1er Parcial 2017

Tema 1:Explique el término

Cloud Computing: Una nube se define como un conjunto de servicios de aplicaciones y almacenamiento basados en Internet suficientes para soportar la mayoría de las necesidades de los usuarios, lo que les permite activar mayor o totalmente el almacenamiento de datos local y el software de aplicación.

Modelos Arquitectónicos: describen un sistema en términos de las tareas de cálculo y de comunicación realizadas por sus elementos computacionales. Los elementos pueden ser ordenadores individuales o conjuntos de ellos con sus interconexiones.

Objeto Remoto: Objeto que puede recibir invocaciones remotas

NTP: Network Time Protocol, es un protocolo de Internet para sincronizar los relojes de los sistemas informáticos a través del enrutamiento de paquetes en redes con latencia variable. NTP utiliza UDP como su capa de transporte

Tema 2

Cite los puntos destacados de la infraestructura de google

Google ha puesto un esfuerzo significativo en el diseño de una sofisticada infraestructura de sistemas distribuidos. Los puntos destacados de esta infraestructura incluyen:

1. **Infraestructura física:** consta de un gran número de ordenadores en red ubicados en centros de datos de todo el mundo;
2. **Sistema de archivos distribuido** diseñado para soportar archivos muy grandes y muy optimizado.
3. **Sistema de almacenamiento distribuido:** ofrece acceso rápido a conjuntos de datos grandes;
4. **Servicio de bloqueo:** ofrece funciones de sistema distribuidas tales como bloqueo distribuido y acuerdo;
5. **Modelo de programación:** gestión de cálculos paralelos y distribuidos muy grandes a través de la infraestructura física

Métodos del protocolo http con breve descripción

GET: pide el recurso cuyo URL se da como argumento. Si el URL se refiere a datos, entonces el servidor responderá enviando de vuelta los datos indicados por el URL. Si el URL se refiere a un programa, entonces el servidor web ejecutará el programa y devolverá su salida al cliente.
HEAD: esta petición es idéntica a GET, sólo que no devuelve datos. Sin embargo, devuelve toda la información sobre los datos, como el tiempo de la última modificación, su tipo o tamaño.
POST: cuando se necesita enviar información o un elemento al servidor y que lo enviado sea almacenado como un "hijo" o subelemento de un elemento o recurso ya existentes en el servidor.
PUT: Crea/Carga un nuevo recurso al servidor, o en caso de que el objeto ya exista en el servidor reemplaza el recurso existente con el recurso que se carga.
DELETE: el servidor borrará el recurso identificado por el URL.
OPTIONS: el servidor proporciona al cliente una lista de métodos aplicables a un URL (por ejemplo, GET, HEAD, PUT) y sus requisitos especiales.
TRACE: Este método Permite monitorear los mensajes que hay entre el cliente y el servidor web. Principalmente se usa con propósitos de diagnósticos de fallas o para revisar si existen servidores intermediarios en la conexión.

Modelos Fundamentales

Estas cuestiones se consideran en tres modelos:

1. **El modelo de interacción:** trata de las prestaciones y de la dificultad de poner límites temporales en un sistema distribuido, por ejemplo para la entrega de mensajes.
2. **El modelo de fallos:** intenta dar una especificación precisa de los fallos que se pueden producir en los procesos y en los canales de comunicación.
3. **El modelo de seguridad:** discute sobre las posibles amenazas para los procesos y los canales de comunicación. Introduce el concepto de canal seguro, que lo es frente a dichas amenazas.

Tema 3

Una de las medidas de tolerancias a fallos de la semántica al menos una vez es

Semántica de invocación "al menos una vez": el invocante recibe un resultado, en cuyo caso el invocante sabe que el método se evaluó al menos una vez, a menos que se reciba una excepción informando que no se recibe ningún resultado.
La semántica de invocación al menos una vez puede alcanzarse mediante la retransmisión de los

mensajes de petición, que enmascara los fallos por omisión de los mensajes de invocación del resultado, La semántica al menos una vez puede padecer los siguientes tipos de fallos:

1. **Fallos por caída** cuando el servidor que contiene el objeto remoto falla.
2. **Fallos arbitrarios.** En casos donde el mensaje de invocación se retransmite, el objeto remoto puede recibirlo y ejecutar el método más de una vez,provocando que se almacenen o devuelvan valores posiblemente erróneos.

Mecanismo de tolerancia a fallos utilizado por el protocolo tcp

Modelo de fallo. los streams TCP utilizan:
-suma de comprobación para detectar y rechazar los paquetes corruptos
-número de secuencia para detectar y eliminar los paquetes duplicados.
Con respecto a la propiedad de validez, los streams TCP utilizan:
- timeouts y
- retransmisión de los paquetes perdidos.

