Tema 2 – Introducción al modelo de Tecnologías Web y Middleware

Introducción

- Revisión de tecnologías Web
  - Cliente: DHTML, ActiveX, applet, plug-ins
  - Servidor: CGI, servlets, páginas activas, extensiones, servidor de aplicaciones
- Middleware
  - Arquitectura y modelo de servicios
  - Modelo de programación
  - Idoneidad del modelo para desarrollar aplicaciones distribuidas
- Servicios Web
  - Componentes básicos: SOAP, WSDL, UDDI
  - SOA y Servicios Web

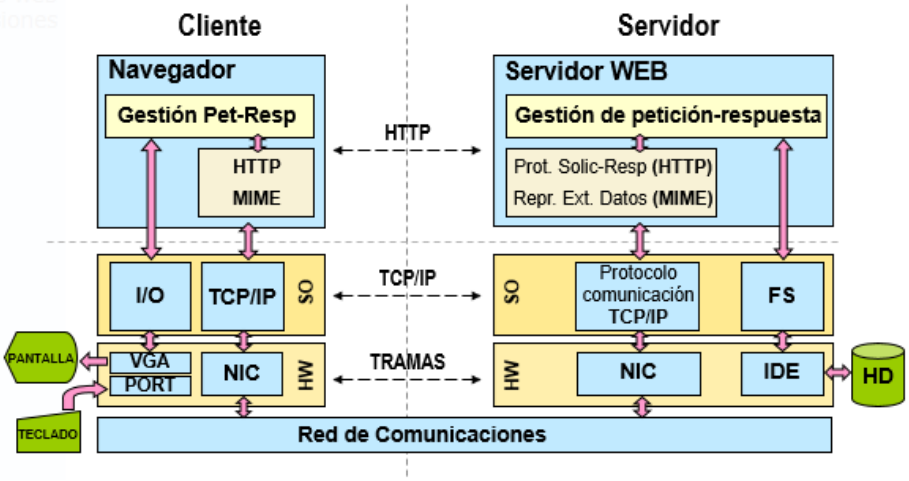
Tecnologías web

Modelos básicos

Modelo Web Básico

En el modelo web básico el cliente pide un recurso y recibe su respuesta, y hasta que no la reciba no puedo realizar otra petición (Petición/Respuesta).

1. Cliente solicita un recurso al servidor
   a. GET: Solicitar contenido
   b. HEAD: Solicitar únicamente la cabecera
   c. POST: Envía datos al servidor
   d. PUT: Envía datos para almacenar
2. Servidor web prepara el recurso HTML para enviarlo en un formato de texto estándar para cliente y servidor Texto (MIME)
   a. HTML (HiperText Markup Language)
      i. Documento de texto
      ii. Etiquetas de formato
      iii. Referencias cruzadas
   b. MIME (Multipurpose Internet Mail Extension)
3. El cliente recibe el recurso enviado por el servidor y lo interpreta para mostrarlo en el navegador.



Modelo CGI (Common Gateway Interface)

Saber sobre CGI:

- Es un método para la transmisión de información hacia un compilador instalado en el servidor
- Su **función principal** es la de añadir una mayor interacción a los documentos web que por medio del HTML se presenta de forma estática.
- El CGI se usa comúnmente para bases de datos, motores de búsqueda, formularios, etc.

Procedimiento de CGI

1. Cliente envía una petición GET a un Servidor Web para solicitar la ejecución de un recurso
2. La petición de ejecución es recibida por el Servidor Web y redirigida a la aplicación CGI
3. Esta aplicación analiza la petición y obtiene los valores necesarios para ejecutar la solicitud (parámetros del programa, puerto, programa que se ejecutará, etc.)
4. La aplicación CGI ejecuta la aplicación solicitada por el cliente con los datos que ha enviado y generar un HTML con la respuesta de la ejecución que se la envía al Servidor Web.
5. El Servidor Web envía el HTML generado por la aplicación CGI al cliente.
6. Cliente interpreta el HTML que ha recibido del Servidor Web

