CAPITULO IV

Introducción a los Servicios Web

CONTENIDO

4.1 Servicios Web

4.1.1 Definición de Servicio Web

Existen varias definiciones de los servicios XML Web Services en base a las empresas que los diseñan, así tenemos:

IBM [www012]. *Los servicios del Web son aplicaciones audescriptivas, autónomas y modulares que se pueden mezclar y combinar con otros servicios del Web para crear productos innovadores, procesos y cadenas de valor.*

Sun Microsystems [www013]. *Un servicio Web describe la funcionalidad específica del negocio expuesta por una compañía, generalmente a través de una conexión de Internet, con el fin de proporcionar una manera para que otra compañía, o programa informático utilice el servicio.*

Microsoft [www014]. *Un servicio del Web es una unidad de la lógica del negocio que proporciona datos y servicios para otras aplicaciones. Las aplicaciones acceden a los Web services vía protocolos Web como **HTTP y SOAP** y formatos de datos universales como **XML**, sin necesidad de preocuparse de cómo cada Web Service es implementado. Los servicios del Web combinan los mejores aspectos del desarrollo basado en componentes y el Web, y son la piedra angular del modelo de programación de Microsoft.Net.*

De lo expuesto anteriormente, un Web Service es un componente de software que se comunica con otras aplicaciones codificando los mensajes en XML y enviando estos mensajes a través de protocolos estándares de Internet tales como el Hypertext Transfer Protocol (**HTTP**).

Un Web Service es similar a un sitio web que no cuenta con un interfaz de usuario y que da servicio a las aplicaciones en lugar de a las personas y, en vez de obtener solicitudes desde el navegador y retornar páginas web como respuesta, recibe solicitudes a través de un mensaje formateado en XML desde una

aplicación, realiza una tarea y devuelve un mensaje de respuesta también en formato XML.

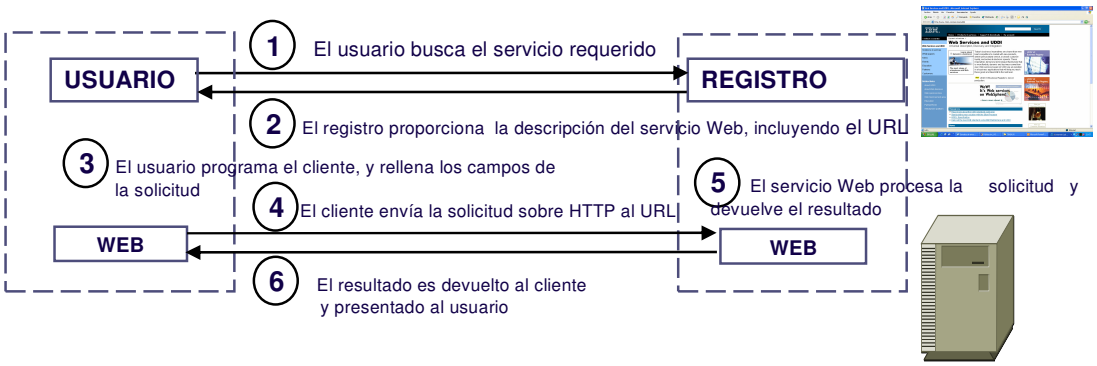

Figura 4.1 ***Descripción del Servicio Web***

4.2 Arquitectura de los Servicios Web XML

Un servicio Web o WebService es un servicio ofrecido por una aplicación que expone su lógica a clientes de cualquier plataforma mediante una interfaz accesible a través de la red utilizando tecnologías (protocolos) estándar de Internet.

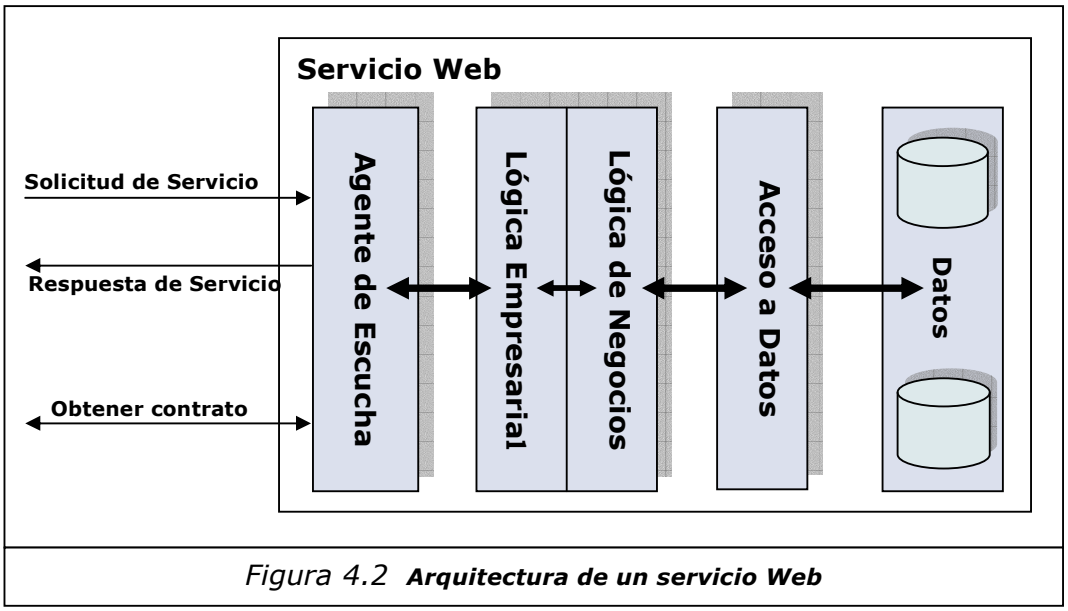

Figura 4.2 ***Arquitectura de un servicio Web***

La arquitectura se divide en cinco capas lógicas [WWW017].

- ✓ ***La capa de datos*** Almacena información requerida por el servicio Web.

- ✓ ***Capa de acceso a datos*** Presenta una vista lógica de los datos físicos a la capa de negocios, aísla la lógica de negocios de los cambios realizados a los almacenes de datos y garantiza la integridad de los datos.

