1-DEFINA EN FORMA APROXIMADA QUE ES LO QUE ENTIENDE POR SISTEMA DISTRIBUIDO
Colección de computadoras independientes que le dan al usuario la idea de constituir un único sistema coherente.
2- ¿COMO TIENEN QUE SER LOS SISTEMAS DISTRIBUIDOS, (DOS CUALIDADES)?
Los sistemas distribuidos deben ser abierto y debe ser escalable.
3-A LOS SIST DISTRIBUIDOS, ¿COMO TAMBIEN SE LOS CONOCE?
Los sistemas distribuidos son conocidos también como Middleware.
4- ¿COMO TIENEN QUE PERMANECER EL HARDWARE Y EL SOFTWARE?
Las diferencias entre los distintos tipos de hardware y sistemas operativos deben permanecer ocultas.

5-EXPLIQUE MIGRACION, REPLICACION, REUBICACION Y CONCURRENCIA
Migración: que un recurso pudiera moverse a otra ubicación
Replicación: del número de copias de un recurso
Reubicación: que un recurso pudiera moverse a otra ubicación mientras está en uso
Concurrencia: que un recurso pueda ser compartido por varios usuarios que compiten por él.

6-EL GRADO DE APERTURA, ¿QUE CARACTERISTICAS TIENE?

Grado de apertura: grado en el que se pueden añadir nuevos servicios a la utilización de recursos sin perjudicar a los existentes.
- Interoperabilidad: las implementaciones de sistemas y componentes de fabricantes pueden coexistir y trabajar juntos.
- **Extensible**: debe ser fácil de configurar para componentes diferentes, aparte de agregar nuevos o reemplazar los existentes sin afectar a los que permanecen en su lugar
- **Flexibilidad**: los componentes son relativamente pequeños y fáciles de reemplazar o adaptar

7-y RESPECTO DE LA ESCALABILIDAD?

Escalabilidad: capacidad de adaptación y respuesta de un sistema con respecto al rendimiento a medida que aumenta de forma significativa el número de usuarios de este.
Se puede medir desde varios puntos de vista:
-Respecto a su tamaño: podemos agregar fácilmente recursos y usuarios. Hay que evitar los algoritmos centralizados.
-Desde el punto de vista geográfico: usuario y recursos pueden radicar muy lejos uno de los otros
-Escalabilidad administrativa: puede ser fácil de manejar incluso involucrando muchas organizaciones administrativas diferentes.

8-LAS TECNICAS DE ESCALAMIENTO PARA QUE ME SIRVEN?
Las técnicas de escalamiento sirven para disminuir el impacto de los problemas de escalar un sistema, las cuales son:
-Ocultar las latencias de comunicación
-Distribución: dividir un componente en partes más pequeñas y distribuir dichas partes a lo largo del sistema
-Replicación: replicar los componentes a lo largo del sistema distribuido

9- ¿QUE CARACTERISTICAS TENDRIA UN SISTEMA DISTRIBUIDO IDEAL?
Un sistema distribuido ideal debe tener una red confiable, segura y homogénea por lo que la topología no debe cambiar, su latencia debería ser cero, el ancho de banda infinito, el costo de transporte igual a cero, que exista un administrador, etc.

10- ¿QUE COSAS SE TIENEN EN CUENTA EN TODOS LOS TIPOS DE SISTEMAS DISTRIBUIDOS?
Se tienen en cuenta la arquitectura, procesos, comunicación, asignación de nombres, sincronización, consistencia y replicación, tolerancia a fallas, seguridad

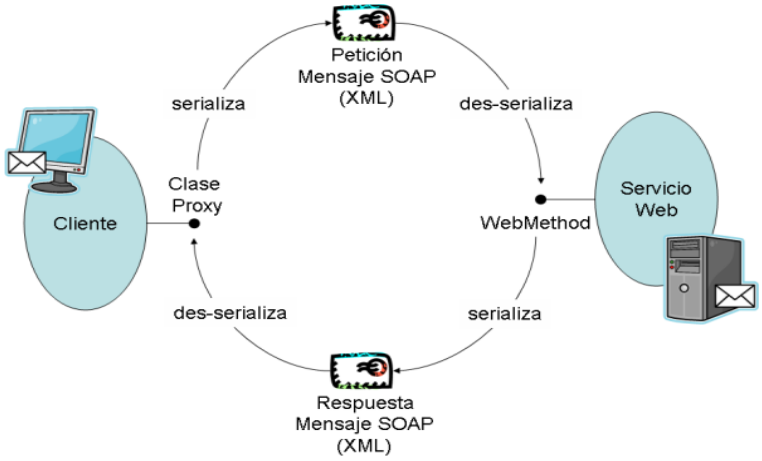
11- ¿COMO FUNCIONAN LOS SISTEMAS DISTRIBUIDOS BASADOS EN LA COORDINACION?
Los mensajes son direccionados por tema, por lo que no tienen la dirección de su receptor. Los procesos que desean recibir mensajes deben suscribirse a un tema específico. El middleware se encarga de que los mensajes se encaucen desde los editores hasta los suscriptores.