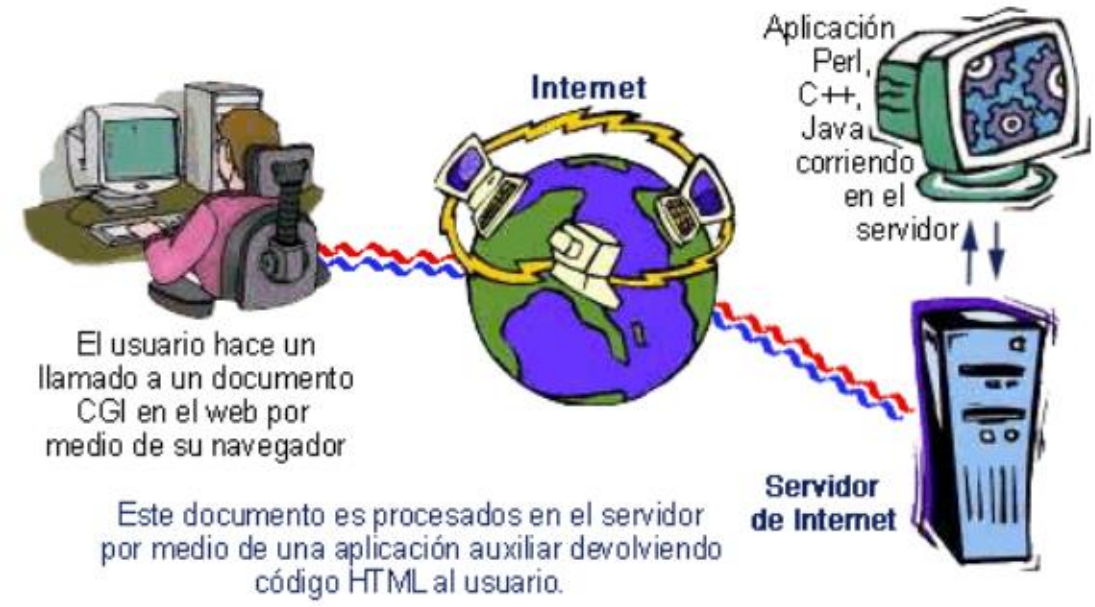
Ventajas CGI

- Ofrece capacidad de respuesta dinámica.
- Libertad elección lenguaje de programación.
- Corre en el servidor cuando el usuario lo solicita, por lo que es dependiente del servidor y no de la computadora del usuario.

Desventajas CGI

- No existe relación entre programa CGI y servidor Web. Por tanto, no existe control sobre la ejecución, resultado, etc.
- Nueva instanciación por cada solicitud. Por tanto, habrá sobrecarga de recursos.

Imagen del funcionamiento del CGI:



HTTP -> Es un protocolo de transporte sin estado. Los mecanismos de mantenimiento de estado (sesión) se deben implementar por parte del servidor o del cliente.

- Servidor:
  - Ficheros, BD's
  - Sistema centralizado. Provoca Sobrecarga
- Cliente:
  - Campos ocultos
  - Cookies

Ampliaciones

Las ampliaciones tienen mayor participación en el mundo de los negocios ya que tienen la necesidad de generar contenido dinámico.

- Ampliación del concepto Web como mero escaparate o catálogo estático
- Mejor gestión de recursos y mayor organización
- Aparición de numerosas tecnologías

Las ampliaciones en la aplicación Web se basada en arquitectura C/S con cliente un navegador y servidor como servidor web, utilizando protocolos de Internet (TCP/IP) para su entendimiento. Hay una introducción anárquica de nuevas tecnologías en la parte cliente y en la servidora.

Ampliaciones en el Cliente (DHTML, Applets de Java, ActiveX)

Ventajas

- Máxima fiabilidad, ya que, proporciona control total sobre el aspecto final y funcionalidad asegurada.

Desventajas

- Pérdida de compatibilidad, ya que tenemos la imposibilidad de obtener know-how debido a la plataforma o a versión no válida

Ampliaciones en el Servidor

Modelo Servlets

1. El cliente envía una petición al servidor web
2. El servidor Web la renvía al servlet
3. El servlet conecta con recursos propios u otros recursos para generar la respuesta HTML y la envía al servidor Web
4. El servidor Web renvía el código HTML generado por el servlet al cliente
5. El cliente interpreta la respuesta.