Se tienen garantizo la entrega incluso cuando alguno de los paquetes subyacentes se haya perdido.
Conexión Rota:
Si la pérdida de paquetes sobrepasa un cierto límite, o la red que conecta un par de procesos en comunicación está severamente congestionada, el software TCP responsable de enviar los mensajes no recibe acuses de recibo de los paquetes enviados y después de un tiempo declarará rota la conexión.

Comunicación ASÍNCRONA, la operación **ENVÍA es NO BLOQUEANTE**,
RECIBE puede tener variantes bloqueantes y no bloqueantes.

Tema 4: Explique 3 características de los sistemas distribuidos

Concurrencia: en una red de computadores, la ejecución de programas concurrentes es la norma.
La coordinación de programas que comparten recursos y se ejecutan de forma concurrente es un tema importante y recurrente.

Inexistencia de reloj global:

1. la cooperación entre programas se realiza mediante el intercambio de mensajes.
2. La coordinación depende de una idea compartida del instante en el que ocurren las

"acciones" de los programas.

3. Existen límites de precisión en los computadores de una red para sincronizar sus relojes.
4. No existe una única noción global del tiempo correcto.

Fallos independientes: todos los sistemas pueden fallar.S.D pueden fallar de formas diferentes:

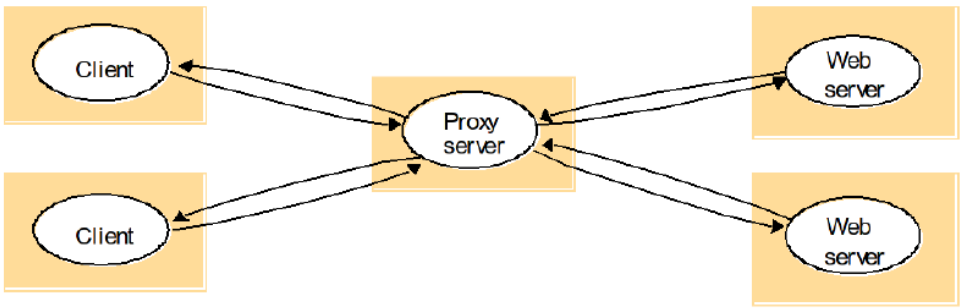
1. Los Fallos en la red. Producen aislamiento de los computadores conectados a él, pero eso no significa que detengan su ejecución. Pueden no ser capaces de detectar cuando la red ha fallado o está excesivamente lenta.
2. Parada de un computador o terminación inesperada de un programa en alguna parte del sistema (crash). No se da a conocer inmediatamente a lo demás componentes con los qué se comunica.

GIT explique para enviar el archivo los persona.java a la rama developer de un repositorio

1. **git init:** crear un nuevo repositorio GIT o
2. **git remote:** conectar a un repositorio remoto.
3. **git checkout developer:** crear ramas o cambiar entre ellas.
4. **git add "persona.java":** agregar archivos al index
5. **git commit -m "se agrego el archivo persona.java":** hacer un comentario al archivo indexado
6. **git pull:** fusionar todos los cambios que se han hecho en el repositorio local
7. **git push:** envía los cambios que se han hecho en la rama principal de los repositorios remotos que están asociados con el directorio que está trabajando.

Servidores proxy caché explicación, incluir gráfica y ejemplo de aplicación

Servidor proxy de tipo web



Un caché es un almacén de objetos de datos utilizados recientemente, y que se encuentra más próximo que los objetos en sí.

Los cachés se utilizan en los navegadores web mantienen un caché de las páginas visitadas recientemente, y de otros recursos web, en el sistema local de ficheros del cliente; entonces utilizan una petición especial HTTP para comprobar si dichas páginas han sido actualizadas en el servidor antes de sacarlas en pantalla.

Los servidores proxy para el web proporcionan un caché compartido de recursos web a las máquinas cliente de uno o más sitios.

El propósito de los servidores proxy es incrementar la disponibilidad y prestaciones del servicio, reduciendo la carga en redes de área amplia y en servidores web.

Imaginemos que abrimos una página web, lo primero que hace el navegador es establecer una conexión TCP con el servidor proxy y envía una solicitud HTTP para el objeto almacenado en dicho servidor proxy.