12-EXPLIQUE SIST DIDTRIBUIDOS DE COMPUTO
En cluster: El HW consta de una colección de estaciones de trabajo similares o PC, conectadas cercanamente por medio de una LAN de alta velocidad. Cada nodo ejecuta el mismo Sistema Operativo.
En malla (grid): consta de sistemas distribuidos construidos generalmente como un conjunto de sistemas de cómputo en donde cada sistema podría caer dentro de un dominio administrativo distinto, y obviamente distintas sus características de HW, SW y de tecnología de red.

13- ¿A QUE SE CONOCE COMO RPC TRANSACCIONAL?
RPC: llamada a procedimiento remoto, el cual es un programa que utiliza una computadora para ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambas, de forma que parezca que se ejecuta local. Los clientes usan procesos que se ejecutan en CPUs remotas, pueden ser síncronas o asíncronas.

14- ¿CUALES SON LOS DESAFIOS DE LOS SISTEMAS DISTRIBUIDOS?
La heterogeneidad de redes, HW de computadores, sistemas operativos, lenguajes de programación e implementación de distintos desarrolladores.

15- ¿COMO GRAFICARIA UN SERVICIO WEB?



16-DESCRIBA WSDL, UDDI Y SOAP

WSDL (Web Service Description Language): lenguaje formal similar a los lenguajes de interfaz usados para soportar la comunicación basada en RPC. Puede ser transformado en forma automática en Resguardo del lado del cliente y del lado del servidor.

SOAP (Simple Object Access Protocol): estructura donde una gran parte de la comunicación entre dos procesos puede ser estandarizada.

UDDI (Universal Description Discovery): define un modo de publicar y encontrar información sobre servicios WEB. Protocolo basado en SOAP que define cómo se comunican los clientes con los registros UDDI. Incluye un esquema XML para mensajes SOAP que define un conjunto de documentos para describir información de empresas y servicios.

17- ¿QUE ASPECTOS SE EVALUAN EN LOS WEB SERVICES? DETALLELOS

Sincronización: es necesario que los servidores se sincronicen entre sí y que sus acciones sean coordinadas.

Consistencia y Replicación: se idean estrategias para guardar en caché y replicar contenido tanto estático como dinámico, es decir, documentos generados a consecuencia de una solicitud. El almacenamiento en caché del lado del cliente ocurre generalmente en 2 lugares:

-Los browsers cuentan con un almacenamiento en la caché que se pueda configurar indicando en qué momento se deberá verificar la consistencia

-El sitio de un cliente ejecuta con frecuencia un proxy web. Un proxy web acepta solicitudes de clientes locales y las transfiere a servidores web. Cuando llega una respuesta, el resultado se transfiere al cliente. La replicación sirve para que el sitio sea continuo y de fácil acceso.

Algunas métricas:

-Métrica de latencia: mide el tiempo requerido para realizar una acción

-Métrica de uso de red: se relaciona con el ancho de banda consumido en función del número de bytes que se van a transferir

-Métrica espacial: mide la distancia entre nodos en función del número de conexiones intermedias de enrutamiento a nivel de red o de conexiones intermedias entre sistemas autónomos

-Métricas de consistencia: informa en qué medida o grado se está desviando una réplica de su copia

maestra.
-Métrica financiera: forma otra clase para medir que tan bien está funcionando una CDN. Está estrechamente relacionada con la infraestructura real de Internet.
-Tolerancia a fallas: se logra principalmente por medio de almacenamiento en caché del lado del cliente y replicación de servidores. Los problemas se agravan para servicios diseñados para manejar fallas bizantinas, un servicio que tolere este tipo de fallas (BTS) puede necesitar actuar como cliente de otro servicio no replicado.
- Seguridad: el método para establecer un canal seguro en la web es usar una capa de socket seguro (SSL), que fue implementada originalmente por Netscape, hoy es conocido como protocolo de seguridad en la capa de transporte (TLS)

18- ¿QUE PROPONEN CASTRO Y LISKOV?
Una solución al sistema BFT debe manejar:
-Los clientes de un servicio BTF deberán ver al servicio como a cualquier otro servicio web
-Deberá garantizar la consistencia interna cuando actúa como cliente
-Deberán tratarse como un servicio BFT que actúa como cliente y como una sola entidad.