- ✓ ***Capa de negocios*** Implementa la lógica de negocios del servicio Web.

- ✓ ***La Lógica Empresarial*** Proporciona una interfaz sencilla que se asigna a las operaciones expuestas por el servicio Web.

- ✓ ***El agente de escucha*** Recibe los mensajes entrantes que contienen solicitudes de servicios, analiza los mensajes y envía la solicitud al método apropiado en la capa de negocios. Si el servicio devuelve una respuesta, el agente de escucha empaqueta la respuesta de la capa de negocios en un mensaje y su envío al cliente.

Un **servicio Web** es un programa servidor que acepta peticiones entrantes de los clientes e intercambia XML sobre HTTP

Los WebService son un conjunto de métodos WebMethods asociados lógicamente y llamados a través de SOAP. Cada WebService tiene dos archivos asociados: uno con extensión "asmx" y otro con extensión "cs", si es C# o "vb" si es Visual Basic. Los Web Services se implementan en la clase derivada de "System.Web.Services.WebService".

Ejemplo de un Servicio Web

```
//inicialización Web Servece mediante ASP
<%@ WebService Language="C#"
Class="HolaMundoWebS" %>

//importación de los espacios de nombres
using System.Web.Services;

//Declaración  web service
[WebService(
Namespace="http://www.localhost/WebServices/",
   Description="Hola, Mundo con Web Service")]

//Definición de la clase que devuelve una cadena
public class HolaMundoWebS {
    [WebMethod(Description="Devuelve la cadena Hola,
Mundo")]
    public string Saludar() {
      return "Hola, Mundo";
  }
}
```

Figura 4.3 ***Ejemplo de un servicio Web***

El resultado del código anterior es el siguiente:

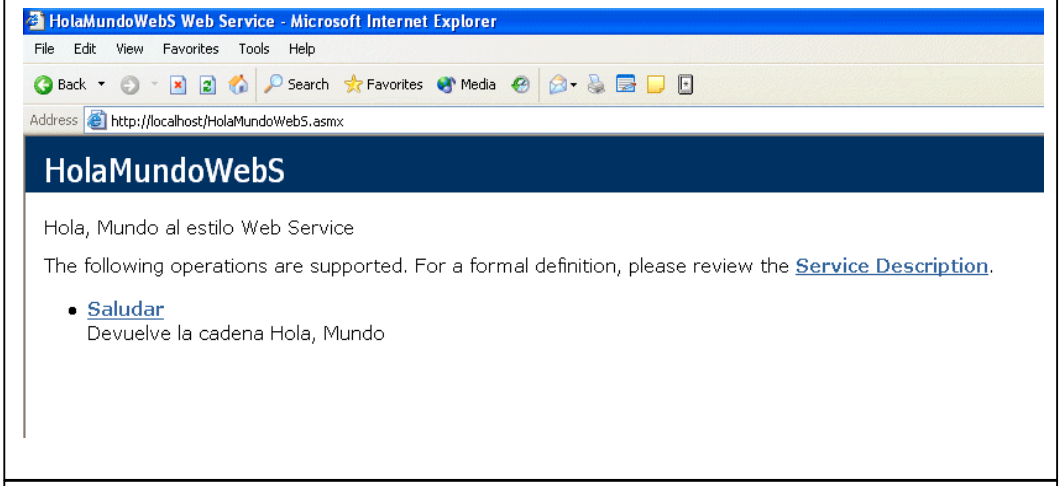

Figura 4.4 ***Resultado de un servicio Web***

Los servicios Web XML deben ser independientes en lo que respecta a la selección de sistema operativo, modelo de objetos y lenguaje de programación con el fin de

funcionar correctamente en la heterogeneidad del Web. Los servicios Web deben caracterizarse por lo siguiente: [LIB004]

- ***Comunicación ubicua:*** La conexión de cualquier sistema o dispositivo a Internet debe garantizar la disponibilidad para cualquier otro sistema o dispositivo conectado a Internet.

- ***Formato de datos universal:*** Cualquier sistema compatible con estándares abiertos como mensajes de texto autodescriptivos puede comprender y compartir los servicios Web XML y permitir la comunicación entre sistemas autónomos y heterogéneos.

- **Interoperabilidad:** Un servicio 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 desde Internet.

- **Interfaces fuertemente tipadas:** No debería haber ambigüedad acerca del tipo de dato enviado y recibido desde un servicio.

- **Aprovechar los estándares de Internet existentes:** En la implementación del servicio WEB y evitar reinventar soluciones a problemas que ya se han resuelto.

- **Soporte para cualquier lenguaje**: Un servicio Web es 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 ligada solo a una infraestructura de componentes en particular.

La llamada a un Servicio Web se la puede realizar mediante dos modelos:

Modelo sincrónico	**Modelo asincrónico**
Quien llama a la función debe esperar al retorno antes de continuar.	Quien llama a la función puede continuar sin esperar que la función a la que se llama retorne.

1.- Modelo sincrónico

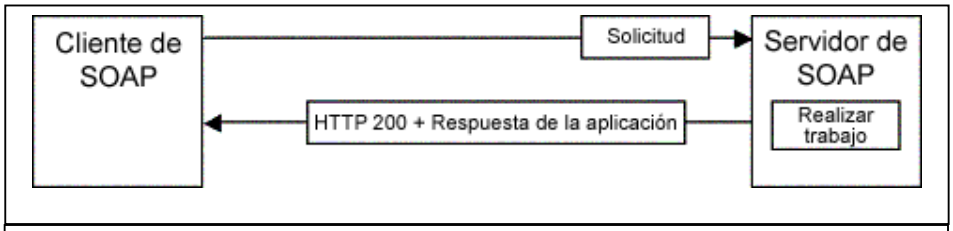

Figura 4.5 ***Descripción modelo síncrono***

La llamada a una función de forma síncrona implica que quien llama a la función debe esperar al retorno antes de continuar. Ésta es la forma más habitual de llamar a cualquier función, incluso a un método de Web.

