Laboratorio 5 – Servicio Web SOAP con Windows Communication Foundation (WCF)

¿Qué es un Servicio WEB?

Un Servicio Web o Web Services en Ingles , es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma.

En la arquitectura de servicios web existen tres partes: proveedor de servicios web, el que pide el servicio web y el publicador. El proveedor de servicios envía al publicador del servicio un fichero WSDL con la definición del servicio web. El que pide el servicio contacta con el publicador y descubre quién es el proveedor (protocolo WSDL) y contacta con el proveedor (protocolo SOAP). El proveedor valida la petición de servicio y envía el dato estructurado en formato XML utilizando el protocolo SOAP. El fichero XML es validado de nuevo por el que pide el servicio utilizando un fichero XSD.

XML

eXtensible Markup Language, traducido como "Lenguaje de Marcado Extensible" o "Lenguaje de Marcas Extensible", es un meta-lenguaje que permite definir lenguajes de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible. Proviene del lenguaje SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML) para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí o integrar información.

```xml
<?xml version="1.0"?>
<Catalog>
   <Book id="bk101">
      <Author>Garghentini, Davide</Author>
      <Title>XML Developer's Guide</Title>
      <Genre>Computer</Genre>
      <Price>44.95</Price>
      <PublishDate>2000-10-01</PublishDate>
      <Description>An in-depth look at creating applications
      with XML.</Description>
   </Book>
   <Book id="bk102">
      <Author>Garcia, Debra</Author>
      <Title>Midnight Rain</Title>
      <Genre>Fantasy</Genre>
      <Price>5.95</Price>
      <PublishDate>2000-12-16</PublishDate>
      <Description>A former architect battles corporate zombies,
      an evil sorceress, and her own childhood to become queen
      of the world.</Description>
   </Book>
</Catalog>
```

Definición de Esquema XML

Mas conocido como XSD (XML Schema Definition), es un mecanismo para comprobar la validez de un documento XML, es decir, definir su estructura: qué elementos, qué tipos de datos, que atributos, en qué orden, cuántas veces se repiten, etc.

Un esquema XSD es un documento XML con la siguiente estructura:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="alumnos" />
</xs:schema>
```

Donde aparecen los siguientes elementos:

- La etiqueta xml define la versión de XML utilizada y la codificación de caracteres del documento.
- La etiqueta raíz xs:schema define el espacio de nombres del esquema XSD así como una serie de atributos opcionales que podéis ampliar aquí.
- La etiqueta xs:element que contiene la definición del elemento raíz del documento XML al que se le va a aplicar el esquema y que explicaremos más adelante.

El documento XML deberá referenciar el esquema XSD con el que se va a validar utilizando la siguiente cabecera:

```
<?xml version="1.0" encoding="UTF-8"?>
<alumnos xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="alumnos.xsd">
  ...
</alumnos>
```

Donde la etiqueta raíz del documento define los atributos:

- ***xmlns:xsi*** para declarar el espacio de nombres del esquema XSD.
- ***xsi:noNamespaceSchemaLocation*** para vincular el documento XML con el esquema local XSD.

Componentes básicos de un esquema XSD

Los componentes imprescindibles para construir un esquema XSD son los siguientes:

- xs:element
- xs:attribute

xs:element

Este componente permite declarar los elementos del documento XML. Entre otros, los principales atributos que podemos utilizar en la declaración son los siguientes:

- ***name:*** Indica el nombre del elemento. Obligatorio si el elemento padre es <xs:schema>.
- ***ref:*** Indica que la declaración del elemento se encuentra en otro lugar del esquema. No se puede usar si el elemento padre es <xs:schema>. No puede aparecer junto con name.
- ***type:*** Indica el tipo de dato que almacenará el elemento. No puede aparecer junto con ref.
- ***default:*** Es el valor que tomará el elemento al ser procesado si en el documento XML no ha recibido ningún valor. Sólo se puede usar con tipo de dato textual.
- ***fixed:*** Indica el único valor que puede contener el elemento en el documento XML. Sólo se puede usar con tipo de dato textual.
- ***minOccurs:*** Indica el mínimo número de ocurrencias que deben aparecer de ese elemento en el documento XML. No se puede usar si el elemento padre es <xs:schema>. Va desde 0 hasta ilimitado (unbounded). Por defecto 1.

- ***maxOccurs:*** Indica el máximo número de ocurrencias que pueden aparecer de ese elemento en el documento XML. No se puede usar si el elemento padre es <xs:schema>. Va desde 0 hasta ilimitado (unbounded). Por defecto 1.