A continuación, el servidor proxy (o caché web) comprueba si tiene una copia del objeto o archivo que queremos recibir, si la tiene, la caché web devuelve el objeto pedido rápidamente.

progr. java que envíe datos climáticos en formato JSON en protocolo tcp o udp

```
public class DatoClimatico {
 Double visibilidad;
 Double velocidadViento;
 Double temperatura;
 Double precipitaciones;

 public DatoClimatico() {
 }
//Aca van los getters y setters
}

public class UDPClientDC {
        public static void main(String a[]) throws Exception {
        // Datos necesarios
        String direccionServidor = "127.0.0.1";//Definir la dirección ip destino
        if (a.length > 0) {
```

```
direccionServidor = a[0];
}
int puertoServidor = 9876;//Definir el puerto destino

try {
 //Crear un buffer
BufferedReader inFromUser =
        new BufferedReader(new InputStreamReader(System.in));
//Crear un socket UDP
DatagramSocket clientSocket = new DatagramSocket();
//Intentar conectarse con la direccion ip
InetAddress IPAddress = InetAddress.getByName(direccionServidor);
System.out.println("Intentando conectar a = " + IPAddress + ":" + puertoServidor +  " via
UDP...");
byte[] sendData = new byte[1024];
byte[] receiveData = new byte[1024];
DatoClimatico p = new DatoClimatico();

try {
System.out.print("Ingrese la visibilidad");
String temp = inFromUser.readLine();
 p.setVisibilidad(Double.parseDouble(temp));
System.out.print("Ingrese la velocidad del Viento");
temp = inFromUser.readLine();
 p.setVelocidadViento((Double.parseDouble(temp)));
System.out.print("Ingrese la temperatura");
temp = inFromUser.readLine();
 p.setTemperatura(((Double.parseDouble(temp))));
}catch(Exception e1) {
 e1.printStackTrace();
}
//Se pasa de Object a JSON
  JSONObject obj = new JSONObject();
obj.put("precipitaciones", p.getPrecipitaciones().toString());
obj.put("temperatura", p.getTemperatura().toString());
obj.put("visibilidad", p.getVisibilidad().toString());
obj.put("velocidadViento", p.getVelocidadViento());
```

```
        String datoPaquete = obj.toString();//Pasa el JSON a String
        sendData = datoPaquete.getBytes(); //Calcula el tama
        System.out.println("Enviar " + datoPaquete + " al servidor. ("+ sendData.length + "
bytes)");
        DatagramPacket sendPacket =
                new DatagramPacket(sendData, sendData.length, IPAddress, puertoServidor);
        clientSocket.send(sendPacket);
        DatagramPacket receivePacket =
                new DatagramPacket(receiveData, receiveData.length);
        } catch (UnknownHostException ex) {
        System.err.println(ex);
        } catch (IOException ex) {
        System.err.println(ex);
        }
        }
}
```

Comandos básicos de GIT

git config: es usado para establecer una configuración específica de usuario

git init: crear un nuevo repositorio GIT:

git add: agregar archivos al index

git clone: clonar repositorios

git commit: archivo está incluído en el HEAD, pero aún no en tu repositorio remoto.

git status: muestra la lista de los archivos que se han cambiado junto con los archivos que están por ser añadidos o comprometidos.

git push: envía los cambios que se han hecho en la rama principal de los repositorios remotos que están asociados con el directorio que está trabajando.

git checkout: crear ramas o cambiar entre ellas.

git remote: conectar a un repositorio remoto.

git branch: listar, crear o borrar ramas. Para listar todas las ramas se usa: git branch para borrar la rama: git branch -d <branch-name>

git pull: fusionar todos los cambios que se han hecho en el repositorio local

git merge: fusionar una rama con otra rama activa:

git tag: Etiquetar se usa para marcar commits específicos con asas simples.

git log: lista de commits en una rama junto con todos los detalles

git reset: resetear el index y el directorio que está trabajando al último estado comprometido se usa este comando:

git rm: remover archivos del index y del directorio que está trabajando

1er parcial 2016

Explique brevemente

a) NTP

El Protocolo de Tiempo de Red (Network Time Protocol, NTP) define una arquitectura para un servicio de tiempo y un protocolo para distribuir la información sobre Internet. El servicio NTP está proporcionado por una red de servidores localizados a través de Internet. Los servidores primarios están conectados directamente a una fuente de tiempo como un radioreloj recibiendo UTC; los servidores secundarios están sincronizados con servidores primarios. Los servidores están conectados en una jerarquía lógica llamada una subred de sincronización, cuyos niveles se llaman estratos.

b) Git, funcionamiento incluyendo la explicación detallada del comando "git push".

GIT: es un software libre y de código abierto distribuido de control de versiones diseñado para administrar desde pequeños proyectos a muy grandes, con rapidez y eficacia.
Explique git push
git push repositorio rama
git push es un comando que sube los cambios hechos en tu ambiente de trabajo a una rama de trabajo tuya y/o de tu equipo remota. Commit identifica los cambios hechos en dicho ambiente de trabajo. Si tu no haces un push de tus cambios, estos jamas se veran reflejados en tu repositorio remoto.

