## **Cuestionario de Sistemas Distribuidos**

## **1- Defina en forma aproximada qué es lo que entiende por sistema distribuido.**

## Es una colección de computadoras independientes que le dan al usuario la idea de construir un único sistema coherente.

## **2- ¿Cómo tienen que ser los sistemas distribuidos (dos cualidades)?**

## Debe proporcionar los medios para que los componentes de una aplicación se puedan conectar entre sí.

## Debe permitir la comunicación entre las diferentes aplicaciones.

## **3- A los sistemas distribuidos, ¿cómo también se los conoce?**

## Como middleware, porque se organizan como una capa de software entre una capa de alto nivel que consta de usuarios y aplicaciones y otra capa subyacente conformada por sistemas operativos y recursos de comunicación.

## **4- ¿Cómo tienen que permanecer el hardware y el software?**

## Las diferencias entre los distintos tipos de hardware y software deben permanecer ocultas.

## **5- Explique migración, replicación, reubicación y concurrencia.**

## Migración: capacidad de un recurso de moverse a otra ubicación.

## Replicación: es el número de copias de un recurso.

## Reubicación: capacidad de un recurso de moverse a otra ubicación mientras está en uso.

## Concurrencia: capacidad de un recurso de ser compartido por varios usuarios que compiten por él.

## **6- El grado de apertura, ¿qué características tiene?**

## Interoperabilidad: es la extensión mediante la cual dos implementaciones de sistemas y los componentes de fabricantes pueden coexistir y trabajar juntos.

## Extensibilidad: es la facilidad de configuración de distintos componentes, agregar nuevos o reemplazar los existentes sin afectar a aquellos que permanecen en su lugar.

Flexibilidad: es la organización del sistema distribuido como una colección de componentes relativamente pequeños y fáciles de reemplazar o adaptar.

## **7- ¿Y respecto de la escalabilidad?**

La escalabilidad se puede medir desde tres puntos de vista.

## Respecto de su tamaño: podemos agregar fácilmente recursos y usuarios.

Desde el punto de vista geográfico: usuarios y recursos pueden ubicarse muy lejos unos de los otros.

Escalabilidad administrativa: puede ser fácil de manejar incluso involucrando muchas organizaciones administrativas diferentes.

## **7.1- ¿Qué cosas tiene que tener en cuenta un algoritmo descentralizado?**

## La información completa del estado del sistema no se guarda en una máquina en particular.

## Las máquinas toman decisiones en base a la información local.

* La falla en una sola máquina no tiene influencia decisiva sobre todo el algoritmo.
* No existen suposiciones implícitas respecto de la existencia de un reloj global.

**8- Las técnicas de escalamiento, ¿para qué me sirven?**

Ocultar las latencias de comunicación: sirve para lograr escalabilidad geográfica, evitando la espera por respuestas de peticiones remotas de servicios.

Distribución: divide un componente en partes más pequeñas y distribuye las partes a lo largo del sistema.

Replicación: permite replicar los componentes del sistema ante la degradación de rendimiento. Incrementa la disponibilidad y ayuda a balancear la carga entre componentes para mejor rendimiento.

**8.1- ¿Cómo es la arquitectura de CORBA? ¿Cuáles son sus componentes?**

Common Object Request Broker Architecture (CORBA) es un estándar definido por Object Management Group (OMG) que permite que diversos componentes de software escritos en múltiples lenguajes de programación y que corren en diferentes computadoras, puedan trabajar juntos; es decir, facilita el desarrollo de aplicaciones distribuidas en entornos heterogéneos.

Componentes:

ORB: facilita la comunicación entre objetos. Envía las peticiones a los objetos y retornar las respuestas a los clientes que las invocan por el proceso de serialización.

IDL: es un 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.

POA: proporciona un conjunto de interfaces para el manejo de referencias y servants. Desviar peticiones de invocación entrantes al servant correcto.

Repositorio de Interfaces:es un servicio que ofrece objetos persistentes que representan la información IDL de los interfaces disponibles en CORBA, de una forma accesible en tiempo de ejecución (run-time).

Repositorio de Implementaciones: contiene información que permite al ORB localizar y activar la implementación de los objetos.

**8.2- ¿Cómo se originan y qué son el stub y el skeleton? ¿Qué es el IDL?**

El IDL es un 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. El IDL sólo puede definir interfaces, no implementaciones.

Tendremos principalmente dos tipos diferentes de IDL en CORBA:

Stub IDL: es la interfaz estática a los servicios declarados en las interfaces IDL, para el cliente todas las llamadas parecen locales. Actuará como proxy del objeto remoto, realizando la invocación de métodos remotos. El cliente puede tener tantos stubs como interfaces IDL existan. Es generado a partir del IDL en el lenguaje de programación del cliente por un compilador IDL.