El modelo **Servlets se caracteriza por:**

- **Portabilidad**, ya que sólo precisa motor JVM.
- **Rendimiento**, solo necesita una única instanciación
- **Sesión**, ya que mantiene información entre diferentes conexiones al servlet
- **Software distribuido**, proporciona recuperación de disco/servidor de servlets y comunicación entre sí
- **Multithread**, permite múltiples peticiones concurrentes

Modelo de Extensiones: ISAPI, NSAPI

0. El servidor Web tiene extensiones que le permiten darle mayor funcionalidad y conectar con otros recursos externos.
1. El cliente envía una petición al servidor Web
2. El servidor Web procesa la solicitud haciendo uso si es necesario de las extensiones que tiene instaladas para completar la solicitud y generar la respuesta HTML.
3. El servidor Web envía la respuesta generada al cliente
4. El cliente interpreta la respuesta del servidor.

Ventajas

- Librerías de acceso dinámico, sólo se instancian una vez. Por tanto, se gana en rendimiento y aprovechamiento de los recursos del equipo.

Desventajas

- Tecnología cerrada a fabricante y servidor web (ISAPI/Microsoft/IIS o NSAPI/Netscape/NES)

Modelo de Páginas Activas (ASP, Liveware, JSP)

1. El cliente solicita un recurso Web al servidor Web
2. El servidor Web recibe la solicitud y recupera las páginas HTML que contiene Scripts en su interior (Páginas Activas)
3. Se ejecutan los scripts de dichas páginas. Estos scripts pueden conectarse con otros recursos externos para obtener la información necesaria y generar un código HTML
4. El código HTML generado por los scripts es enviado al cliente
5. El cliente interpreta la respuesta del servidor Web

Ventajas

- Habilita el concepto de sesión de usuario
- Buena gestión de recursos externos, como conexión a una BD
- Control sobre el protocolo petición/respuesta
- Operaciones ejecutadas en el servidor Web. Por tanto, independiza la aplicación Web del Navegador (cliente)

Desventajas

- Necesidad de conocer lenguaje script
  - Microsoft/ASP/VBScript
  - Netscape/Liveware/Javascript
  - Sun Microsystem/JSP/Java Script

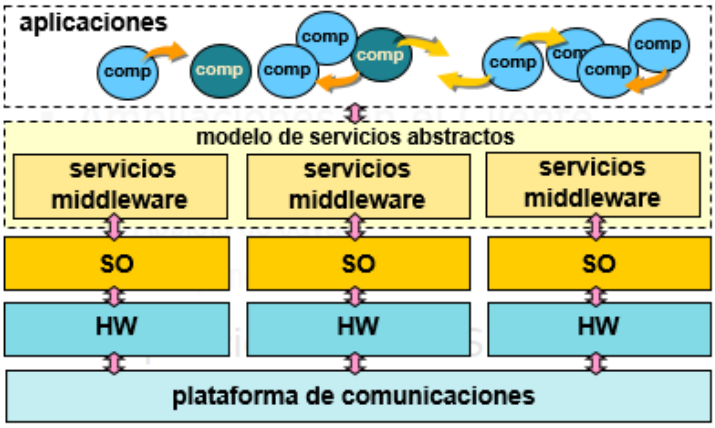
Modelo de Servidor de Aplicaciones: Middleware

El servidor de aplicaciones se separa del servidor Web. Por tanto, son independientes.

Cuando el cliente solicita un contenido estático, el servidor Web procesa la solicitud, genera la respuesta y se la envía al cliente. Si el cliente solicita un contenido dinámico, el servidor Web envía la petición al servidor de Aplicaciones.

El servidor de Aplicaciones recibe la respuesta y la procesa con su lógica de negocio y si es necesario conectado con otros recursos externos. Finalmente, genera un código HTML que reenvía al servidor Web. El servidor Web reenvía al cliente el código HTML generado por el servidor de Aplicaciones.

Un servidor Web puede está conectado a múltiples servidores de Aplicaciones esto genera una capa de servidores de Aplicaciones, un middleware entre el servidor Web y el servidor ejecuta la Aplicación, es decir, para el servidor Web es indiferente que máquina haya por debajo ejecutando la aplicación que ha solicitado él simplemente se conecta a un modelo de servicios abstractos que procesas su solicitud y le devuelve una respuesta.