A nivel de trabajo git push trabaja a nivel de repositorio, es decir con tu repositorio remoto, mientras que git commit trabaja en tu repositorio local.
Cuando ocupas el comando git status y anteriormente hiciste un commit sin haber hecho git push (sin haber aplicado los cambios en tu repositorio remoto) puedes verificar los archivos que localmente modificaste. A esto me refería que "identifica" un commit.

c) Session beans (incluir tipos)

Los session beans son objetos no persistentes que implementan la lógica del negocio que se ejecuta en el servidor. Es posible pensar que un session bean es una extensión lógica del programa cliente que se ejecuta en el servidor y contiene información específica al cliente. El tiempo de vida es limitado por la duración de la sesión del cliente, por lo cual no son persistentes en el tiempo.

Cada session bean mantiene una interacción con un cliente que se desarrolla a través de la ejecución de los distintos métodos que provee. Existen dos tipos de session beans que varían en la forma de modelar esta interacción: stateless y stateful.
Debido a su tiempo de vida reducido y a la no persistencia de sus datos, un session bean -ya sea stateless o stateful- no sobrevive a fallas en el container o en el servidor. En este caso el bean es eliminado de la memoria, siendo necesario que el cliente vuelva a iniciar una conexión para poder continuar con su uso.

d) Web Service.

Un servicio web (en inglés, web service o web services) es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones.

Tema 2 Cite:

Atributos de las transacciones en EJB:

a)Required

b)RequiresNew

c)Mandatory, NotSupported, Supports, Never

Códigos de estado HTTP con una breve descripción

a)1xx: Respuestas informativas

b)2xx: Peticiones correctas

c)3xx: Redirecciones, 4xx Errores del cliente, 5xx Errores de servidor

Características de los sistemas distribuidos

Concurrencia: en una red de computadores, la ejecución de programas concurrentes es la norma. La coordinación de programas que comparten recursos y se ejecutan de forma concurrente es un tema importante y recurrente.

Inexistencia de reloj global: la cooperación entre programas se realiza mediante el intercambio de mensajes. La coordinación depende de una idea compartida del instante en el que ocurren las "acciones" de los programas. Existen límites de precisión en los computadores de una red para sincronizar sus relojes. No existe una única noción global del tiempo correcto.

Fallos independientes: todos los sistemas pueden fallar. Los S.D pueden fallar de formas diferentes:

a. **Fallos en la red**. Producen aislamiento de los computadores conectados a él, pero eso no significa que detengan su ejecución. Pueden no ser capaces de detectar cuando la red ha fallado o está excesivamente lenta.

b. **Parada de un computador** o terminación inesperada de un programa en alguna parte del sistema (crash). No se da a conocer inmediatamente a lo demás componentes con los que se comunica.

Tema 3: Complete con la expresión adecuada:

a) El lenguaje de definición de servicios web está representado mediante un URL, cuyo contenido tiene el formato de un documento **xml**

Los documentos de lenguaje de descripción de servicios Web (WSDL) es un documento XML que describe los servicios Web como una colección de puntos finales o puertos

b) En java, al utilizar el protocolo TCP, el método o forma ***connect*** es utilizado para recibir mensajes y el método o forma **accept** es utilizado para enviar mensajes. ? (solo conozco accept y connect) https://www.programacion.com.py/escritorio/java-escritorio/sockets-en-java-udp-y-tcp

ServerSocket: está diseñada para ser utilizada por un servidor para crear un conector en el puerto de servidor que escucha las peticiones de conexión de los clientes. Su método accept toma una petición connect de la cola, o si la cola está vacía, se bloquea hasta que llega una petición. El resultado de ejecutar accept es una instancia de Socket, un conector que da acceso a streams para comunicarse con el cliente.

c) El modo **Llamada a procedimiento** del protocolo NTP es similar al funcionamiento del algoritmo de Cristian. Un servidor acepta solicitudes de otros computadores, que el procesa respondiendo con su marca de tiempo (lectura actual del reloj)

d) Un método para la comprobación de fallos en la transmisión es **timeouts**

e) La **tasa de deriva del reloj** alude a la proporción en que el reloj de un computador difiere del reloj de referencia perfecto.

f) En una comunicación asíncrona, la primitiva “envía” es **no bloqueante** y la primitiva “recibe” es

bloqueante o no bloqueante

Tema 4

Cite y explique los desafíos de los Sistemas Distribuidos

HETEROGENEIDAD: Internet permite que los usuarios accedan a servicios y ejecuten aplicaciones sobre un conjunto heterogéneo de redes y computadores.

a) **Middleware:** estrato software que provee una abstracción de programación, así como un enmascaramiento de la heterogeneidad subyacente de las redes, hardware, sistemas operativos y lenguajes de programación.
b) **Código móvil:** código que puede ser enviado desde un computador a otro y ejecutarse en éste.

