SISTEMAS DISTRIBUIDOS CUESTIONARIO 1

Sistema distribuido: Colección de computadoras independientes que le dan al usuario la idea de constituir un único sistema coherente.

Como deben ser: ser abierto y debe ser escalable.

Como también se los conoce: como Middleware.

Como deben permanecer el hardware y software: diferencias entre los distintos tipos de hardware y sistemas operativos deben permanecer ocultas.

Migración, replicación, reubicación 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.

Grado de apertura:

- **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

Escalabilidad: 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.

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

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.

Que se tiene en cuenta en todos los sistemas distribuidos: (debe tener en cuenta)

- Arquitectura
- Procesos
- Comunicación
- Asignación de nombres
- Sincronización
- Consistencia
- Replicación
- Tolerancia a fallas
- Seguridad

Sistemas distribuidos basados en la coordinación:

- 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.

Sistemas distribuidos de cómputo:

- **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.

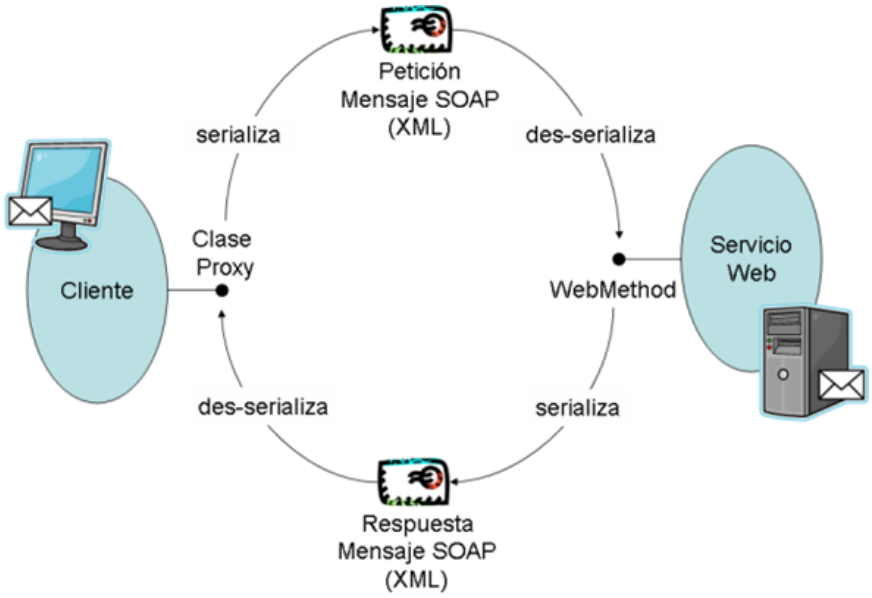
A que se conoce como RPC transaccional: es la llamada a un procedimiento remoto, el cual es un programa que utiliza una computadora para ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambas, de forma que parezca que se ejecuta local. Los clientes usan procesos que se ejecutan en CPUs remotas, pueden ser síncronas o asíncronas.

Desafíos sistemas distribuidos:

- heterogeneidad de redes
- HW de computadores
- sistemas operativos
- lenguajes de programación e implementación de distintos desarrolladores.

Gráfico servicio web:

(graficaría)



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.
- **UDDI (Universal Description, Discovery, and Integration):** define un modo de publicar y encontrar información sobre servicios Web.
    - **UDDI tiene dos funciones:**
        - Es un protocolo basado en SOAP que define cómo se comunican los clientes con los registros UDDI.
        - Es un conjunto de registros duplicados globales en particular.
    - **Registro UDDI:**
        - **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.
- **SOAP (Simple Object Access Protocol):** estructura donde una gran parte de la comunicación entre dos procesos puede ser estandarizada.

Que aspectos se evalúan en los webs services:

- **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)

Que propone 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.

Canal seguro se establece en 2 (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.

Modelos arquitectónicos: Estilo arquitectónico se formula en términos de componentes, el intercambio entre componentes y cómo estos elementos se pueden configurar juntos en un sistema.

- **Cuales son:**
  - en capas
  - basadas en objetos
  - centradas en datos
  - basadas en eventos.

Invocaciones RPC – RMI – CORBA y compárelos:

- **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
- **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.

Construir 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.

Construir 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.

IDL: lenguaje declarativo para la definición de interfaces de objetos CORBA. Es una forma independiente del lenguaje en la que los programadores y los usuarios de los objetos pueden estar seguros de que se invoca a la operación correcta del objeto correcto, aunque la única información adicional que se necesita es la referencia al objeto.

SERVICIO WEB CUESTIONARIO 2

Servicio 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).

SOA: SOA (Service Oriented Architecture) es un tipo de arquitectura de software que se basa en la integración de aplicaciones mediante servicios.

Que permite SOA: 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.

Que son SOAP y REST relacionado con SOA: Son servicios que representan dos tipos muy diferentes de una implementación de arquitectura SOA.

REST: (Representational state transfer) Patrón de arquitectura construido bajo verbos simples que se adaptan fácilmente con HTTP

SOAP: (simple object access protocol) Protocolo de mensajería basado en XML que puede utilizarse por una variedad de protocolos de transporte (como HTTP)

Como se manifiestan los diferentes estados durante el ciclo de vida para SOAP y REST:

- **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.
- **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).

SOAP permite a los diseñadores: encapsular la complejidad del sistema, dando lugar a interfaces generadas automáticamente que permiten facilitar el diseño del sistema.

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

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.

Tipo de tecnología es rest: REST es un web service para sistemas distribuidos que utiliza el protocolo HTTP para transportar datos.

Tipos de datos transmite rest: Permite transmitir prácticamente cualquier tipo de dato debido a que está definido en el header content type.

- XML
- JSON (más utilizado por su velocidad y tamaño)
- Binarios (imágenes y documentos)
- Texto

Con que modelo está relacionado REST: con el modelo MVC.

Cual está fuertemente acoplado: SOAP.

Característica saliente de REST y SOAP:

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

Ventaja para REST y SOAP:

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

Característica tecnológica para REST y SOAP

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

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
    - Síncrono y asíncrono

Respecto a seguridad REST y SOAP:

- **REST:**
    - HTTPS.
    - Comunicación punto a punto y segura.
- **SOAP**
    - WS security.
    - Comunicación origen y destino segura.

Paginas blancas, amarillas y verdes; y con que están relacionadas:

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.

Stack de web services:

Web Services Protocol Stack: 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 UDDI se utiliza normalmente para el descubrimiento de servicios.

A través de que concepto la UDDI establece una distinción entre la abstracción y la implementación: 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.

Como es el proceso de publicación de la UDDI:

- Primero es 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.

WDSL distinguir a los puertos, de los mensajes: 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.

WSDL como define a los servicios: 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.