2.- Modelo asincrónico

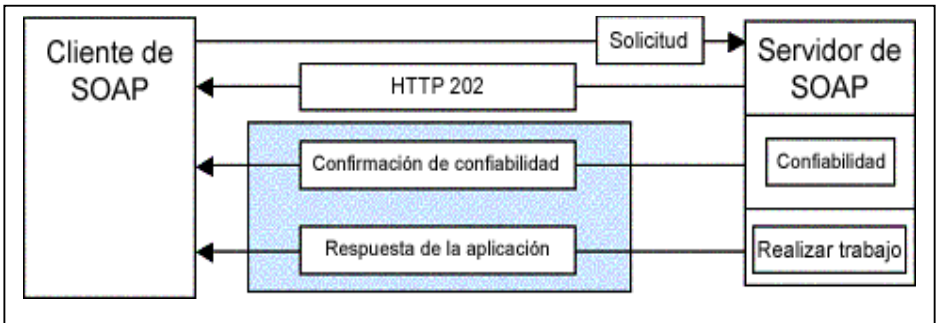

Figura 4.6 ***Descripción del modelo asíncrono***

En las llamadas asíncronas, se puede continuar sin esperar a que la función a la que se llama retorne, con lo que se mantiene más de una "línea de ejecución" del código. Esta forma no se utiliza con demasiada frecuencia, puesto que resulta más compleja y, en la mayoría de los casos, o la función retorna rápidamente o, incluso si se demora, no tenemos nada que hacer antes de su retorno. Aún así, en un servicio Web es frecuente que la llamada tarde un poco y que se pueda hacer otras cosas antes de que retorne el servicio, por ejemplo llamar a otro servicio de Web.

Por ejemplo un servicio de Web que efectúa una consulta de precios en diez sitios diferentes y que cada consulta dura dos segundos. Si hiciéramos todas las llamadas de forma síncrona (esperando cada llamada a la siguiente), el tiempo total será de veinte segundos. Si disparamos todas las llamadas de forma asíncrona ("al mismo tiempo"), el tiempo total será de pocos segundos.

4.2.1 Ventajas e inconvenientes de los Servicios Web en el desarrollo de aplicaciones distribuidas

Los servicios Web se construyen sobre el acoplamiento débil del modelo de programación Web tradicional, y se extienden para su uso en otro tipo de aplicaciones. Existen tres diferencias principales entre los servicios Web y las aplicaciones Web distribuidas [WWW15]:

a. Los servicios Web utilizan mensajes SOAP cuyo cuerpo contiene cualquier tipo de mensaje XML que una aplicación desee enviar y las aplicaciones tradicionales en cambio hacen uso de mensajes **MIME** [RFC 1344]

b. Los servicios Web no son dependientes del protocolo Http.

c. Los servicios Web proporcionan metadatos que describen los mensajes, producen y consumen.

- ***Ventajas de los Servicios Web respecto aplicaciones distribuidas***

Los servicios Web XML frente a otras aplicaciones distribuidas tienen las siguientes ventajas:

- Son la puerta a nuevas oportunidades empresariales, facilitan la comunicación entre aplicaciones asociadas.
- Permiten que las aplicaciones compartan información.
- Son unidades de código discretas, cada una de las cuales se encarga de un conjunto limitado de tareas.
- Están basados en XML, el lenguaje universal del intercambio de información de Internet y pueden utilizarse en cualquier plataforma o sistema operativo, independientemente del lenguaje de programación utilizado.
- Aumenta el flujo de ingreso, ya que ponen sus propios servicios Web XML a disponibilidad de otros.
- Cualquier servicio Web puede interactuar con cualquier otro servicio Web, a través del protocolo estándar SOAP.
- Permite que se comuniquen utilizando HTTP y XML. Cualquier dispositivo que trabaje con estas tecnologías puede ser huésped y acceder a los servicios Web.
- SOAP y la tecnología derivada de los servicios Web cuentan con el apoyo de varias compañías
- Mayor modularización y distribución de aplicaciones.

- Ahorran tiempo, dinero y disminuyen la complejidad del sistema. Los diseñadores de la aplicación ya no se preocupan de los detalles de puesta en práctica de los servicios que ellos invocan, reduciendo la duración del ciclo de creación ya que utilizan las funcionalidades desarrolladas por terceros.

- ***Inconvenientes de los Servicios Web con respecto aplicaciones distribuidas***

  - El utilizar XML para el envío de datos, hace que ocupen mayor ancho de banda en su transmisión.
  - Al ser una tecnología nueva ha hecho que herramientas y estándares estén evolucionando rápidamente
  - Dependencia de la disponibilidad: Servicios, Comunicaciones.
  - Los Web Services hacen uso de tecnologías que han sido atacadas anteriormente. Usando Web Services, la ausencia de técnicas de seguridad estándar es un obstáculo para la adopción de la tecnología.
  - La calidad de un Web Service es un parámetro que no queda claro, pero esta tecnología está en desarrollo y la mayoría de los protocolos en los que se basa aún no son estándar.

4.3 Modelo de programación de los Servicios Web

El modelo de programación sobre el cual está basado .NET es el de software como servicio, a través de **SOA** (Service Oriented Architecture - Arquitectura Orientada a Servicios). En base a este modelo los servicios se exponen una funcionalidad bien definida a la aplicación que la requiera. De manera que la aplicación final

ejecuta un conjunto de estos servicios, añade su lógica particular y le presenta una interfaz al usuario final.

Un servicio funciona como una aplicación independiente, no una isla. Teniendo sus propias reglas de negocio, datos, procedimientos de administración y operación. Expone toda su funcionalidad utilizando una interfaz basada en mensajes, por lo tanto carece de una interfaz de usuario.

SOA contiene tres actores: Un solicitante de servicio, un proveedor de servicios, y un registro de servicios. [WWW016]

> ***Un proveedor de servicio*** es responsable de crear una descripción de servicio, publicando o publicitando la descripción del servicio en uno o más registros de servicios, y recibir mensajes de invocación de servicios Web de uno o más solicitantes.
>
> ***Un solicitante de servicio*** es responsable de encontrar una descripción de servicio publicada en uno o más registros de servicios y de utilizar las descripciones de servicio para lograr invocar los servicios Web hospedados por los proveedores de servicios.
>
> ***Un registro de servicios*** es responsable de anunciar descripciones de servicios Web publicadas por los proveedores de servicios y permitir los solicitantes de servicios buscar en la colección de descripción de servicios contenidos en el registro de servicios. Una vez encontrada la información, el servicio del registro no es necesario y el resto de la interacción se da directamente entre el solicitante del servicio y el proveedor de servicio.