EXTENSIBILIDAD
La extensibilidad de un sistema de cómputo es la característica que determina si el sistema puede ser extendido y re implementado en diversos aspectos.

1. Los sistemas abiertos se caracterizan porque sus interfaces están publicadas.
2. Los sistemas distribuidos abiertos se basan en la providencia de un mecanismo de comunicación uniforme e interfaces públicas para acceder a recursos compartidos.
3. Los sistemas distribuidos abiertos pueden construirse con hardware y software heterogéneo, posiblemente de diferentes proveedores.

SEGURIDAD: Entre los recursos de información que se ofrecen y se mantienen en los sistemas distribuidos, muchos tienen un alto valor intrínseco para sus usuario. La seguridad de los recursos de información tiene tres componentes:
confidencialidad (protección contra el descubrimiento por individuos no autorizados); integridad (protección contra la alteración o corrupción); y
disponibilidad (protección contra interferencia con los procedimientos de acceso a los recursos).

ESCALABILIDAD: Los sistemas distribuidos operan efectiva y eficientemente en muchas escalas diferentes, desde pequeñas intranets a Internet. Se dice que un sistema es escalable si conserva su efectividad cuando ocurre un incremento significativo en el número de recursos y el número de usuarios.

El diseño de los sistemas distribuidos escalables presenta los siguientes retos:

- **Control del costo de los recursos físicos:** según crece la demanda de un recurso, debiera ser posible extender el sistema, a un coste razonable, para satisfacerla.
- **Control de las pérdidas de prestaciones:** considere la administración de un conjunto de datos cuyo tamaño es proporcional al número de usuarios o recursos del sistema
- **Prevención de desbordamiento de recursos software:** un ejemplo de pérdida de escalabilidad se muestra en el tipo de número usado para las direcciones Internet.
- **Evitar cuellos de botella de prestaciones:** en general, para evitar cuellos de botella de prestaciones, los algoritmos deberían ser descentralizados.

TRATAMIENTO DE FALLOS

1. **Detección de fallos:** algunos fallos son detectables. Por ejemplo, se pueden utilizar sumas de comprobación (checksums) para detectar datos corruptos en un mensaje o un archivo.
2. **Enmascaramiento de fallos:** algunos fallos que han sido detectados pueden ocultarse o atenuarse.
3. **Tolerancia de fallos:** la mayoría de los servicios en Internet exhiben fallos; es posible que no sea práctico para ellos pretender detectar y ocultar todos los fallos que pudieran aparecer en una red tan grande y con tantos componentes. Sus clientes pueden diseñarse para tolerar ciertos fallos, lo que implica que también los usuarios tendrán que tolerarlos generalmente. Por ejemplo, cuando un visualizador web no puede contactar con un servidor web no hará que el cliente tenga que esperar indefinidamente mientras hace sucesivos intentos; informará al usuario del problema, dándole la libertad de intentarlo más tarde.
4. **Recuperación frente a fallos:** la recuperación implica el diseño de software en el que, tras una caída del servidor, el estado de los datos pueda reponerse o retractarse (roll back) a una situación anterior.
5. **Redundancia**: puede lograrse que los servicios toleran fallos mediante el empleo redundante de componentes.

CONCURRENCIA

Tanto los servicios como las aplicaciones proporcionan recursos que pueden compartirse entre los

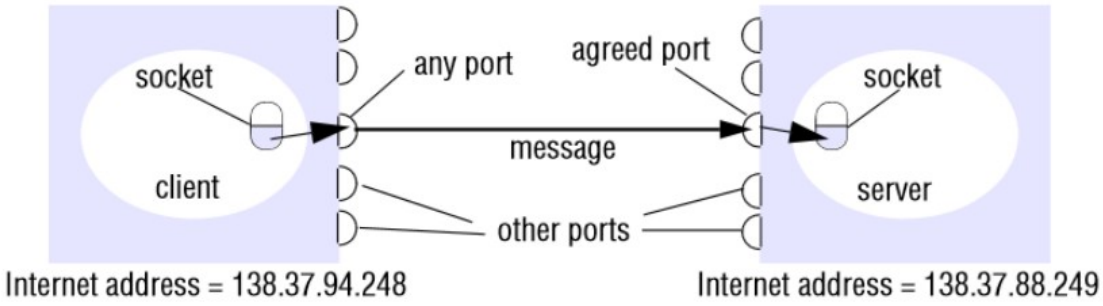
clientes en un sistema distribuido. Existe por lo tanto una posibilidad de que varios clientes intenten acceder a un recurso compartido a la vez.