```
<xs:element name="nombre" type="xs:string" default="TicArte" minOccurs="1" maxOccurs="unbounded" />
```

```
<xs:element ref="contacto" minOccurs="1" maxOccurs="unbounded" />
```

xs:attribute

Este componente permite declarar los atributos de los elementos del documento XML. Entre otros, los principales atributos que podemos utilizar en la declaración son los siguientes:

- ***name:*** Indica el nombre del atributo.
- ***ref:*** Indica que la declaración del atributo se encuentra en otro lugar del esquema. No puede aparecer junto con name.
- ***type:*** Indica el tipo de dato que almacenará el atributo. No puede aparecer junto con ref.
- ***use:*** Indica si la existencia del atributo es opcional (optional), obligatoria (required) o prohibida (prohibited). Por defecto opcional.
- ***default:*** Es el valor que tomará el elemento al ser procesado si en el documento XML no ha recibido ningún valor. Sólo se puede usar con tipo de dato textual.
- ***fixed:*** Indica el único valor que puede contener el elemento en el documento XML. Sólo se puede usar con tipo de dato textual.

```
<xs:attribute name="moneda" type="xs:string" default="euro" use="required" />
```

```
<xs:attribute ref="moneda" use="required" />
```

Tipos de datos

Tipos de datos simples y complejos

Existe dos grandes grupos de tipos de datos que se pueden utilizar en los esquemas XSD:

- Tipos de datos simples (xs:simpleType): Se dividen en los siguientes.
- Tipos de datos predefinidos.
- Tipos de datos construidos con nuestras propias restricciones y basados en los tipos de datos predefinidos.
- Tipos de datos complejos (xs:complexType): Se dividen en los siguientes.
- Elementos dentro de otros elementos.
- Elementos que tienen atributos.
- Elementos mixtos que tienen datos y otros elementos.
- Elementos vacíos.

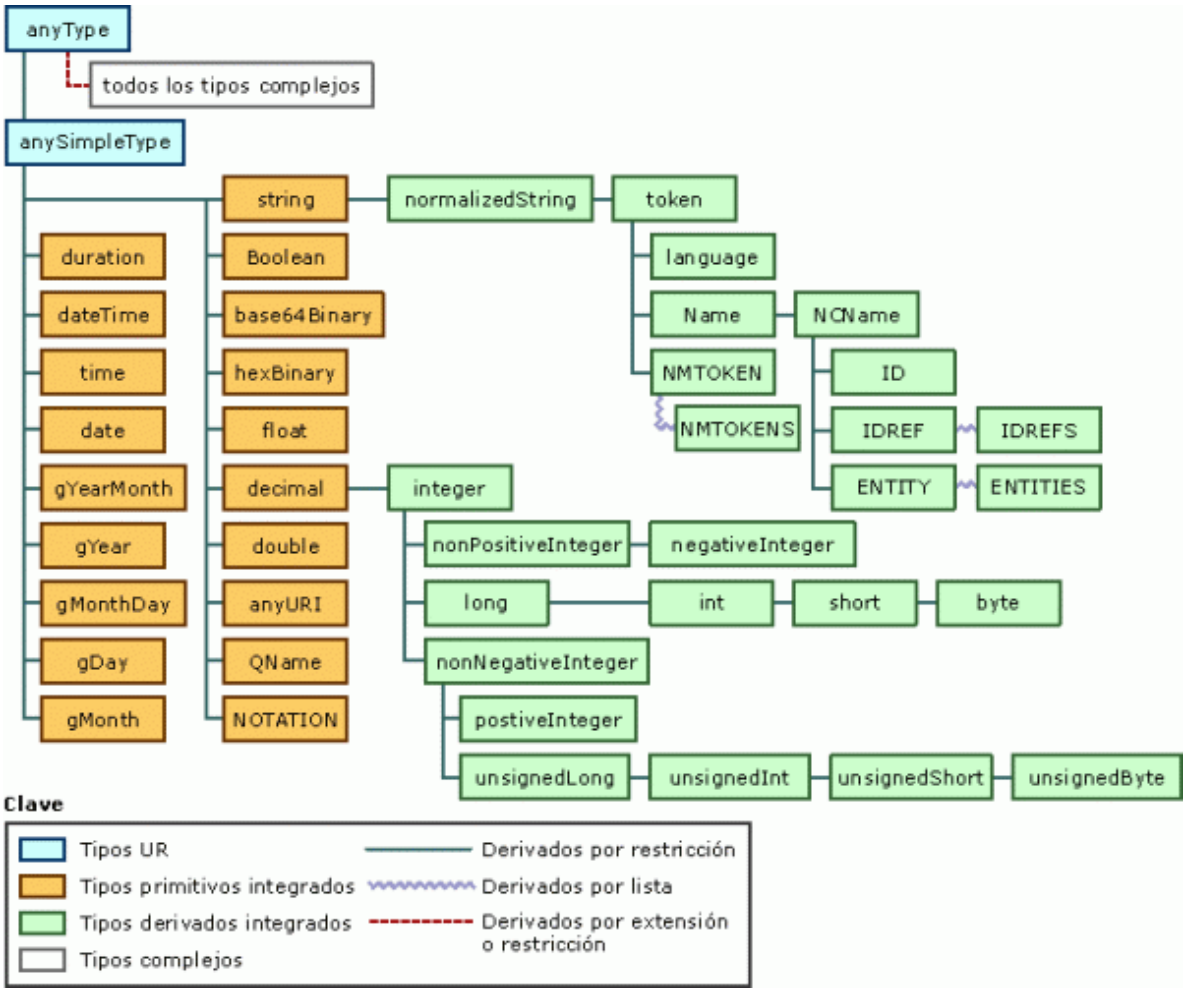
Tipos de datos predefinidos

Son 44 los tipos de datos que define el esquema XSD, clasificados de una manera jerárquica, en el que los elementos hijos poseen las mismas características del padre más alguna particularidad añadida que los distinga

El tipos de datos predefinido principal es xs:anyType, el más genérico y del cual derivan el resto de tipos de datos predefinidos como podemos ver en la siguiente imagen.

Los tipos de datos predefinidos los podemos dividir en:

- Primitivos: Descienden directamente de xs:anySimpleType.
- No primitivos: Descienden de alguno de los primitivos.



Los tipos de datos primitivos que podemos utilizar son los siguientes.

Tipo de Dato	Descripción
string	Cadena de caracteres.
boolean	Valores booleanos que pueden ser **true** o **false**.
decimal	Números reales.
float	Números en punto flotante de 32 bits.
double	Números en punto flotante de 64 bits.
duration	Duración representada en "Años + Meses + Días + Horas + Minutos + Segundos" (P10Y5M4D10H12M50S).
dateTime	Fecha y hora en formato CCYY-MM-DDThh:mm:ss donde CC es el siglo, YY el año, MM el mes, DD el día, precedido del carácter (-) indica un número negativo. La T es el separador de hh horas, mm minutos, y ss segundos. Puede seguirse de una Z para indicar la zona UTC (Universal Time Coordinated).

time	Hora en formato hh:mm:ss.sss.
date	Fecha en formato CCYY-MM-DD.
gYearMonth	Sólo el año y mes en formato Gregoriano CCYY-MM.
gYear	Sólo el año en formato Gregoriano CCYY.
gMonthDay	Sólo el mes y el día en formato Gregoriano --MM-DD.