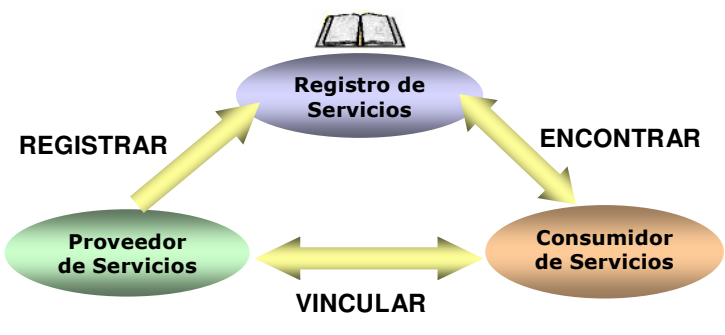

Figura 4.7 ***Proceso de los Servicios Web***

El siguiente diagrama ilustra la forma en que se utilizan los servicios Web entre un cliente y un servidor Web.

En el modelo de servicio Web:

1. Crea el archivo .asmx que incluye el espacio de nombre, clases, propiedades y métodos.

2. Declara métodos como métodos Web que pueden accederse por Internet.

El siguiente es un ejemplo de un archivo .asmx simple:

```
//Se inicializa las directivas de ASP.Net para indicar que es un servicio web y se
//indica el lenguaje usado para el código
<%@ WebService Language="VB" Class="MathService" %>
//Declaración de los espacios de nombres
 Imports System.Web.Services
 Imports System
//Nombre de la clase
Class MathService
//Indicar la descripción de la clase
  <WebMethod()>Public Function Add(int1 As Integer, int2 As Integer) As Integer
   return(int1 + int2)
End Function
 End Class
Cliente
```

Figura 4.8 ***Ejemplo de un Servicio Web***

Cliente Web

En el modelo de servicio Web, el cliente (si se accede a un servicio Web construido con .NET) realiza lo siguiente:

1. Invoca al servicio Web desde el explorador para determinar los métodos que están disponibles. Cuando invoca un servicio Web desde un explorador, accede a la página de descripción, que enumera los métodos que se incluyen en el servicio Web. El protocolo que se usa en este caso es HTTP y los datos se devuelven como XML.

2. Invoca un método del servicio Web desde el explorador. Cuando invoca un método de servicio Web desde un explorador, el protocolo que se usa es HTTP y los datos se devuelven como XML.

Servidor Web

También puede invocar métodos del servicio Web utilizando código en una página ASP.NET. Para invocar un servicio Web desde una página ASP.NET, deberá:

1. Encontrar los servicios que están disponibles. Esto requiere encontrar el URL para el servicio Web.

2. Usar la herramienta de generación proxy de línea de comando llamada WSDL.exe para generar un proxy para el servicio Web. WSDL.exe lee el documento **WSDL** del servicio Web y genera un archivo .vb o .cs que contiene código fuente para el proxy.

3. Compilar el proxy (el archivo .vb o .cs que creó en el paso anterior) en un archivo .dll en el directorio /bin del sitio Web.

4. Abrir la página Web ASP.NET.

    a. Crear una instancia de la clase proxy.
    b. Invocar los métodos del servicio Web.
    c. Usar los datos devueltos por el servicio Web.

Al invocar un servicio Web desde una aplicación Web se necesita crear una referencia Web dentro del proyecto al servicio Web. Esta referencia creará el objeto **PROXY** que se usará para la comunicación con el servicio Web mediante el protocolo SOAP. [WWW016]:

> ***Un Proxy*** es, *una imagen de la clase que se quiere representar, pero que no contiene la lógica de la aplicación* LIB[001].

La clase Proxy contiene ubicación y transporte de lógica. Un Proxy permite al cliente acceder a un servicio Web como si éste fuese un objeto COM local. El objeto Proxy debe estar en el mismo servidor que la aplicación Web. Cuando se agrega una referencia Web a un servicio Web, Visual Studio .NET automáticamente crea el código del objeto Proxy con el nombre reference.vb o reference.cs.

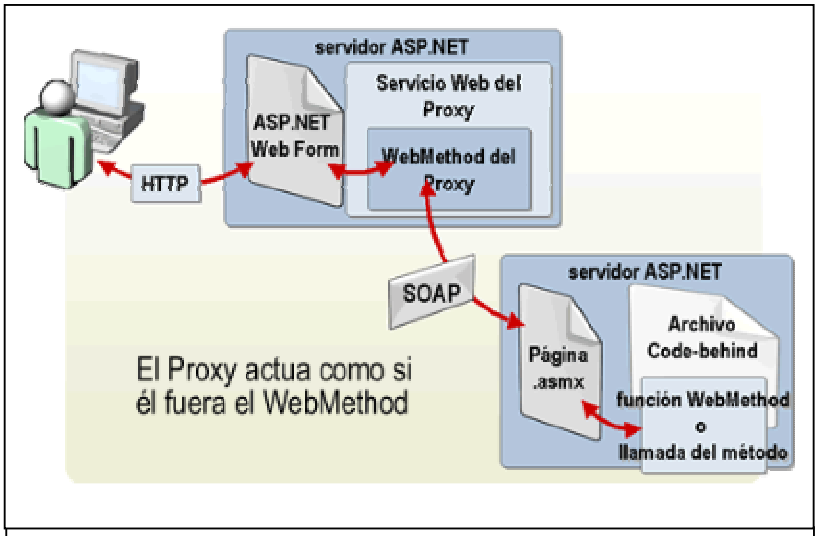

Figura 4.9 ***Descripción del objeto Proxy***

4.3 Arquitecturas para el desarrollo de los servicios Web

A continuación, las diferencias de las arquitecturas físicas útiles en modelos de aplicaciones .NET.