TRANSPARENCIA

1. **Transparencia de acceso** que permite acceder a los recursos locales y remotos empleando operaciones idénticas.
2. **Transparencia de ubicación** que permite acceder a los recursos sin conocer su localización.
3. **Transparencia de concurrencia** que permite que varios procesos operan concurrentemente sobre recursos compartidos sin interferencia mutua.
4. **Transparencia de replicación** que permite utilizar múltiples ejemplares de cada recurso para aumentar la fiabilidad y las prestaciones.
5. **Transparencia frente a fallos** que permite ocultar los fallos, dejando que los usuarios y programas de aplicación completen sus tareas a pesar de fallos del hardware o de los componentes software.
6. **Transparencia de movilidad** que permite la reubicación de recursos y clientes en un sistema sin afectar la operación de los usuarios y los programas.
7. **Transparencia de prestaciones:** que permite reconfigurar el sistema para mejorar las prestaciones según varía su carga.
8. **Transparencia al escalado:** que permite al sistema ya las aplicaciones expandirse en tamaño sin cambiar la estructura del sistema o los algoritmos de aplicación.

Escribaunobjeto en formato JSON y XML que represente a un mismo dato climático de un lugar y tiempo específico

Sockets definición, funcionamiento, tipos, incluir gráficos explicativos



Sockets

Ambas formas de comunicación (UDP y TCP) utilizan la abstracción de sockets, que proporciona los puntos extremos de la comunicación entre procesos.

Los sockets (conectores) se originan en UNIX BSD aunque están presentes en la mayoría de las versiones de UNIX, incluido Linux y también Windows NT y Macintosh OS.

La comunicación entre procesos consiste en la transmisión de un mensaje entre un conector de un proceso y un conector de otro proceso.

Funcionamiento

Un socket queda definido por un par de direcciones IP local y remota, un protocolo de transporte y un par de números de puerto local y remoto. Para que dos programas puedan comunicarse entre sí es necesario que se cumplan ciertos requisitos:

1. Que un programa sea capaz de localizar al otro.
2. Que ambos programas sean capaces de intercambiarse cualquier secuencia de octetos, es decir, datos relevantes a su finalidad.

Para ello son necesarios los tres recursos que originan el concepto de socket:

1. Un protocolo de comunicaciones, que permite el intercambio de octetos.
2. Un par de direcciones del Protocolo de Red (Dirección IP, si se utiliza el Protocolo TCP/IP), que identifica la computadora de origen y la remota.
3. Un par de números de puerto, que identifica a un programa dentro de cada computadora.

Los sockets permiten implementar una arquitectura cliente-servidor o peer to peer. La comunicación debe ser iniciada por uno de los programas que se denomina programa cliente. El segundo programa espera a que otro inicie la comunicación, por este motivo se denomina programa servidor.

Tipos de socket

En la actualidad existen varios tipos de socket y cada uno por lo regular se asocia a un tipo de protocolo, por ejemplo:
SOCK_STREAM: está asociado al protocolo TCP, este brinda seguridad en la transmisión de datos, seguridad en la recepción, en la integridad y en la secuencia, entre otros.
SOCK_DGRAM: está asociado al protocolo UDP, e indica que los paquetes viajarán en tipo datagramas, el cual tiene una comunicación asíncrona.

2do Final 2017

Tema 1: Defina

a) **Ritmo deriva del reloj:** Tasa de deriva del reloj: proporción en que el reloj de un computador difiere del reloj de referencia perfecto. Incluso, si los relojes de todos los computadores de un sistema distribuido se ponen en hora a la vez, a partir de un tiempo

sus relojes variarán en una magnitud significativa a menos que se apliquen correcciones.

b) **Sistema Distribuido Síncrono:** En la forma SÍNCRONA, los procesos receptor y emisor se sincronizan con cada mensaje. En este caso, tanto envía como recibe son operaciones bloqueantes.
P.emisor, Envía – bloquea hasta que se produce el corresp recibe.
P.receptor, invoca un Recibe, se bloquea hasta que llega un mensaje.

Tema 2: Cite y explique brevemente

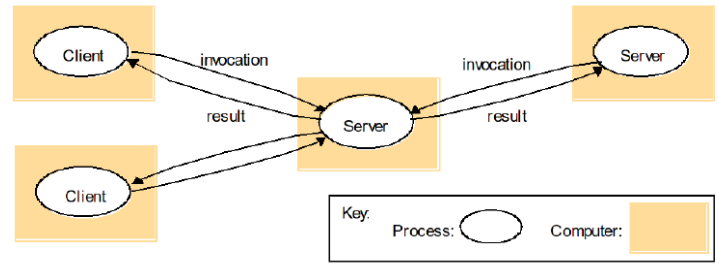
Dos modelos arquitectónicos, incluir gráficos

Modelos Arquitectónicos: describen un sistema en términos de las tareas de cálculo y de comunicación realizadas por sus elementos computacionales.
Ejemplos: modelo cliente-servidor y el modelo de procesos «de igual a igual» (peer to peer).