gDay	Sólo el día en formato Gregoriano ---DD.
gMonth	Sólo el mes en formato Gregoriano --MM--.
hexBinary	Secuencia de dígitos hexadecimales.
base64Binary	Secuencia de dígitos hexadecimales en base 64.
anyURI	Cualquier identificador URI.
QName	Cualquier nombre cualificado del espacio de nombres
NOTATION	Tipo de datos para atributo compatible con DTD.

Los tipos de datos no primitivos que podemos utilizar son los siguientes.

Tipo de dato	Descripción
normalizedString	Representa cadenas normalizadas de espacios en blanco. Este tipo de datos se deriva de string.
token	Representa cadenas convertidas en tokens. Este tipo de datos se deriva de normalizedString.
language	Representa identificadores de lenguaje natural (definidos por RFC 1766). Este tipo de datos se deriva de token.
IDREFS	Representa el tipo de atributo IDREFS. Contiene un conjunto de valores de tipo IDREF.
ENTITIES	Representa el tipo de atributo ENTITIES. Contiene un conjunto de valores de tipo ENTITY.
NMTOKEN	Representa el tipo de atributo NMTOKEN. NMTOKEN es un juego de caracteres de nombres (letras, dígitos y otros caracteres) en cualquier combinación. A diferencia de Name y NCName, NMTOKEN, no tiene restricciones del carácter inicial. Este tipo de datos se deriva de token.
NMTOKENS	Representa el tipo de atributo NMTOKENS. Contiene un conjunto de valores de tipo NMTOKEN.
Name	Representa nombres en XML. Name es un token que empieza con una letra, carácter de subrayado o signo de dos puntos, y continúa con caracteres de nombre (letras, dígitos y otros caracteres). Este tipo de datos se deriva detoken.
NCName	Representa nombres sin el signo de dos puntos. Este tipo de datos es igual que Name, excepto en que no puede empezar con el signo de dos puntos. Este tipo de datos se deriva de Name.

ID	Representa el tipo de atributo ID definido en la recomendación de XML 1.0. El IDno debe incluir un signo de dos puntos (NCName) y debe ser único en el documento XML. Este tipo de datos se deriva de NCName.
IDREF	Representa una referencia a un elemento que tiene un atributo ID que coincide con el ID especificado. IDREF debe ser un NCName y tener un valor de un elemento o atributo de tipo ID dentro del documento XML. Este tipo de datos se deriva de NCName.
ENTITY	Representa el tipo de atributo ENTITY definido en la recomendación de XML 1.0. Es una referencia a una entidad sin analizar con un nombre que coincide con el especificado. ENTITY debe ser un NCName y declararse en el esquema como nombre de entidad sin analizar. Este tipo de datos se deriva de NCName.
integer	Representa una secuencia de dígitos decimales con un signo inicial (+ o -) opcional. Este tipo de datos se deriva dedecimal.
nonPositiveInteger	Representa un número entero menor o igual que cero. nonPositiveInteger consta de un signo negativo (-) y una secuencia de dígitos decimales. Este tipo de datos se deriva de integer.
negativeInteger	Representa un número entero menor que cero. Consta de un signo negativo (-) y una secuencia de dígitos decimales. Este tipo de datos se deriva de nonPositiveInteger.
long	Representa un entero con un valor mínimo de -9223372036854775808 y un valor máximo de 9223372036854775807.Este tipo de datos se deriva de integer.
int	Representa un entero con un valor mínimo de -2147483648 y un valor máximo de 2147483647. Este tipo de datos se deriva de long.