19-UN CANAL SEGURO SE ESTABLECE EN DOS FASES. EXPLIQUELO
El establecimiento de un canal seguro se realiza en dos fases:
-El cliente informa al servidor acerca de algoritmos criptográficos que puede manejar, así como también, cualquier método de compresión que soporte.
-La segunda fase es la autenticación: siempre se requiere que el servidor se autentique a sí mismo, por lo que el cliente pasa un certificado que contiene una clave pública firmada por una autoridad de certificación. Si el servidor requiere que el cliente se autentifique, también el cliente deberá enviar un certificado al servidor.

20- ¿CUALES SON LOS MODELOS ARQUITECTONICOS?
Estilo arquitectónico se formula en términos de componentes, el intercambio entre componentes y cómo estos elementos se pueden configurar juntos en un sistema. Los estilos arquitectónicos que tenemos son en capas, basadas en objetos, centradas en datos y basadas en eventos.

21-**DEFINA LAS INVOCACIONES PARA RPC, RMI Y CORBA Y COMPARELOS EN TODOS SUS ASPECTOS MUY IMPORTANTE**
RPC: lógica de negocios escrita como procedimientos que se invoca en forma remota
RMI: lógica de negocios escrita como objetos cuyos métodos son invocados en forma remota. Define la forma de comunicación remota entre objetos Java situados en máquinas virtuales distintas.
CORBA: lógica de negocios escrita como objetos que son invocados en forma remota.
Comparación:
-RMI es más lento que RPC por la interpretación byte-code en la JVM, pero se puede mejorar con el compilador JIT
-RMI tiene una ventaja adicional sobre RPC porque pertenece al mundo orientado a objetos de Java. Los sistemas RPC tradicionales son lenguajes neutrales, de modo que no pueden proveer la funcionalidad que no estén disponibles en sus plataformas destino. RMI por su naturaleza Java se conecta a los sistemas usando métodos nativos, por lo que puede proveer tecnología distribuida en forma directa, natural y potente.

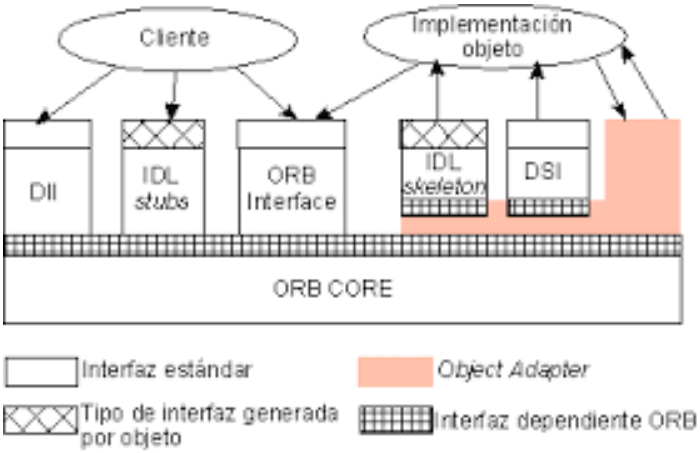
22- ¿COMO SE HACE PARA CONSTRUIR UN WEB SERVICE Y COMO SE CONSTRUYE UNA APLICACIÓN EN CORBA? EXPLIQUE LAS DIFERENCIAS.

Construcción de un Web Service:
Archivo--Nuevo--Proyecto--Sitio web vacío de ASP.NET--En la solución se agrega un elemento--Servicio Web asmx--Para invocarlo tenemos que agregar referencia de servicio-- Servicio Web de esta solución.

