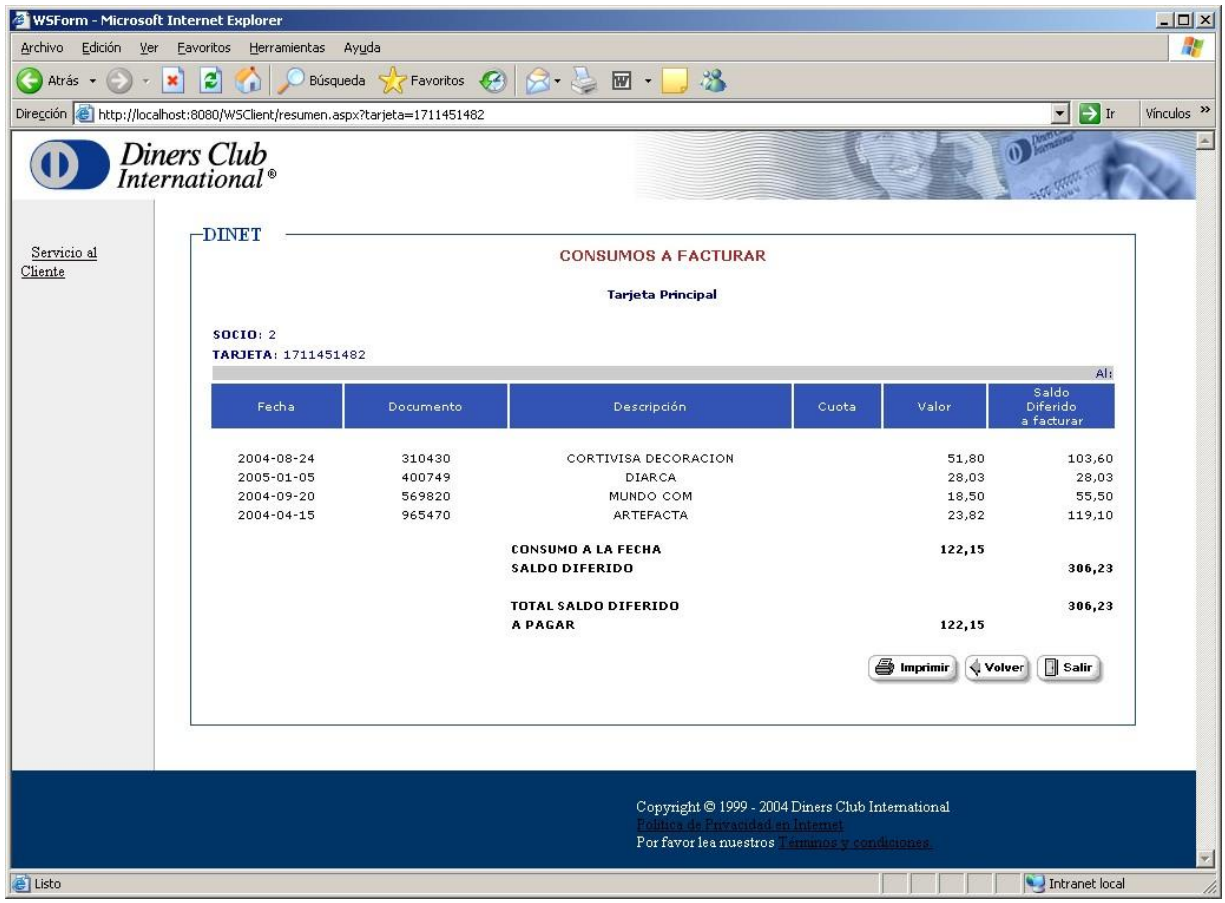


Fig. 4.34: (Cliente Web: Detalle del estado de cuenta)

CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- La reducción de costos y la optimización de recursos son dos de las principales prioridades en los presupuestos tecnológicos en las empresas como es Diners Club del Ecuador, por lo que el outsourcing, o contratación de servicios a proveedores externos, aparece para intentar lograr esos objetivos; esto hace que la tecnología se preocupe de que la interoperatibilidad entre la empresa y su outsorcing puedan mantener comunicación y acceso en el menor tiempo posible , de forma abierta sin dependencia ni complicaciones y de allí una de las razones de la existencia de los Web Services para poder integrarlos.