short	Representa un entero con un valor mínimo de -32768 y un valor máximo de 32767. Este tipo de datos se deriva deint.
byte	Representa un entero con un valor mínimo de -128 y un valor máximo de 127. Este tipo de datos se deriva de short.
nonNegativeInteger	Representa un número entero mayor o igual que cero. Este tipo de datos se deriva de integer.
unsignedLong	Representa un entero con un valor mínimo de cero y un valor máximo de 18446744073709551615. Este tipo de datos se deriva de nonNegativeInteger.
unsignedInt	Representa un entero con un valor mínimo de cero y un valor máximo de 4294967295. Este tipo de datos se deriva de unsignedLong.
unsignedShort	Representa un entero con un valor mínimo de cero y un valor máximo de 65535. Este tipo de datos se deriva deunsignedInt.
unsignedByte	Representa un entero con un valor mínimo de cero y un valor máximo de 255. Este tipo de datos se deriva deunsignedShort.
positiveInteger	Representa un número entero mayor que cero. Este tipo de datos se deriva de nonNegativeInteger.

Lenguaje de Descripción de Servicios

WSDL, las siglas de Web Services Description Language, es un formato del Extensible Markup Language (XML) que se utiliza para describir servicios web (WS). La versión 1.0 fue la primera recomendación por parte del W3C y la versión 1.1 no alcanzó nunca tal estatus. La versión 2.0 se convirtió en la recomendación actual por parte de dicha entidad.

WSDL describe la interfaz pública a los servicios Web. Está basado en XML y describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Las operaciones y mensajes que soporta se describen en abstracto y se ligan después al protocolo concreto de red y al formato del mensaje.

Así, WSDL se usa a menudo en combinación con SOAP y XML Schema. Un programa cliente que se conecta a un servicio web puede leer el WSDL para determinar qué funciones están disponibles en el servidor. Los tipos de datos especiales se incluyen en el archivo WSDL en forma de XML Schema. El cliente puede usar SOAP para hacer la llamada a una de las funciones listadas en el WSDL.

El WSDL nos permite tener una descripción de un servicio web. Especifica la interfaz abstracta a través de la cual un cliente puede acceder al servicio y los detalles de cómo se debe utilizar.

Estructura del WSDL

Tipos de datos

<types>: Esta sección define los tipos de datos usados en los mensajes. Se utilizan los tipos definidos en la especificación de esquemas XML.

Mensajes

<message>: Aquí definimos los elementos de mensaje. Cada mensaje puede consistir en una serie de partes lógicas. Las partes pueden ser de cualquiera de los tipos definidos en la sección anterior.

Tipos de puerto

<portType>: Con este apartado definimos las operaciones permitidas y los mensajes intercambiados en el Servicio.

Bindings

<binding>: Especificamos los protocolos de comunicación usados.

Servicios

<service>: Conjunto de puertos y dirección de los mismos. Esta parte final hace referencia a lo aportado por las secciones anteriores.

Con estos elementos no sabemos qué hace un servicio pero sí disponemos de la información necesaria para interactuar con él (funciones, mensajes de entrada/salida, protocolos...)

El siguiente es un ejemplo de la descripción de un servicio Web. Esta estructura se optiene al incluir al final de la URL del Servicio **?WSDL**

Ej.- http://xxxxxxx.cl/servicioWeb**?WSDL**