Skeleton IDL: es el representante estático del cliente en el servidor. Para el servidor todas las llamadas parecen locales. Es generado a partir del IDL por un compilador IDL y realiza la deserialización de las invocaciones del cliente.

**8.3- ¿El ORB qué función cumple?**

Se encarga de enviar las peticiones a los objetos y retornar las respuestas a los clientes que las invocan por el proceso de serialización. Facilitar la comunicación entre objetos.

**8.4- ¿Cuáles son los servicios de CORBA relacionados con el concepto de publicación-descubrimiento?**

Naming Service: es un directorio de objetos fundamental y con capacidad de ampliación para sistemas distribuidos diseñados para los sistemas empresariales que proporciona un directorio de objetos de páginas blancas.

Trading Service: es un método muy flexible de registro y búsqueda de recursos en sistemas distribuidos basado en una descripción de los requisitos en lugar de una simple asignación de nombre. Admite consultas de búsqueda complejas al agrupar los servicios en categorías, de forma muy similar a como se organiza la guía telefónica de páginas amarillas.

**8.5- ¿En qué se diferencian una invocación estática de una dinámica?**

En la invocación estática, los métodos están definidos estáticamente en tiempo de compilación (fuerte chequeo de tipos en tiempo de compilación), mientras que en la invocación dinámica los métodos se descubren dinámicamente en tiempo de ejecución. Entre sus características se encuentran:

Invocaciones estáticas

* Más fáciles de programar.
* Chequeo de tipos más robusto.
* Buen rendimiento.
* Autodocumentado.

Invocaciones dinámicas

* Sistema más extensible.
* Soporte de búsqueda de servicios en run-time.
* Permite escribir código más genérico.

**9- ¿Qué característica tendría un sistema distribuido ideal?**

* La red es confiable, segura y homogénea.
* La topología no cambia.
* La latencia es igual a cero.
* El ancho de banda es infinito.
* El costo de transporte es igual a cero.
* Existe un administrador.

**10- ¿Qué cosas se tienen en cuenta en todos los tipos de sistemas distribuidos?**

* Arquitectura

## Procesos

## Comunicación

## Asignación de nombres

## Sincronización

## Consistencia y replicación

## Tolerancia a fallos

## Seguridad

## **11- ¿Cómo funcionan los sistemas distribuidos basados en la coordinación?**

## En los sistemas distribuidos basados en la coordinación los mensajes son direccionados por tema (no contienen 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 de los editores lleguen a los suscriptores.

## **12- Explique sistemas distribuidos de cómputo. ¿Cómo es el sistema en clúster y en grid?**

## El sistema en clúster posee hardware subyacente que consta de una colección de estaciones de trabajo o computadoras similares que están conectadas por una LAN de alta velocidad.

## El sistema en grid consta de sistemas distribuidos construidos como un conjunto de sistemas de cómputo en donde cada sistema puede caer dentro de un dominio administrativo distinto con diferentes características de hardware, software y tecnología de red.

## **13- ¿A qué se conoce como RPC Transaccional?**

## Son Remote Procedure Calls transaccionales o llamadas a procedimientos remotos que se encapsulan en una transacción.

**14- ¿Cuáles son los desafíos de los sistemas distribuidos?**

## Heterogeneidad: se aplica a redes, hardware, sistemas operativos, lenguajes de programación e implementaciones de diferentes desarrolladores.

Extensibilidad: es la característica que determina si el sistema puede extenderse de varias maneras.

Seguridad: tiene 3 componentes:

* Confidencialidad: protección contra individuos no autorizados.
* Integridad: protección contra la alteración o corrupción.
* Disponibilidad: protección contra la interferencia que impide el acceso a los recursos.

Escalabilidad: se dice que un sistema es escalable si conserva su efectividad cuando ocurre un aumento significativo en el número de recursos y en el número de usuarios.

Tolerancia a fallas: los programas clientes de los servicios pueden diseñarse para tolerar ciertos fallos.

Concurrencia: existe la posibilidad de acceso concurrente a un mismo recurso. Para que un objeto sea seguro en un entorno concurrente, sus operaciones deben sincronizarse de forma que sus datos permanezcan consistentes.

Transparencia:

* De acceso: permite acceder a los recursos locales y remotos empleando operaciones idénticas.
* De ubicación: permite acceder a los recursos sin conocer su localización.
* De concurrencia: permite que varios procesos operen concurrentemente sobre recursos compartidos sin interferencia mutua.
* De replicación: permite replicar los recursos sin que los usuarios y los programadores necesiten su conocimiento.
* Frente a fallos: permite ocultar fallos.
* De movilidad: permite la reubicación de recursos y clientes en un sistema sin afectar la operación de los usuarios y los programas.
* De rendimiento: permite reconfigurar el sistema para mejorar el desempeño según varíe su carga.
* Al escalado: permite al sistema y a las aplicaciones expandirse en tamaño sin cambiar la estructura del sistema o los algoritmos de aplicación.