4.3.1 Aplicaciones en tres niveles con XML Web Services

Las aplicaciones en tres niveles que utilizan los XML Web Services son adecuadas tanto para aplicaciones basadas en Web como para aplicaciones simples. Esta arquitectura es útil cuando se necesita la capacidad de una aplicación de escritorio pero los usuarios se conectan desde muchos sitios distintos a través de una interfaz http.

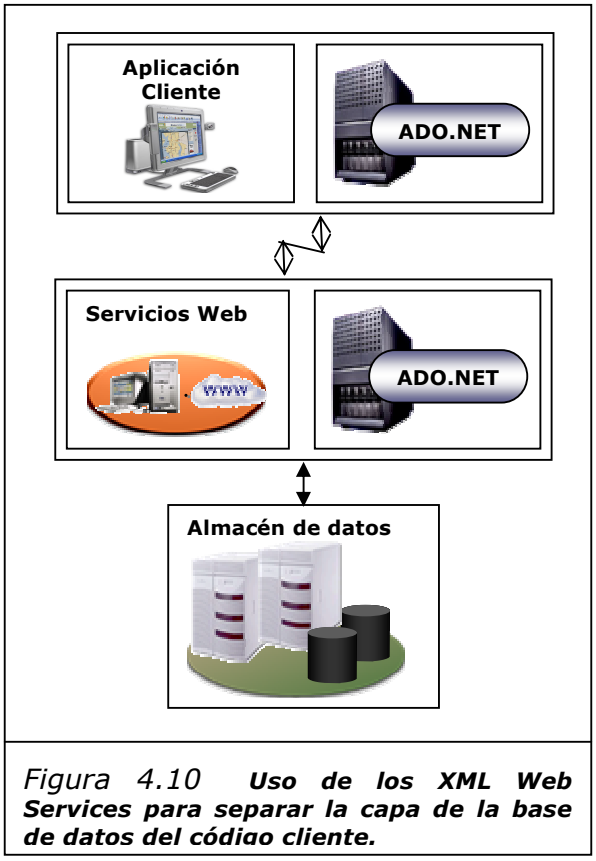

Figura 4.10 ***Uso de los XML Web Services para separar la capa de la base de datos del código cliente.***

- El SQL reside en el interior de los XML Web Services. Los conjuntos de datos se generan en el servidor y se envían al cliente como una secuencia XML, donde se pueden volver a convertir en conjunto de datos.

- El conjunto de datos devueltos por el XML Web Services se puede enlazarse directamente a los controles de los formularios.

- Los conjuntos de datos devueltos por los XML Web Services se pueden utilizar para cargar datos manualmente en los distintos controles de los formularios.

- Todas las reglas de empresa están dentro del código de los formularios.

- ***Ventajas de aplicaciones en tres niveles con Servicios Web XML***

  - El desarrollo puede ser rápido y sencillo, puesto que se pueden utilizar enlaces de datos para conectar directamente conjuntos de datos ADO.NET con muchos de los controles utilizados para generar la interfaz de usuario. Así, las funciones básicas de la aplicación se pueden instalar y ejecutar rápidamente.

  - Los usuarios pueden ejecutar la aplicación desde cualquier equipo con conexión a Internet (o a una Intranet).

  - El acceso a la base de datos está separado de su propio componente, por lo que no hace falta incrustar SQL en el código de la aplicación cliente.

  - El mantenimiento de los equipos cliente se reduce gracias a que únicamente los XML Web Services contienen información de conexión.

  - La capa de acceso a la base de datos se puede actualizar desde una ubicación central. Además, si hace un pequeño cambio en el código de esta capa, no hará falta volver a distribuir los componentes al cliente.

- ***Inconvenientes de aplicaciones en tres niveles con Servicios WebXML***

  - Todas las reglas de empresa están contenidas en el código cliente, de modo que, si tiene que cambiar una regla de empresa, deberá actualizar todos los clientes. A menos que utilice un método de actualización automático, el proceso de mantenimiento puede convertirse en una labor interminable. El uso de SQL Server permite guardar algunas reglas de empresa en procedimientos almacenados para disminuir el costo y la duración del mantenimiento.

- Todos los nombres de campo están en el código fuente o de las propiedades de control. Si modifica un nombre de campo, se debe buscar y modificar todos los casos que haya en la aplicación. Cuando además hay enlaces de datos presentes, se debe comprobar todos los formularios y cambiar las propiedades.

- La conexión a través de una interfaz HTTP es más lenta que la conexión directa con la base de datos en un 20%.

- Si el usuario no tiene acceso a Internet (o a una intranet), no podrá usar la aplicación.

4.3.2. Aplicaciones N niveles con Servicios Web XML

En este diagrama se puede observar cómo se usan los XML Web Services para tener acceso a la capa de datos. El conjunto de datos escrito se devuelve, a través de la capa HTTP, a la capa de reglas de empresa. La aplicación cliente puede entonces utilizar el conjunto de datos para representar los datos en la interfaz de usuario. [WWW016]

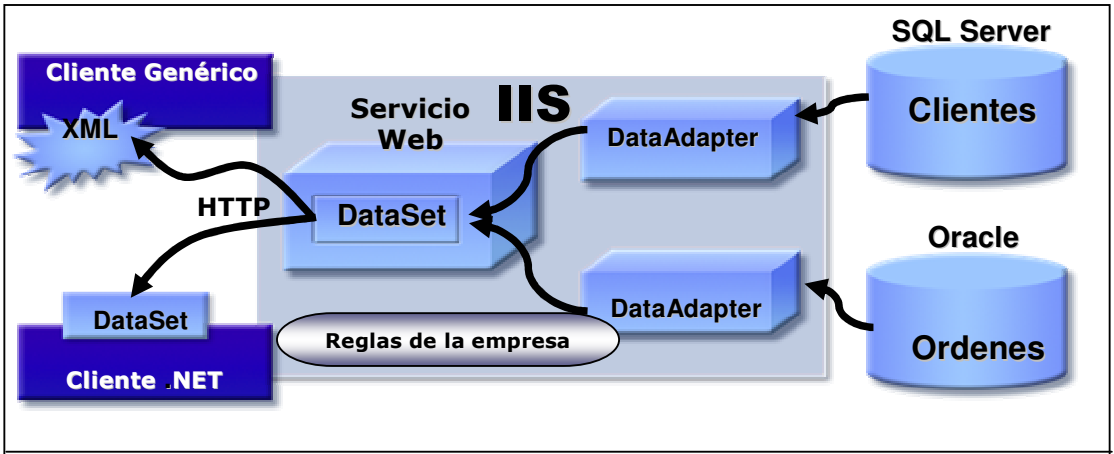

Figura 4.11 ***Separación de los procesos de empresa en distintos equipos para facilitar la implementación y el mantenimiento***