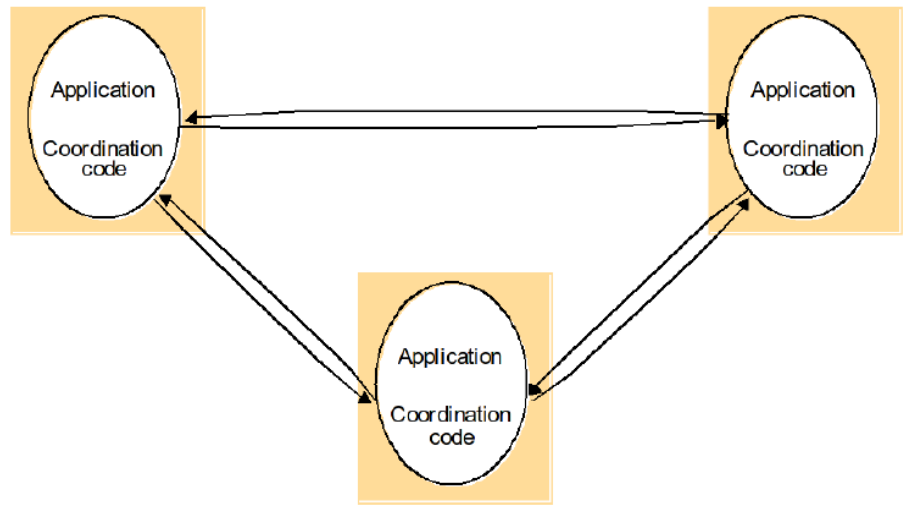
El modelo cliente-servidor admite varias modificaciones debido a:

1. La partición de datos o la replicación en servidores cooperativos.
2. El uso de caché para los datos en clientes y servidores proxy.
3. El uso de código y agentes móviles.
4. Los requisitos para añadir o eliminar dispositivos móviles de forma conveniente.

Clientes que invocan a servidores individuales



Una aplicación distribuida basada en procesos de igual a igual



Cuatro desafíos de Transparencia en los Sistemas Distribuidos

1. **Transparencia de acceso** que permite acceder a los recursos locales y remotos empleando operaciones idénticas.
2. **Transparencia de ubicación** que permite acceder a los recursos sin conocer su localización.
3. **Transparencia de concurrencia** que permite que varios procesos operan concurrentemente sobre recursos compartidos sin interferencia mutua.
4. **Transparencia de replicación** que permite utilizar múltiples ejemplares de cada recurso para aumentar la fiabilidad y las prestaciones.

Explique la Escalabilidad como uno de los desafíos de los Sistemas Distribuidos. Explique los retos y ejemplos asociados

ESCALABILIDAD: Los sistemas distribuidos operan efectiva y eficientemente en muchas escalas diferentes, desde pequeñas intranets a Internet. Se dice que un sistema es escalable si conserva su efectividad cuando ocurre un incremento significativo en el número de recursos y el número de usuarios.

El diseño de los sistemas distribuidos escalables presenta los siguientes retos:

1. **Control del costo de los recursos físicos:** según crece la demanda de un recurso, debiera ser posible extender el sistema, a un coste razonable, para satisfacerla.
2. **Control de las pérdidas de prestaciones:** considere la administración de un conjunto de datos cuyo tamaño es proporcional al número de usuarios o recursos del sistema
3. **Prevención de desbordamiento de recursos software:** un ejemplo de pérdida de

escalabilidad se muestra en el tipo de número usado para las direcciones Internet.

4. **Evitar cuellos de botella de prestaciones:** en general, para evitar cuellos de botella de prestaciones, los algoritmos deberían ser descentralizados.

Características en la comunicación del protocolo de control de transmisión (TCP)

Las principales características del protocolo TCP son las siguientes:

1. TCP permite colocar los datagramas nuevamente en orden cuando vienen del protocolo IP.
2. TCP permite que el monitoreo del flujo de los datos y así evita la saturación de la red.
3. TCP permite que los datos se formen en segmentos de longitud variada para "entregarlos" al protocolo IP.
4. TCP permite multiplexar los datos, es decir, que la información que viene de diferentes fuentes (por ejemplo, aplicaciones) en la misma línea pueda circular simultáneamente.
5. Por último, TCP permite comenzar y finalizar la comunicación amablemente.