Modelo Mixto

AJAX (Asynchronous Javascript And Xml) es una tecnología de desarrollo web que es el conjunto de otras tecnologías existentes como (HTML o XHTML, CSS, JavaScript, DOM, XML o JSON). Son aplicaciones ricas de la parte cliente, son de aspecto similar al de aplicaciones standalone, ya que, pretende evitar los efectos de refresco.

Se fundamenta en la ejecución en el cliente la parte de JavaScript y conexión asíncrona con el servidor (XML), es decir, comunicación en segundo plano y uso del objeto XMLHttpRequest

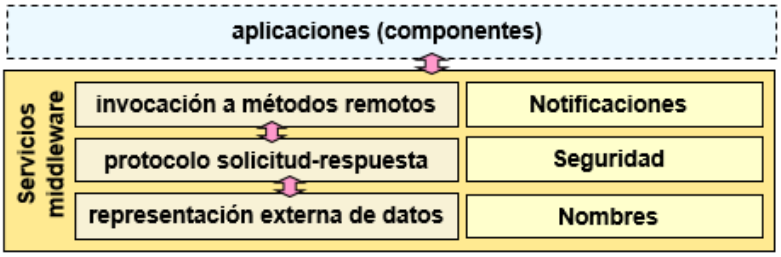
Middleware

Es un nivel de abstracción en términos de servicios cuya finalidad es proporcionar una visión única del sistema, independiente de la infraestructura que lo forme. Abstraer de la heterogeneidad y complejidad de las redes de comunicaciones, sistemas operativos y lenguajes de programación.

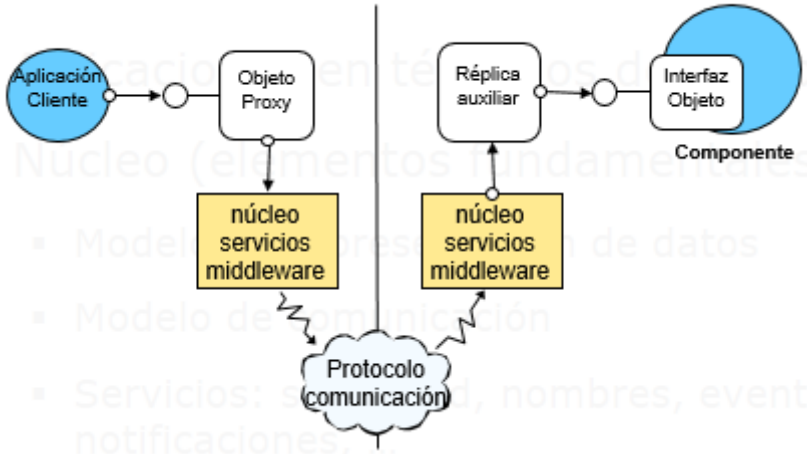
Punto de Vista de la arquitectura

La arquitectura de un middleware está compuesta por:

- Aplicaciones en términos de componentes
- Núcleo (elementos fundamentales)
    - Modelo de representación de datos
    - Modelo de comunicación
    - Servicios: seguridad, nombres, eventos y notificaciones, etc.

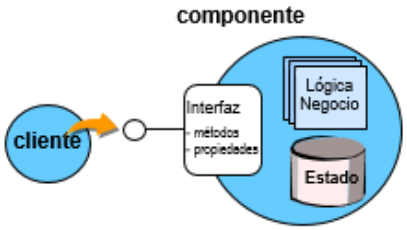


- Arquitectura de comunicación que proporciona
    - Interfaz
    - Proxy y réplica
    - núcleo



Middleware desde el punto de vista del programador

Desde el punto de vista de un programador el middleware hay que observar los componetes. La estructura de un componente está compuesta por:



- Interfaz: métodos+atributos
- Estado: estructura de datos, etc.
- Comportamiento: lógica de aplicación

El acceso a los servicios se realiza mediante la API de Servicios Middleware

Modelo de componentes

- Reutilización de código
- Transparencia con respecto a la plataforma sobre la que se ejecutan y con respecto al lenguaje de programación.
- Capacidad de personalización a través de las propiedades
- Comunicación transparente entre ellos y con el contexto mediante eventos
- Tipos de componentes
    - Cliente
    - Servidor: Encapsulado de servicios o Almacén de datos