El diseño de aplicaciones con los XML Web Services y n niveles es útil cuando se busca la capacidad de una aplicación de escritorio pero los usuarios se van a conectar desde ubicaciones remotas y necesitan obtener los datos a través de una interfaz HTTP. Mantener las reglas de empresa en la parte cliente mejora el tráfico de la red, pero puede suponer problemas de actualización y mantenimiento si las reglas cambian con frecuencia. Dado que .NET puede copiar nuevas DLL sin registrarse, este problema ya no es tan grave como en el pasado.

Esta arquitectura es útil para aplicaciones basadas en Web donde los datos son proporcionados por un servidor Web, pero mostrados por otra aplicación Web en otro servidor Web.

Para crear este tipo de aplicación, se aplica las siguientes técnicas de desarrollo.

- Crear la interfaz de usuario de la aplicación cliente mediante Windows Forms o Formularios Web.
- Crear un componente de reglas de empresa como un proyecto separado de biblioteca de clases.
- Crear un componente de capas de datos como un proyecto separado de biblioteca de clases. Esta capa de datos utiliza clases para envolver el acceso a cada tabla. Se recomienda usar conjuntos de datos escritos; proporcionan la flexibilidad de la clase de conjunto de datos y escritura estable para cada columna de las tablas.

- ***Ventajas de aplicaciones N niveles con Servicios Web XML***

  - Centraliza las reglas de empresa en un componente fácil de crear, usar y reutilizar, con lo que el desarrollo y el mantenimiento resultan tareas sencillas.

- Proporciona un lenguaje de alto nivel con el que desarrollar reglas de empresa, por oposición a crear procedimientos almacenados y lenguaje SQL limitado para comprobar reglas de empresa.

- Centraliza el acceso a los datos en un componente. Esto significa que se repetirá menos código a lo largo de la aplicación; todos los formularios que necesitan tener acceso a una tabla concreta utilizan siempre el mismo componente.

- Si utiliza conjuntos de datos escritos, podrá consultar los nombres de columna en lugar de tener que memorizarlos.

- Las rutinas de acceso a datos centralizados facilitan el mantenimiento, puesto que cualquier cambio que se haga en una rutina de acceso a datos sólo hay que hacerlo una vez.

- Proporciona la flexibilidad para separar componentes en diferentes equipos físicos en cualquier momento, con lo que aumenta la escalabilidad y se logra una mejor centralización del código.

- La capa de acceso a datos es centralizada.

- Capacidad de ampliación de la aplicación; se puede agregar una granja Web para manejar un número muy alto de solicitudes de usuario a la base de datos.

- Los usuarios se pueden conectar a través de Internet y tener acceso a los datos desde una aplicación de escritorio o basada en Web.

- ***Inconvenientes de aplicaciones en n niveles con los XML Web Services***

  - Su desarrollo requiere algo más de tiempo porque hay que generar componentes separados.

- Hay que controlar un número mayor de componentes, lo cual puede resultar difícil de comprender para los nuevos programadores.

- Si los XML Web Services no funcionan, no podrá utilizar la aplicación.

4.4 .Net vs otras tecnologías para Servicios Web

Actualmente existen 2 tecnologías líderes en la competencia de ofrecer soluciones para desarrollar los servicios Web. J2EE y .NET. Adicionalmente están evolucionando con gran aceptación en el mercado otras tecnologías entre ellas un proyecto llamado MONO (.Net para Linux); el cual es una iniciativa de desarrollo abierto patrocinada por **Ximian[]** que está trabajando en desarrollar una versión basada en Linux de código abierto **(open source**) de la plataforma de desarrollo de Microsoft .NET. Su objetivo es permitir a los desarrolladores de Linux construir e implementar aplicaciones .NET multiplataforma. Este proyecto implementará varias tecnologías desarrolladas por Microsoft que han sido enviadas al **ECMA** para su estandarización. [LIB004]:

4.4.1 J2EE vs .NET

- ***Ventajas de .Net frente a J2EE***

a) Una ventaja de .Net frente a **J2EE** es la posibilidad de emplear múltiples lenguajes de programación, ya que J2EE sólo trabaja con uno: Java. La alta variedad de lenguajes es obligatoria por las necesidades de los programadores. Un lenguaje moderno y orientado a objetos como Java puede resultar totalmente ineficaz e inadecuado al abordar problemas que involucren cálculos matemáticos masivos y complejos.

b) Con respecto a los **BYTECODES** de Java y el MSIL de .NET, MSIL es de mayor nivel ya que introduce la seguridad de tipos, y el código MSIL puede ser verificado antes de su ejecución para garantizar su seguridad. En Java en cambio los bytecodes son interpretados, mientras que el CLR compila el código MSIL a código nativo, obteniendo un mayor rendimiento la plataforma .NET.

c) Con respecto al soporte para el desarrollo de clientes, J2EE posee a los **SERVLETS** y a las páginas JSP, mientras que .NET posee a ASP.NET. En ASP.NET es posible utilizar cualquier lenguaje y puede obtener un rendimiento superior a los servlets y **JSP**s, ASP.NET también introduce un modelo de desarrollo basado en componentes similar al existente en el desarrollo de las interfaces de aplicaciones de escritorio que permite abstraerse del navegador cliente, lo que lo dota de una potencia muy superior a los servlets y JSPs.

d) Las herramientas de desarrollo incluidas por Microsoft en su Visual Studio .Net son simples, intuitivas y sencillas de manejar que las herramientas de desarrollo equivalentes en J2EE suministradas por otras empresas (entre ellas Sun). Cualquier programador manejará rápidamente la programación de interface de usuario de Visual Studio .Net, al igual que sucedía con sus versiones anteriores.