Construcción de una aplicación CORBA:

Definir la interfaz del objeto u objetos utilizando IDL →Compilar el fichero IDL. Esto produce el código stub y skeleton que proporciona transparencia en la localización. →Esto significa que cooperan con la librería ORB para convertir una referencia a un objeto en una conexión por red a un servidor remoto y codifican los argumentos proporcionados en una operación sobre la referencia del objeto, la transportan al método correcto en el objeto denotado por dicha referencia, ejecutan el método y devuelven los resultados→Identificar las interfaces y clases generadas por el compilador IDL que necesitaremos usar o especializar para invocar o implementar operaciones→Escribir código para inicializar el ORB e informarle de cualquier objeto CORBA que hayamos creado→Compilar todo el código generado junto con el código de nuestra aplicación con un compilador Java→Ejecutar la aplicación distribuida.

CORBA: arquitectura orientada a objetos, los objetos poseen muchas características como la herencia, interfaces y polimorfismo, pero proporciona estas características a lenguajes que no son orientados a objetos como C o COBOL, pero trabaja mas eficiente con lenguajes orientados a objetos como Java o C++. Provee una infraestructura que permite la comunicación de objetos independiente de plataforma y de implementación.



- **ORB (Object Request Broker):** facilita la comunicación entre objetos. Se encarga de enviar las peticiones a los objetos y retornar las respuestas a los clientes que las invocan por el proceso de serialización.
- IDL: lenguaje que se utiliza para definir las interfaces entre los componentes de una aplicación y es el elemento que soporta la interoperabilidad de la tecnología.
    - **Stub IDL:** interfaz estática a los servicios declarados en las interfaces IDL, para el cliente todas las llamadas parecen locales.
    - **Skeleton IDL:** es el representante estático del cliente en el servidor, para el servidor todas las llamadas parecen locales.

Sistemas distribuidos basados en coordinación: tiene como objetivo la cooperación y la comunicación entre componentes (servicios, objetos, módulos). Su funcionamiento posee una estructura para realizar la coordinación:

- **Desacoplamiento referencial:** se da cuando no se requiere que los componentes que intervienen en una determinada tarea o actividad estén en funcionamiento
- **Desacoplamiento temporal:** se da cuando se requiere que los componentes que intervienen en una determinada tarea o actividad estén en funcionamiento.

1-DEFINA SEVICIO WEB

Tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones (las aplicaciones pueden estar en lenguajes de programación diferentes y sobre cualquier plataforma)

2-QUE ES SOA? ¿QUE ES LO QUE PERMITE?

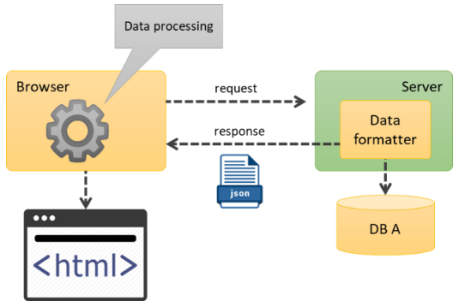
SOA (Service Oriented Architecture) es un tipo de arquitectura de software que se basa en la integración de aplicaciones mediante servicios. Permite la creación de una arquitectura que se integra (o forma parte) de sistemas de información y/o aplicaciones que puedan llegar a ser bastante escalables, que permitan modelar todo el proceso de negocios de una organización además es capaz de invocar y exponer servicios web.

3-QUE SON SOAP Y REST RELACIONADOS CON SOA?

Son servicios que representan dos tipos muy diferentes de una implementación de arquitectura SOA.

4-DEFINA REST

Patrón de arquitectura construido bajo verbos simples que se adaptan fácilmente con HTTP



5-DEFINA SOAP

Protocolo de mensajería basado en XML que puede utilizarse por una variedad de protocolos de transporte (como HTTP)

6-EXPLIQUE LA FORMA EN QUE SE MANIFIESTAN LOS DIFERENTES ESTADOS DURANTE EL CICLO DE VIDA PARA SOAP Y PARA REST

En SOAP el movimiento por los diferentes estados se realiza interactuando con un extremo único del servicio que puede encapsular y proporcionar acceso a muchas operaciones y tipos de mensaje. En cambio, REST se permite un número limitado de operaciones y dichas operaciones se aplican a recursos representados y direccionados por direcciones HTTP, los mensajes capturan el estado actual o requerido del recurso (lo que hace que REST funcione mejor en aplicaciones WEB donde se puede hacer uso de HTTP para tipos de datos que no son XML).

7-QUE ES LO QUE SOAP LE PERMITE A LOS DISEÑADORES?