J2EE

Es una plataforma Middleware basada en Java. Ofrece 3 tecnologías para el desarrollo de componentes:

Componentes Web

- Responden a una solicitud HTTP
- Servlets: programas Java que componen página web
- JSP: páginas web que contiene código Java

Componentes Enterprise JavaBean (EJB)

- Modelo de componentes distribuidos
- Unidades de software reusables que contienen lógica de empresa
- Tipos:
    - Beans de sesión
        - Ejecuta solicitudes del cliente y es destruido cuando las operaciones del cliente se completan
        - Con estado: matienen datos entre solicitud y solicitud
        - Sin estado: no mantienen estado entre solicitudes
    - Beans de entidad
        - Objeto persistente que modela los datos de un almacén
    - Beans dirigidos por mensaje
        - No son invocados por el cliente
        - Procesan mensajes asíncronos

APIs plataforma J2EE de SUN

- RMI: Proporciona un mecanismo de acceso a objetos remotos como si fueran locales. Con RMI-JRMP solamente se podían instanciar remotamente objetos java. Con RMI-IIOP se pueden instanciar objetos COBRA.
- JCA: Arquitectura que para interactuar con una variedad de EIS, incluye ERP, CRM y otra serie de sistemas heredados.
- JBDC: Acceso a base de datos relacionales
- JTA: Manejo y la coordinación de transacciones a través de EIS heterogéneos
- JNDI: Acceso a información en servicios de directorio y servicios de nombres
- JMS: Envío y recepción de mensajes
- JMail: Envío y recepción de correo
- JIDL: Mecanismo para interactuar con servicios COBRA
- Otras APIs: tratamiento de XML, integración con sistema heredados utilizando Servicios Web

J2EE establece 4 tipos diferente de contenedos:

- Contenedores Web
    - Servlets
    - JSP
- Contenedor EJB
    - Componentes Enterprise JavaBean
- Contenedor Applet
    - Applets: aplicaciones java ejecutadas en navegadores
- Contenedor de aplicaciones cliente
    - Aplicaciones Java estándar

Implementaciones

Plataforma .NET

- MS .NET Framework y .NET Remoting

    - Propietario de Microsoft
    - Es el único proveedor

Plataforma J2EE

- Java EE sdk
    - Aplicación de referencia
    - No eficaz`
- Oracle Weblogic
- Oracle application server
- Iplanet
- IBM Websphere
- Jboss (+ Tomcat), Geronimo, Glassfish

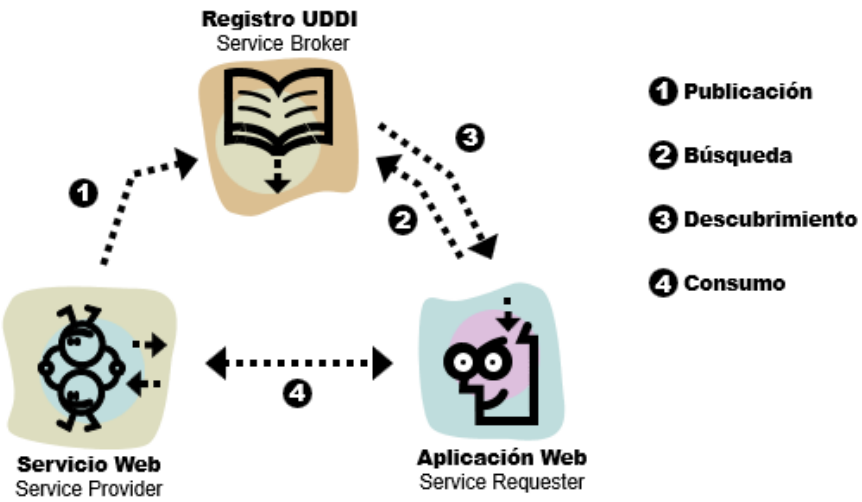
Servicios Web

Definición

Componentes que ejecutan procesos o funciones de negocio significativas, con una interfaz claramente definida y accesible a través de Internet, basada en el intercambio de documentos electrónicos en formato XML, y que pueden ser combinados entre sí.