```
<definitions name="StockQuote"
      targetNamespace="http://example.com/stockquote.wsdl"
      xmlns:tns="http://example.com/stockquote.wsdl"
      xmlns:xsd1="http://example.com/stockquote.xsd"
      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
      xmlns="http://schemas.xmlsoap.org/wsdl/">

  <types>
    <schema targetNamespace="http://example.com/stockquote.xsd"
        xmlns="http://www.w3.org/2000/10/XMLSchema">
      <element name="TradePriceRequest">
        <complexType>
          <all>
            <element name="tickerSymbol" type="string"/>
          </all>
        </complexType>
      </element>
      <element name="TradePrice">
        <complexType>
          <all>
            <element name="price" type="float"/>
          </all>
        </complexType>
      </element>
    </schema>
  </types>

  <message name="GetLastTradePriceInput">
    <part name="body" element="xsd1:TradePriceRequest"/>
  </message>

  <message name="GetLastTradePriceOutput">
    <part name="body" element="xsd1:TradePrice"/>
  </message>

  <portType name="StockQuotePortType">
    <operation name="GetLastTradePrice">
      <input message="tns:GetLastTradePriceInput"/>
      <output message="tns:GetLastTradePriceOutput"/>
    </operation>
```

```
  </portType>

  <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
    <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetLastTradePrice">
      <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>

  <service name="StockQuoteService">
    <documentation>My first service</documentation>
    <port name="StockQuotePort" binding="tns:StockQuoteSoapBinding">
      <soap:address location="http://example.com/stockquote"/>
    </port>
  </service>