Les permite encapsular la complejidad del sistema, dando lugar a interfaces generadas automáticamente que permiten facilitar el diseño del sistema.

8-DEFINA SOAP COMO PROTOCOLO

Protocolo estándar que define como dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML

9- ¿COMO ES SOAP RELACIONADO CON LA ROBUSTEZ?

- Es bastante robusto, debido a que tiene un tipado más fuerte que REST, además de que permite agregar los atributos y permite definir espacios de nombre, evitando la ambigüedad.

10- ¿QUE TIPO DE TECNOLOGÍA ES REST? ¿QUE TIPO DE DATOS TRANSMITE?

- REST es un web service para sistemas distribuídos que utiliza el protocolo HTTP para transportar datos.
- Permite transmitir prácticamente cualquier tipo de dato debido a que esta definido en el header content type. Algunos de estos tipos de datos pueden ser:
    - XML
    - JSON (más utilizado por su velocidad y tamaño)
    - Binarios (imágenes y documentos)
    - Texto

11- ¿CON QUE MODELO ESTA RELACIONADO REST?

- Con el modelo MVC (model, view, controller)

12- CUAL DE LOS DOS ESTA FUERTEMENTE ACOPLADO

- SOAP: los componentes están fuertemente acoplados

13- NOMBRE UNA CARACTERÍSTICA SALIENTE ENTRE REST Y SOAP

- REST: Componentes débilmente acoplados.
- SOAP: Componentes fuertemente acoplados.

14- NOMBRE UNA VENTAJA PARA REST Y SOAP

- REST: Fácil de construir y adoptar.
- SOAP: Generalmente fácil de utilizar.

15- NOMBRE UNA DESVENTAJA PARA REST Y SOAP

- REST: Manejar el espacio de nombres puede ser engorroso.
- SOAP: Los clientes necesitan saber las operaciones y su semántica antes del uso.

16- NOMBRE UNA CARACTERISTICA TECNOLOGICA PARA REST Y SOAP

- REST: Pocas operaciones con muchos recursos
- SOAP: Muchas operaciones con pocos recursos

17- NOMBRE LOS PROTOCOLOS INVOLUCRADOS PARA REST Y SOAP

- **REST:**
  - HTTP:
    - GET
    - POST
    - PUT
    - DELETE
  - XML auto descriptivo
  - Sincrono
- **SOAP:**
  - SMTP
  - HTTP POST
  - MQ
  - Tipado fuerte, XML schema
  - Sincrono y asíncrono

18- RESPECTO A LA SEGURIDAD ¿QUÉ PUEDE ARGUMENTAR ENTRE REST Y SOAP?

Seguridad:

- **Rest:**
  - https.
  - comunicación punto a punto segura.
- **SOAP:**
  - WS SECURITY.
  - Comunicación origen a destino segura.

19-EXPLIQUE LAS PAGINAS BLANCAS, AMARILLAS Y VERDES, ¿Y CON QUE ESTAN REALACIONADAS?

Todas esas páginas son un tipo de registro del UDDI, (Universal Description, Discovery and Integration que es un registro en el catálogo que se hace en XML.

El registro de un negocio en UDDI tiene tres partes:

Páginas blancas - dirección, contacto y otros identificadores conocidos.

Páginas amarillas - categorización industrial basada en taxonomías.

Páginas verdes - información técnica sobre los servicios que aportan las propias empresas.

20- ¿COMÓ ES EL STACK DE WEB SERVICES?

Web Services Protocol Stack

La Pila de protocolos para Servicios Web es una colección de protocolos y estándares para redes de Computadores que son utilizados para definir, localizar, implementar y hacer que un Servicio Web interactúe con otro. La Pila de Protocolos para servicios está comprendida principalmente por cuatro áreas:

- **Servicio de Transporte:** responsable del transporte de mensajes entre las Aplicaciones de red y los protocolos en los cuales se incluyen protocolos tales como HTTP, SMTP, FTP, así como también el más reciente Blocks Extensible Exchange Protocol (BEEP).
- **Mensajería XML:** responsable por la codificación de mensajes en un formato común XML así que ellos puedan ser entendidos en cualquier extremo de una conexión de red. Actualmente, esta área incluye protocolos tales como XML-RPC, SOAP y REST.
- **Descripción del Servicio**: usado para describir la interfaz pública de un Servicio Web específico. El formato de interfaz Web Services Description Language - WSDL es típicamente usado para este propósito.
- **Descubrimiento de servicios**: centraliza servicios en un registro común tal que los servicios Web de la red puedan publicar su localización y descripción, y hace que sea fácil descubrir que servicios están disponibles en la red. Actualmente, la API Universal Description Discovery and Integration - UDDI se utiliza normalmente para el descubrimiento de servicios.