- Es un paradigma para el desarrollo de sistemas distribuidos B2B.
- Proporciona sistema débilmente acoplados
- Reutilización y composición
- Interoperabilidad basada en contratos bien definidos
- Comunicación entre los negocios a nivel de aplicaciones:
    - Utilización de estándares abiertos: XML, SOAP, UDDI y WSDL

Arquitectura



Protocolos

- XML (eXtensible Markup Language): Solución para el intercambio de datos de una forma transparente.
- SOAP (Simple Object Access Protocol):
    - Plataforma para el intercambio de servicios sobre la red
    - Protocolo con el que hablan los servicios.
    - Permite a las aplicaciones invocar métodos de objetos remotos
    - Actúa sobre HTTP (en la práctica: sustituye a HTTP)
- UDDI (Universal Description, Discovery and Integration)
    - Permite descubrir con quién comunicarse y dónde
- WSDL (Web Service Descripton Language)
    - Describe los mensajes SOAP que definen un servicio Web en particular.
    - Utiliza un IDL (Interface Definition Language) de servicios
    - Los registros UDDI apuntan a una hoja WSDL

WSDL (Web Services Description Language)

Saber sobre WSDL:

- WSDL es un documento XML para describir Servicios Web.
- WSDL describe la interfaz pública a los servicios Web.
- Está basado en XML.
- 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.
- Se usa a menudo en combinación con SOAP y XML.
- Nos permite tener una descripción de un servicio web.
- Especifica la interfaz abstracta a través de la cal un cliente puede acceder al servicio y los detalles de cómo se debe utilizar.
- Descripción de localización del servicio (implementación concreta).

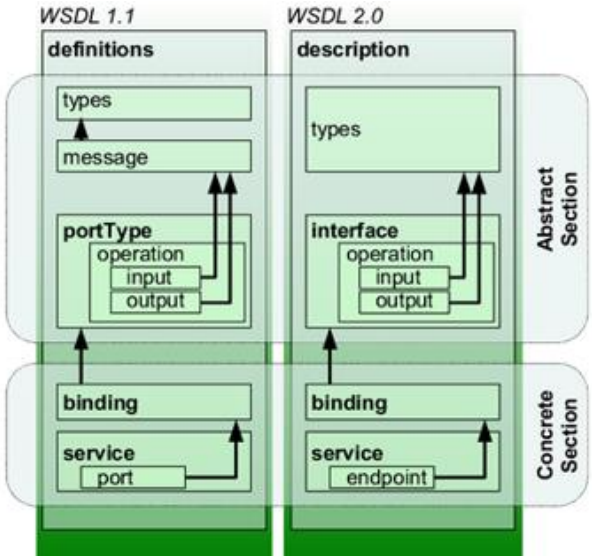
Anatomía de un documento WSDL:

- **<definitions>,** contiene la definición de uno o más servicios. Secciones concedptuales:
    - Types -> proveen definiciones de los tipos de datos utilizados para describir los mensajes intercambiados
    - Message -> representa una defínción abstracta de los datos que están siendo transmitidos. Un mensaje de divide en una serie de partes lógicas, cada una de las cuales se asocia con alguna definición de algún sistema de tipos.
    - PortType -> conjunto de operaciones abstractas. Cada operación hace referencia a un mensaje de entrada y uno de salida.
    - Binding -> especifica un protocolo concreto y las especificaciones del formato de los datos de las operaciones y los mensajes definidos por un porType en concreto.
    - Service -> para unir un conjunto de puertos relacionados.
        - Port -> especifica una dirección para un binding, para así definir un único nodo de comunicación
- **<message> y <portType>**, que operaciones provee el servicio
- **<binding>,** cómo se invocan las operaciones

- **<service>,** dónde se ubica el servicio
- **<documentation>,** puede contener información del servicio para el usuario

Herramientas para generar la descripción de servicio WSDL

- WSTK de IBM
- .NET Studio de Microsoft
- AXIS
- gSOAP Y cSOAP



UDDI (Universal Description, Discovery and Integration)

Saber:

- Es el catálogo de negocios de Internet.
- El registro en el catálogo se hace en XML.
- Su objetivo es ser accedido por los mensajes SOAP y dar paso a documentos WSDL.