 </definitions>
```

¿Que es Windows Communication Foundation (WCF)?

Windows Communication Foundation o WCF (también conocido como Indigo), es la plataforma de mensajería que forma parte de la API de la Plataforma .NET 3.0 (antes conocida como WinFX, y que no son más que extensiones para la versión 2.0). Se encuentra basada en la Plataforma .NET 2.0 y de forma predeterminada se incluye en el Sistema Operativo Microsoft Windows Vista.

Fue creada con el fin de permitir una programación rápida de sistemas distribuidos y el desarrollo de aplicaciones basadas en arquitecturas orientadas a servicios (también conocido como SOA), con una API simple; y que puede ejecutarse en una máquina local, una LAN, o sobre Internet en una forma segura.

Arquitectura WCF

Contratos y descripciones

Los contratos definen varios aspectos del sistema de mensajes. El contrato de datos describe cada parámetro que constituye cada mensaje que un servicio puede crear o utilizar. Los documentos de Lenguaje de definición de esquemas XML (XSD) definen los parámetros de mensaje, permitiendo a cualquier sistema que entienda XML procesar los documentos. El contrato del mensaje define partes específicas del mensaje utilizando los protocolos SOAP y permite el control más fino sobre las partes del mensaje, cuando la interoperabilidad exige tal precisión. El contrato de servicios especifica las firmas de método actuales del servicio y se distribuye como una interfaz en uno de los lenguajes de programación compatibles, como Visual Basic o Visual C#.

Las directivas y enlaces estipulan las condiciones exigidas para comunicarse con un servicio. Por ejemplo, el enlace debe especificar (como mínimo) el transporte utilizado (por ejemplo, HTTP o TCP) y una codificación. Las directivas incluyen los requisitos de seguridad y otras condiciones que se deben cumplir para comunicarse con un servicio.

Tiempo de ejecución de servicio

La capa del tiempo de ejecución del servicio contiene los comportamientos que solo se producen durante la operación actual del servicio, es decir, los comportamientos en tiempo de ejecución del servicio. La limitación de peticiones controla cuántos mensajes se procesan que puede variar si la demanda para el servicio crece a un límite preestablecido. Un comportamiento de error especifica lo que sucede cuando se produce un error interno en el servicio, por ejemplo, controlando qué información se comunica al cliente. (Demasiada información puede dar ventaja a un usuario malintencionado para organizar un ataque.) El comportamiento de los metadatos rige cómo y si los metadatos se ponen a disposición del mundo externo. El comportamiento de la instancia especifica cuántas instancias del servicio se pueden ejecutar (por ejemplo, un **singleton** especifica solo una instancia para procesar todos los mensajes). El comportamiento de la transacción habilita la recuperación de operaciones de transacción si se produce un error. Comportamiento de la

expedición es el control de cómo se procesa un mensaje por la infraestructura de WCF.

La extensibilidad habilita la personalización de procesos en tiempo de ejecución. Por ejemplo, la inspección del mensaje es la facilidad para inspeccionar partes de un mensaje y la filtración de parámetros permite que se realicen acciones preestablecidas basándose en filtros que actúan en encabezados del mensaje.

Mensajería