**18- ¿Qué proponen Castro y Liskov? ¿Con respecto a qué tema?**

Propusieron el sistema BFT, relacionado a tres temas:

1. Los clientes de un servicio BFT deben ver el servicio como cualquier otro sitio web.
2. Un servicio BFT debe garantizar la consistencia interna cuando actúa como cliente.
3. Los servicios externos deben tratarse como un servicio BFT que actúa como cliente y como una sola entidad.

**19- Un canal seguro se establece en dos fases. Explíquelo.**

Fase 1: El cliente informa al servidor acerca de algoritmos criptográficos que puede manejar, así como también de cualquier método de compresión que soporte. La selección en sí es hecha siempre por el servidor y eso se le informa al cliente.

Fase 2: Siempre se requiere que el servidor se autentique a sí mismo, razón por la cual pasa al cliente un certificado que contiene su 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.

El cliente genera un número aleatorio que será usado por ambos lados para construir una clave de sesión y envía este número al servidor, cifrada con la clave pública del servidor. Además, si se requiere autentificación del cliente, éste firma el número con su clave privada.

En este punto el servidor puede verificar la identidad del cliente después de lo cual se establece el canal seguro.

**20- ¿Cuáles son los modelos arquitectónicos?**

Modelos en capas: Al componente de la capa Ni se le permite llamar a componentes de la capa subyacente Ni-1, pero no al resto de las capas. El control fluye de capa a capa y mientras las peticiones se mueven hacia abajo en la jerarquía, los resultados se mueven hacia arriba.

Basada en objetos: Cada objeto corresponde a un componente y estos componentes se conectan a través de llamadas a procedimientos remotos (coincide con las arquitecturas cliente-servidor).

Centradas en datos: Giran alrededor de la idea de que los procesos se comunican a través de un repositorio común. Se basan en un sistema de archivos distribuidos compartidos donde casi todas las comunicaciones se realizan a través de archivos.

Basadas en eventos: Los procesos se comunican a través de la propagación de eventos, los que opcionalmente transportan datos. En N sistemas distribuidos la propagación de eventos se asocia con los sistemas de *publicación-suscripción*. La idea es que los procesos publiquen eventos después de los cuales el middleware asegura que las recibirán solo aquellos procesos suscritos a tales eventos.

**21- Defina las RPC, RMI y CORBA.**

RPC: lógica de negocios escrita como procedimientos que se invocan de forma remota.

Está basada en la extensión de la noción de llamados a procedimientos locales de modo que el procedimiento invocado no necesita existir en el mismo espacio de direcciones como el procedimiento que hace el llamado.

RMI: lógica de negocios escrita como objetos cuyos métodos son invocados en forma remota.

RMI provee un modelo directo y simple de computación distribuida con objetos Java. Este extiende el modelo Java para que se pueda correr en cualquier lado, brindando portabilidad y seguridad

CORBA: lógica de negocios escrita como objetos que son invocados en forma remota.

**22- ¿Cómo se hace para construir un Web Service y cómo se construye una aplicación en CORBA? Explique las diferencias.**

El desarrollo de aplicaciones con Web Service consta de los siguientes pasos:

1. Crear el Web Service definiendo los Web Methods con su implementación, lógica, parámetros y valor de retorno.
2. Publicar el Web Service para que pueda ser accedido por los clientes.
3. Crear el Web Form cliente con los campos del formulario.
4. Enviar los campos del formulario por variables de sesión o por cookies.
5. Crear el Web Form del lado del servidor que lea y guarde los parámetros pasados por sesión o por cookies en variables locales.
6. En el Web Form del servidor instanciar el Web Service y llamar al Web Method pasándole los parámetros necesarios y guardando en una variable local el resultado.
7. Mostrar el resultado en el Web Form.

El desarrollo de aplicaciones en CORBA consta de los siguientes pasos:

1. Diseñar y definir las interfaces de los objetos.
2. Generar los stubs y skeletons y otros requisitos para el soporte de clases.
3. Escribir las implementaciones de los objetos del servidor.
4. Usar el compilador de Java del lado del cliente para compilar el código escrito en Java y además las clases de Java que fueron generadas por el compilador IDL. Generar el JAR que contiene los binarios .class y otros archivos de recursos necesarios.
5. Publicar el namespace de los objetos directamente accesibles para que los programas cliente puedan acceder.
6. Escribir la aplicación del lado del cliente. Este código se va ejecutar en una estación de trabajo o PC.
7. Compilar el código de la aplicación cliente usando el compilador de Java JDK.