e) C# es un lenguaje interesante, fácil de aprender por los programadores de Java (Microsoft ofrece un conversor de Java a C#), que en caso de estandarizarse podría resultar un lenguaje muy conveniente para ciertas tareas de programación en diferentes plataformas. No está escrito en ninguna parte que los lenguajes no puedan evolucionar y, en ese sentido, C# es una rama evolutiva más del árbol de los lenguajes orientados a objetos.

f) Microsoft ha impulsado con gran energía los servicios Web y ha resaltado su importancia entre toda la comunidad de desarrolladores (utilicen o no los productos de esta compañía). La plataforma .Net se ha diseñado considerando los servicios Web siendo estos propios de la plataforma. Ofrece una nueva versión de ASP, ASP .Net, que puede considerarse un entorno de programación potencializado en lugar de un entorno basado en scripts. .Net con respecto a servicios Web tiene su propia plataforma, y aunque J2EE respondió ya con el lanzamiento del Java Web Services Developer Pack. La facilidad, rapidez y sencillez con la que se pueden construir servicios Web con el Asistente de servicios Web de Visual Studio .Net son superiores a las de las herramientas para construir servicios Web dentro del entorno de J2EE.

- ***Ventajas de J2EE frente a .Net***

a) Las implementaciones de J2EE pueden adquirirse a distintas compañías, mientras que .Net solo puede comprarse a Microsoft. Al haber distintas organizaciones implementando J2EE ofrece mayor variedad para los usuarios finales y permite la existencia de una cierta competencia entre ellas para obtener mejores productos que no existe en el caso de Microsoft.

b) Debido al proceso evolutivo de los productos de Microsoft, y en muchos casos, por motivos de compatibilidad la seguridad frente a virus informáticos de los productos de Microsoft es menor que los basados en Java, pues desde un comienzo Java se fundamentó en un estricto modelo de doble seguridad.

c) Las aplicaciones Java corren sobre varios sistemas operativos (como Windows 2000, Solaris, Mac OS, Windows 9x, Linux, y sistemas operativos para

dispositivos móviles) y de arquitecturas hardware. Hasta ahora, .Net corre solamente sobre sistemas operativos de Microsoft (aunque esta situación puede cambiar en el futuro), siendo J2EE el único entorno de desarrollo que ofrece una independencia real de la plataforma.

d) Java es una tecnología abierta (el código de la plataforma completa puede ser revisado y estudiado) y se basa en gran parte en estándares de organizaciones de normalización y empresariales. Esto posibilita que los desarrolladores puedan conocer y entender completamente cómo hace las cosas Java y aprovecharlo para sus aplicaciones y al basarse en estándares empresariales, simplifica la integración con productos de múltiples compañías. En contraposición, solo el código fuente del lenguaje C# de la plataforma .Net ha sido abierto al público aunque Microsoft permite a compañías con las que le unen intereses comunes el acceso al código fuente de ciertas partes de .Net.

e) Aunque Java fue creado originalmente por una compañía: Sun MicroSystems, J2EE es ahora el producto de la colaboración de más de 400 empresas y organizaciones de todo tipo. La plataforma .Net es y será el producto de una sola compañía sobretodo teniendo en cuenta que la mayor parte de su código no es público.

f) La tecnología Java goza ya de una cierta trayectoria. Y J2EE ha probado su eficacia en muchos entornos y situaciones empresariales distintas.

La presente tabla muestra un resumen de las similitudes entre J2EE Y .Net

CARACTERISTICAS	SIMILITUDES J2EE Y .NET
Acceso	**RMI** en J2EE. El Framework en .NET
Modelo de Componentes	En J2EE el modelo de componentes viene dado por los **JAVABEANS Y ENTERPRISE JAVABENS**. Los eventos en el .NET aparecen incluidos en el sistema de tipos común, CTS y son soportados por distintos lenguajes.
Despliegue	Propósito empresarial
Escalabilidad	Ambas tecnologías proporcionan soporte para adaptar antiguos sistemas, todavía en funcionamiento, para poder utilizar el concepto de servicio Web. Por un lado J2EE ofrece para la integración el **JCA** (J2EE Conector Architecture) capaz de adaptar sistemas. .NET ofrece esta integración a través de 'Host Integration Server 2000': COM TI (COM Transaction Integrator) para los grandes sistemas 'mainframe', y BizTalk Server 2000 que integran sistemas basados en protocolos B2B.
Servidores de aplicación	Los servidores de aplicaciones J2EE y .Net proporcionan diferenciación entre acceso de datos y lógica de negocio, separados por una capa intermedia de presentación implementada mediante ASP .Net (.Net) ó **SERVLETS** (J2EE).
Programación Orientada a Objetos	Visual Basic .Net y C# son lenguajes orientados a objetos, al igual que Java, y en su diseño ha tenido mucha importancia la existencia de Internet.
Multiplataforma	J2EE y .Net son multiplataforma. Al usar .Net una compilación en dos pasos, permitiría proporcionar en el futuro entornos de ejecución para diferentes plataformas de forma similar a Java
Clientes pesados	Para el desarrollo de los clientes pesados (fat clients) ambas plataformas poseen sus librerías y su colección de controles. Para J2EE esta Java Swing, y para .NET están los WinForms, los cuales son bastante similares, diferenciándose en la gestión de los eventos, que con Java se realiza mediante clases internas y en .NET se realiza mediante el sistema de eventos basado en delegados
API para el acceso a fuentes de datos	Ambas plataformas poseen APIs para el acceso a fuentes de datos, JDBC para J2EE y ADO.NET para .NET..NET tiene la posibilidad de trabajar en modo desconectado con los DataSet y está integrado con XML.
XML	Con respecto a XML, ambas plataformas hacen uso de él. Sin embargo .NET esta mucho más integrado con XML que J2EE, haciendo uso de él a varios niveles como en ADO.NET, los servicios Web, SOAP, etc.

Tabla 4.1 ***Similitudes entre la tecnología J2EE y .Net***

A continuación las principales diferencias tanto en tecnología como servicios entre Microsoft .NET y J2EE.