Secciones conceptuales de un registro UDDI (el registro de un negocio en UDDI tiene tres partes):

- **Blanca (Páginas blancas)**: Similar a la información que aparece en el directorio telefónico, que incluye nombre, teléfono, y dirección
- **Amarilla (Páginas amarillas)**: Similar a su equivalente telefónico, e incluyen categorías de catalogación industrial tradicionales, ubicación geográfica, etc.
- **Verde (Páginas verdes)**: Información técnica acerca de los servicios ofrecidos por los negocios

Estructura central de un registro UDDI

- **businessEntity**: Información sobre un negocio o entidad. Utilizada por el negocio para publicar información descriptiva sobre si mismo y los servicios que ofrece.
- **businessService**: Servicios o procesos de negocios que provee la estructura businessEntity
- **bindingTemplate**: Datos importantes que describen las características técnicas de la implementación del servicio ofrecido
- **tModel**: Especificación y categorización técnica

Características de UDDI.

- **De autenticación**: Mecanismo que permite identificar el proveedor y consumidor de servicios.
- **De publicación**: Mecanismo para que los proveedores de servicios se registren (ellos mismos y sus servicios) en el Registro UDDI.
- **De consulta**: Permite a los suscriptores buscar los servicios disponibles y obtener el servicio una vez localizado.

SOAP (Simple Object Access Protocol)

Saber sobre SOAP:

- Define cómo dos objetos en diferentes procesos pueden comunicarse por medio del intercambio de datos XML.
- Es un protocolo ligero basado en XML para el intercambio de información en un entorno descentralizado y distribuido.
- Facilita la intercomunicación entre objetos de cualquier tipo, sobre cualquier plataforma y en cualquier lenguaje.
- Comunicación (débilmente acoplada) máquina-a-máquina.
- Una petición/respuesta de SOAP puede viajar sobre cualquier protocolo de aplicación: HTTP, SMTP, etc.

W3C en la especificación determina solo HTTP como oficial. Es un protocolo extensible desarrollado por el W3C. Además, permite el intercambio a través de firewall.

Se conforma de tres partes:

- **Sobre o envoltura**: define un marco de referencia general para expresar qué hay en el mensaje, quién debe de atenderlos, y si es opcional u obligatorio. Identifica un mensaje XML como SOAP.
    - Definición Wikipedia: define qué hay en el mensaje y cómo procesarlo.
- **Reglas de codificación**: definen un mecanismo de serialización que se puede utilizar para intercambiar instancias de tipos de datos definidos por la aplicación.
    - Definición Wikipedia: sirve para expresar instancias de tipos de datos.
- **Representación RPC/Document**: define una convención que puede ser utilizada para representar las llamadas y respuestas a procedimientos remotos.
    - Definición Wikipedia: representar llamadas a procedimientos y respuestas.

Características principales del protocolo SOAP:

- **Extensibilidad**: seguridad y WS-routing son extensiones aplicadas en el desarrollo.
- **Neutralidad**: bajo el protocolo TCP puede ser utilizado sobre cualquier protocolo de aplicación como HTTP, SMTP o JMS.
- **Independencia**: permite cualquier modelo de programación.

Estructura de mensaje SOAP

Es un documento XML ordinario con la siguiente estructura:

- **Envoltura (obligatoria)**: raíz de la estructura. Identifica al mensaje SOAP como tal.
- **Header (Cabecera)**: permite enviar información relativa a cómo debe ser procesado el mensaje.
    - **Header Blocks**: delimitan las unidades de información necesarias para el header.
- **Body (cuerpo) (obligatoria)**: contiene la información relativa a la llamada y la respuesta.
- **Fault (errores)**: contiene la información relativa a los errores que se hayan producido durante el procesado del mensaje (envío y recepción del mismo).

Conclusiones

- Gran diversidad de tecnologías alrededor de la Web
- Idoneidad del modelo de componentes y plataforma middleware para el desarrollo de grandes aplicaciones distribuidas sobre Internet
- Idoneidad del modelo web para gestionar la capa de presentación (B2C)
- Necesidad de conocer tecnologías y herramientas para seleccionar adecuadamente
- Idoneidad del modelo de servicios Web para gestionar la interacción B2B