21- ¿A TRAVES DE QUE CONCEPTO LA UDDI ESTABLECE UNA DISTICINCION ENTRE LA ABSTRACCION Y LA IMPLEMENTACION?

UDDI establece una distinción similar entre la abstracción y la implementación con el concepto de tModels. La estructura tModel, abreviatura de "Technology Model" (modelo de tecnología), representa huellas digitales técnicas, interfaces y tipos abstractos de metadatos. El resultado de los tModels son las plantillas de enlace, que son la implementación concreta de uno o más tModels. Dentro de una plantilla de enlace se registra el punto de acceso de una implementación particular de un tModel. Del mismo modo que el esquema de WSDL permite separar la interfaz y la implementación, UDDI ofrece un mecanismo que permite publicar por separado los tModels de las plantillas de enlace que hacen referencia a ellos. Por ejemplo, un grupo industrial o de estándares publica la interfaz canónica para un sector particular y, a continuación, varias empresas escriben implementaciones de esta interfaz.

22- ¿COMÓ ES EL PROCESO DE PUBLICACION DE LA UDDI (A GRANDES RASGOS)?

Cuando publique un servicio web en un registro UDDI, deberá realizar estas acciones:

- Especificar el tipo de negocio al que da soporte su servicio web. Normalmente, esto significa elegir un tipo de negocio existente en una lista, pero también se puede crear un nuevo tipo de negocio. Para cada tipo de negocio hay una clave de empresa asociada. Los servicios web habilitados para bus de integración de servicios utilizan esta clave, en combinación con la clave de servicio, para encontrar el servicio web en el registro.
- Añadir un modelo técnico. Los modelos técnicos son categorías genéricas. Mediante estos modelos un usuario del registro UDDI puede buscar un tipo de servicio, en lugar de tener que conocer los detalles de acceso de un servicio específico. Los servicios web habilitados para bus interactúan con registros UDDI en el nivel de servicios Web individuales y, por lo tanto, no utilizan modelos técnicos.
- Añadir el servicio web. El registro UDDI asigna una clave de servicio a su servicio, y lo publica. Los servicios web habilitados para bus utilizan esta clave, en combinación con la clave de empresa, para encontrar el servicio web en el registro.

23- ¿COMÓ HACE WDSL PARA DISTINGUIR A LOS PUERTOS, DE LOS MENSAJES?

WDSL diferencia a los puertos de los mensajes debido a que un documento WDSL se compone de un elemento raíz el cual a su vez se compone de diferentes elementos, entre los cuales existen dos elementos diferentes para ellos, uno para puertos y otro para los mensajes.

24- ¿WSDL COIMO DEFINE A LOS SERVICIOS? La respuesta es extensa, conteste a detalle.

Los documentos WSDL definen los servicios como colecciones de puntos finales de red o puertos. En WSDL, la definición abstracta de puntos finales y de mensajes se separa de la instalación concreta de red o de los enlaces del formato de datos. Esto permite la reutilización de definiciones abstractas: mensajes, que son descripciones abstractas de los datos que se están intercambiando y tipos de puertos, que son colecciones abstractas de operaciones. Las especificaciones concretas del protocolo y del formato de datos para un tipo de puerto determinado constituyen un enlace reutilizable. Un puerto se define por la asociación de una dirección de red y un enlace reutilizable; una colección de puertos define un servicio. Por esta razón, un documento WSDL utiliza los siguientes elementos en la definición de servicios de red:

- **Types**: contenedor de definiciones del tipo de datos que utiliza algún sistema de tipos (por ejemplo, XSD).
- **Message**: definición abstracta y escrita de los datos que se están comunicando.
- **Operation**: descripción abstracta de una acción admitida por el servicio.
- **Port Type:** conjunto abstracto de operaciones admitidas por uno o más puntos finales.
- **Binding**: especificación del protocolo y del formato de datos para un tipo de puerto determinado.
- **Port**: punto final único que se define como la combinación de un enlace y una dirección de red.
- **Service**: colección de puntos finales relacionados.