Característica	.NET	J2EE
Tipo de tecnología	Producto	Estándar
Empresas que lo ofrecen	Microsoft	Más de 30
Librerías de desarrollo	.NET Framework SDK	Java core API
Intérprete	CLR	JRE
Páginas dinámicas	ASP.NET	Servlets, JSP
Componentes	.NET Managed Components	EJB
Acceso a bases de datos	ADO.NET	JDBC, SQL/J
Servicios Web	SOAP, WDSL, UDDI	SOAP, WDSL, UDDI
Interfaces gráficas	Win Forms y Web Forms	Java Swing
Herramienta de programación	Visual Studio .NET Interoperabilidad	Dependiente del fabricante
Còdigo	Cerrado	Abierto
Sistema Operativo	Diversos Sistemas Operativos Microsoft	Diversos Sistemas Operativos
Seguridad	Seguridad implementada en el desarrollo	Mayor seguridad (Virus)
Lenguajes utilizados	C#, Visual Basic, C++, otros	Java
Lenguaje intermedio	IL	Bytecodes

Tabla 4.2 ***Resumen comparativo entre la tecnología .Net y J2EE***
www.ciberteca.net/articulos/programacion/net/analogias.asp

4.4.2 Arquitecturas J2EE y .NET

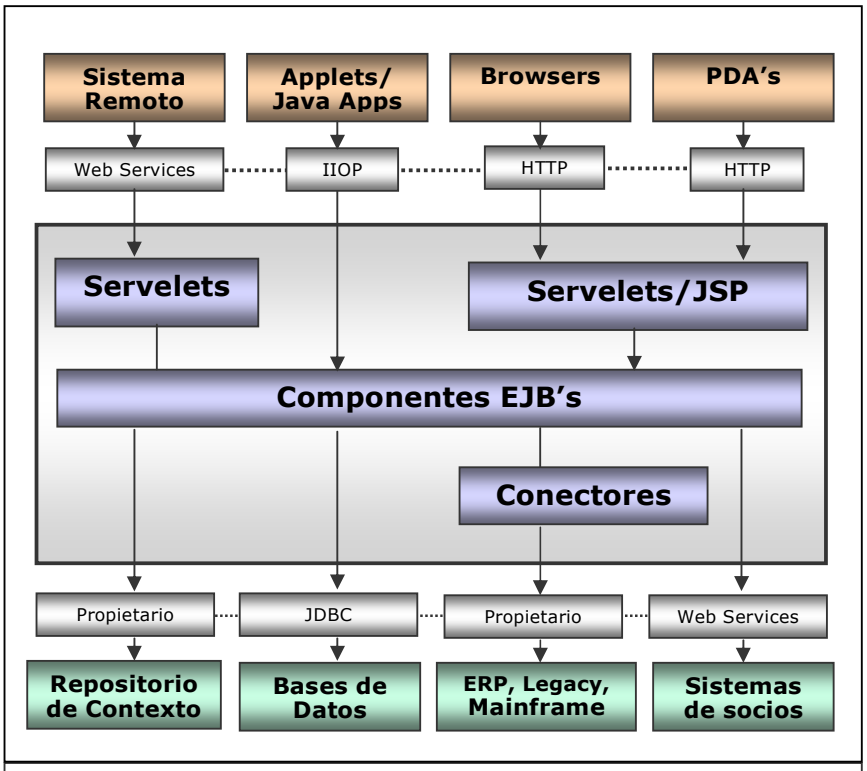

Figura 4.12 ***Arquitectura J2EE***

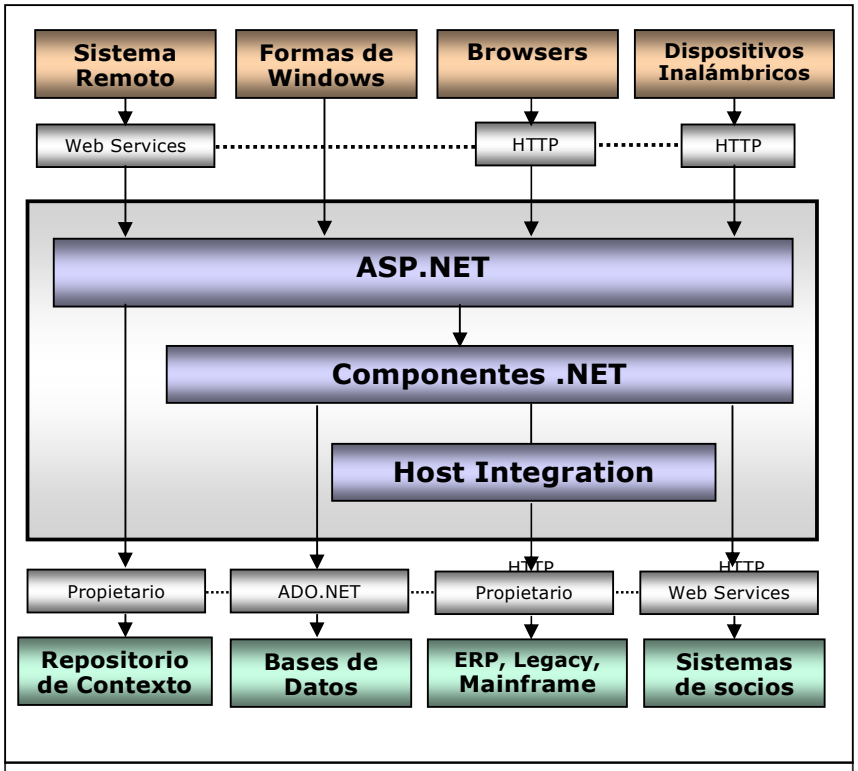

Figura 4.13 ***Arquitectura .NET***

En conclusión, J2EE ofrece una solución basada en Java. J2EE es una especificación donde están involucradas numerosas compañías de primer nivel mundial para asegurar su éxito. Una de las mayores desventajas, es que su elección implica la utilización de un único lenguaje de programación. Pero, sin embargo, una de las mayores ventajas es la libre elección del sistema operativo sobre el que se va a desarrollar. Por su parte, Microsoft .NET ofrece una solución con mayor rendimiento, escalabilidad y más fácil de implantar en un mismo entorno.