- La tecnología de Web Services ha evolucionado drásticamente en el transcurso del tiempo desde que apareció hasta hoy de tal forma que las casas de software IBM, Microsoft, CA, Sun Micorsoft System entre otras han puesto todo su potencial orientado a dar facilidades de desarrollo a los programadores y soluciones a las demandas de interoperatibidad empresarailes existentes hoy en día, esto ratifica que la tecnología de los Web Services vale la pena que sea estudiada, profundizada y aplicada como solución en línea empresarial.

- Los Web Services son componentes a nivel de internet que pueden ser usados por otros desarrolladores que necesiten las mismas características, siempre y cuando el servicio

se encuentre publicado en la Web, esto reduce los tiempos de construcción y además los costos de desarrollo ya que permiten la reutilización de procesos ya diseñados.

- Diners Club del Ecuador cuenta ahora con toda la puerta abierta para poder integrar sus procesos a la web sin preocuparse de la tecnología que maneja a nivel de back end, mediante la construcción de Web Services la interoperatibilidad con su front end o tercerizadora es transparente e independiente.

- Websydian con sus patrones o librerías es el complemento sencillo para producir Web Services sobre los procesos ya realizados previamente en la herramienta Advantge Plex, de tal forma que se aprovecha el desarrollo ya existente a nivel de procesos.

- Los desarrolladores de Advantage Plex no tienen que tener conocimeintos profundos de esta nueva tecnología de Web Services lo cual les facilita el aprendizaje y les motiva a incursionar en desarrollos o soluciones usando Web Services.

- El haber realizado esta tésis en las herramientas Advantage Plex y Websydian que Diners Club del Ecuador posee; avaliza que tanto como empresa y desarrolladores se puede dar paso a los nuevos requerimientos de negocios que Diners tiene sin inconvenientes de plataforma o lenguajes d eprogramación aún de forma interna.

5.2 Recomendaciones

- Para desarrollar un Web Service en cualquier herramienta de desarrollo, se debe tener conocimientos claros de cada uno de los elementos que conllevan al momento de construir el mismo.

- Antes de compilar y generar las funciones realizadas en Advantage Plex, es necesario tener bien configurado la herramienta para el lenguaje deseado y no obtener errores de compilación.

- El Web Service realizado en Advantage Plex con Websydian, trabaja de mejor manera en un Servidor Web Apache (modo consola) o en IIS (Internet Information Server) de Microsoft.

- La mejor manera de probar un archivo WSDL es hacerlo mediante un programa test de Web Service como son SoapScope, Web Service Developer de IBM, etc, los cuales realizan pruebas de estructura y diseño del WSDL, mas no el cliente o invocador del Web Service.

- Se recomienda configurar los .INI del archivo del Web Service (SoapDisA.exe) e incorporar la Job Description de la Base de Datos del AS400, con su respectivo nombre de base de datos, username y password.

- Para el caso de integrar un Web Service en un dominio de Internet, se recomienda por seguridad que el archivo WSDL resida en otro dominio, para estar aislado de ataques que perjudiquen tanto a la Base de Datos como al invocador.

- Las seguridades para los Web Services no se encuentran establecidas por la W3C, por lo que se recomienda colocar seguridades a nivel de capas u otro tipo de tecnología que protejan contra individuos que atenten con el normal comercio de las empresas.

- El incursionar en una herramienta nueva siempre implica un precio, especialmente si no existe soporte suficiente en la misma, el poner en marcha cualquier tipo de requerimiento toma más tiempo del previsto, y el ponerse en contacto con un soporte es mas complicado cuando la casa proveedora del software no se encuentra cerca, tiene diferente zona horaria, y además diferente lenguaje. Todos estos son puntos a notar al adquirir una herrameinta no tan conocida como Advantage Plex y Websydian.

- Aunque la herramienta Advantage Plex y Websydian son herraminetas CASE que pretenden minimizar el esfuerzo y conocimiento del programador si es necesario tener conocimientos básicos de XML, SOAP y WSDL antes de incursionar en la programación de los Web Services.

BIBLIOGRAFIA

- http://geneura.ugr.es/~maribel/xml/introduccion/index.shtml
- http://codorniz.arcos.inf.uc3m.es/msaa/Cursos/Sistemas%20distribuidos%20en%20dotnet.ppt.
- http://www.fisica.uson.mx/carlos/WebServices/WS_UDDI.htm
- http://www.fisica.uson.mx/carlos/WebServices/WS_WSDL.htm
- http://desarrolloweb.com/articulos/1545.php?manual=54
- http://www.w3.org/2002/ws/
- http://www.w3.org/TR/ws-arch/ (http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/)
- http://www-106.ibm.com/developerworks/webservices/newto/?Open&ca=daw-ws-news#1

BIOGRAFIA