La capa de mensajería se compone de canales. Un canal es un componente que procesa un mensaje de alguna manera, por ejemplo, autenticando un mensaje. Un conjunto de canales es también se denomina un pila de canales. Los canales funcionan en los mensajes y encabezados del mensaje. Esto es diferente de la capa en tiempo de ejecución del servicio, que se ocupa principalmente de procesar el contenido de los cuerpos de los mensajes.

Hay dos tipos de canales: canales de transporte y canales de protocolo.

Los canales de transporte leen y escriben mensajes de la red (o algún otro punto de la comunicación con el mundo externo). Algunos transportes utilizan un codificador para convertir los mensajes (que se representan como conjuntos de información XMLs) hacia y desde la representación de la secuencia de bytes utilizada por la red. Son ejemplos de transportes HTTP, canalizaciones con nombre, TCP y MSMQ. Son ejemplos de codificaciones XML y binario optimizado.

Los canales de protocolo implementan protocolos de procesamiento de mensajes, a menudo leyendo o escribiendo encabezados adicionales en el mensaje. Los ejemplos de tales protocolos incluyen WS-Security y WS-Reliability.

La capa de la mensajería muestra los posibles formatos y patrones de intercambio de los datos. WS-Security es una implementación de la especificación WS-Security que habilita la seguridad en la capa del mensaje. El canal de mensajería WS-Reliable habilita la garantía de entrega del mensaje.

Los codificadores presentan una variedad de codificaciones que se pueden utilizar para satisfacer las necesidades del mensaje. El canal HTTP especifica que el Protocolo de transporte de hipertexto se utiliza para la entrega del mensaje. El canal TCP especifica de manera similar el protocolo TCP. El canal de flujo de transacciones rige los patrones de mensajes de transacción. El canal de la canalización con nombre habilita la comunicación entre procesos. El canal de MSMQ habilita la interoperación con aplicaciones MSMQ.

Alojamiento y activación

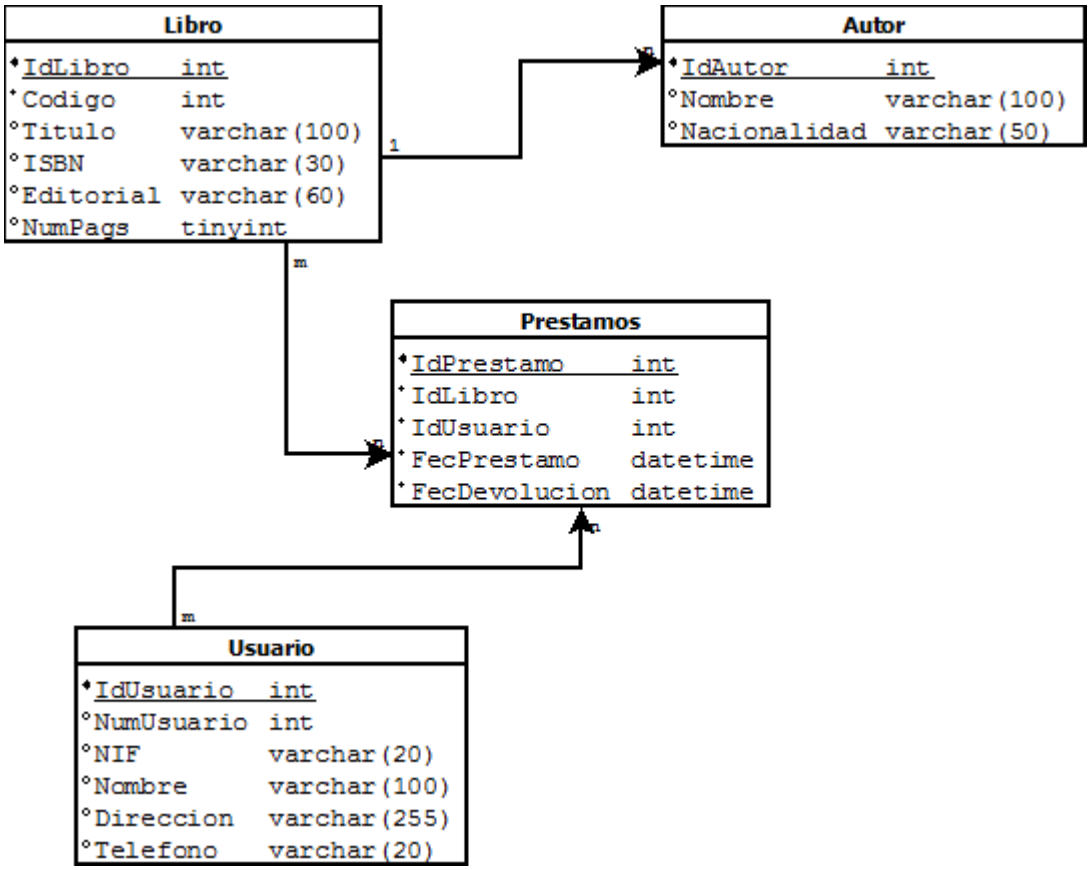
En su forma final, un servicio es un programa. Como otros programas, un servicio se debe ejecutar en un ejecutable. Esto se conoce como un auto hospedado service.

Los servicios también pueden ser hospedado, o ejecutar en un ejecutable administrado por un agente externo, como IIS o Windows Activation Service (WAS). Permite a las aplicaciones que se active automáticamente cuando se implementa en un equipo que ejecuta. Los servicios también se pueden ejecutar manualmente como ejecutables (archivos .exe). Un servicio también se puede ejecutar automáticamente como un servicio de Windows. Los componentes COM + también se pueden hospedar como servicios WCF.

Desarrollo de Laboratorio

1. PASO 1 (20 puntos)

Crear un Proyecto WCF Services Application y al igual que se realizó en el laboratorio 4 (punto 2,) se debe Mapear la siguiente base de datos relacional.



2. PASO 2 (20 puntos)

El paso 1, creara un servicio el cual contiene un Interfaz del Servicio y la implementación de esta interfaz, debe modificar la Interfaz e implementar los siguientes métodos:

- ObtenerTodoslosLibros
- ObtenrUnLibroPorCodigo
- CrearLibro
- ActualizarLibro

NOTA:

- SOLO SE DEBE HACER SELECT A LA TABLA LIBRO, NO HACER EL JOIN CON OTRAS TABLAS.
- RECORDAR QUE LOS METODOS DE DEFINEN EN LA INTERFAZ (IServices1.cs) E SE DESARROLLAN EN LA CLASE QUE IMPLEMENTA LA INTERFAZ (Services1.svc.cs)

3. **PASO 3 (20 puntos)**

Crear una nueva aplicación MVC (Laboratorio 3), que consulte el servicio web del PASO 2, y muestre , permita crear y editar libros.