Escriba los relojes vectoriales correspondientes a la figura

Tema 4: Complete con V o F, fundamente lo falso

a) El algoritmo de chandy y lamport establece que para sincronizar un reloj de un computador este debe colocar su tiempo a:"t +Tround/2". Siendo t el tiempo del computador con reloj de referencia y Tround el tiempo que tomo obtener el mensaje que contenía el tiempo t.( )

b) El protocolo UDP está implementado sobre el protocolo DNS de la capa de transporte**(F)**
el Servicio de Nombres de Dominio en Internet (Domain Name Service, DNS) está implementado sobre UDP

c) El método Cristian se utiliza para detectar interbloqueos en transacciones que gestionan los

recursos( ).

d) La anotación @webservice se utiliza para indicar que el método de java referenciado será un web service( )

Tema 5: Complete con la expresión adecuada

a) La semántica de invocación "como máximo una vez", las medidas de tolerancia a fallos son **todas las medidas de tolerancia a fallos; ej fallo por caída del servidor, fallo de omisión, fallos arbitrarios.** .
b) En una comunicación asíncrona, la primitiva "envía" es **no bloqueante** y la primitiva "recibe" es **bloqueante y no bloqueante.**

1er Final 2016

Tema 1: Defina

Protocolo HTTP:

HTTP: basado en protocolo petición – respuesta.

Cliente: Navegador.

Servidor: WebServer.

1. HTTP Especifica los mensajes involucrados: En ellos, los métodos, los argumentos, resultados, reglas para representar EMPAQUETAR en dichos mensajes de intercambio.
2. Soporta un conjunto fijo de métodos (GET, PUT, POST, etc.) que son aplicables a todos los recursos.
3. Además permite la negociación de contenidos y una autenticación del tipo clave de acceso.

Sockets

Ambas formas de comunicación (UDP y TCP) utilizan la abstracción de sockets, que proporciona los puntos extremos de la comunicación entre procesos.
Los sockets (conectores) se originan en UNIX

Servidor proxy y cache

Un caché es un almacén de objetos de datos utilizados recientemente, y que se encuentra más próximo que los objetos en sí. Al recibir un objeto nuevo en un computador se añade al almacén del caché, reemplazando, si fuera necesario, algunos objetos existentes

Los servidores proxy para el web proporcionan un caché compartido de recursos web a las máquinas cliente de uno o más sitios. El propósito de los servidores proxy es incrementar la disponibilidad y prestaciones del servicio, reduciendo la carga en redes de área amplia y en servidores web.

Tema 2: Cite

a) **Tres modelos fundamentales**
   El modelo de interacción
   El modelo de fallos
   El modelo de seguridad

Tema 3: Complete con la expresión adecuada

a) Uno de los motivos principales para construir sistemas distribuidos es el de **Compartir recursos .**

b) Los sistemas distribuidos pueden fallar de formas diferentes, en general se engloban cómo "fallos independientes" a los fallos **fallos en la red, Parada de un computador o terminación inesperada de un programa en alguna parte del sistema (crash).**

c) Los servidores NTP se sincronizan entre sí en uno de estos tres modos: y modo simétrico.

d) El protocolo DNS, está implementado sobre el protocolo **UDP** de la capa de transporte.
e) La clase java que representa un paquete UDP es **DatagramPacket y DatagramSocket.**

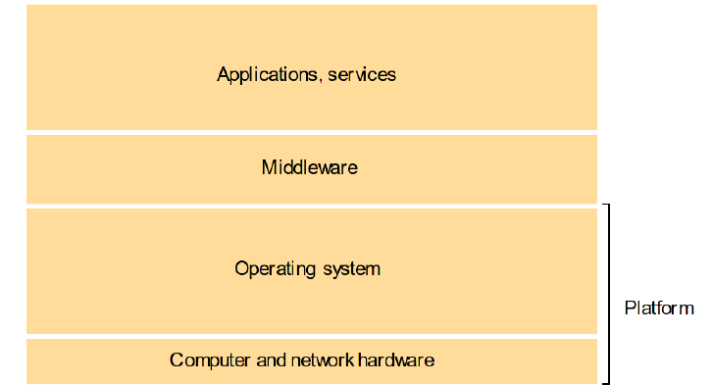
Tema 4. Explique detalladamente

Modelo "Peer to peer":

Una **red peer-to-peer**, red de pares, red entre iguales o red entre pares (P2P, por sus siglas en inglés) es una red de ordenadores en la que todos o algunos aspectos funcionan sin clientes ni servidores fijos, sino una serie de nodos que se comportan como iguales entre sí. Es decir, actúan simultáneamente como clientes y servidores respecto a los demás nodos de la red. Las redes P2P permiten el intercambio directo de información, en cualquier formato, entre los ordenadores interconectados.

Arquitectura en capas, ejemplifica de 2 y 3 capas

Capas de servicio software y hardware en los sistemas distribuidos



Algoritmo instantáneo de chandy y lamport

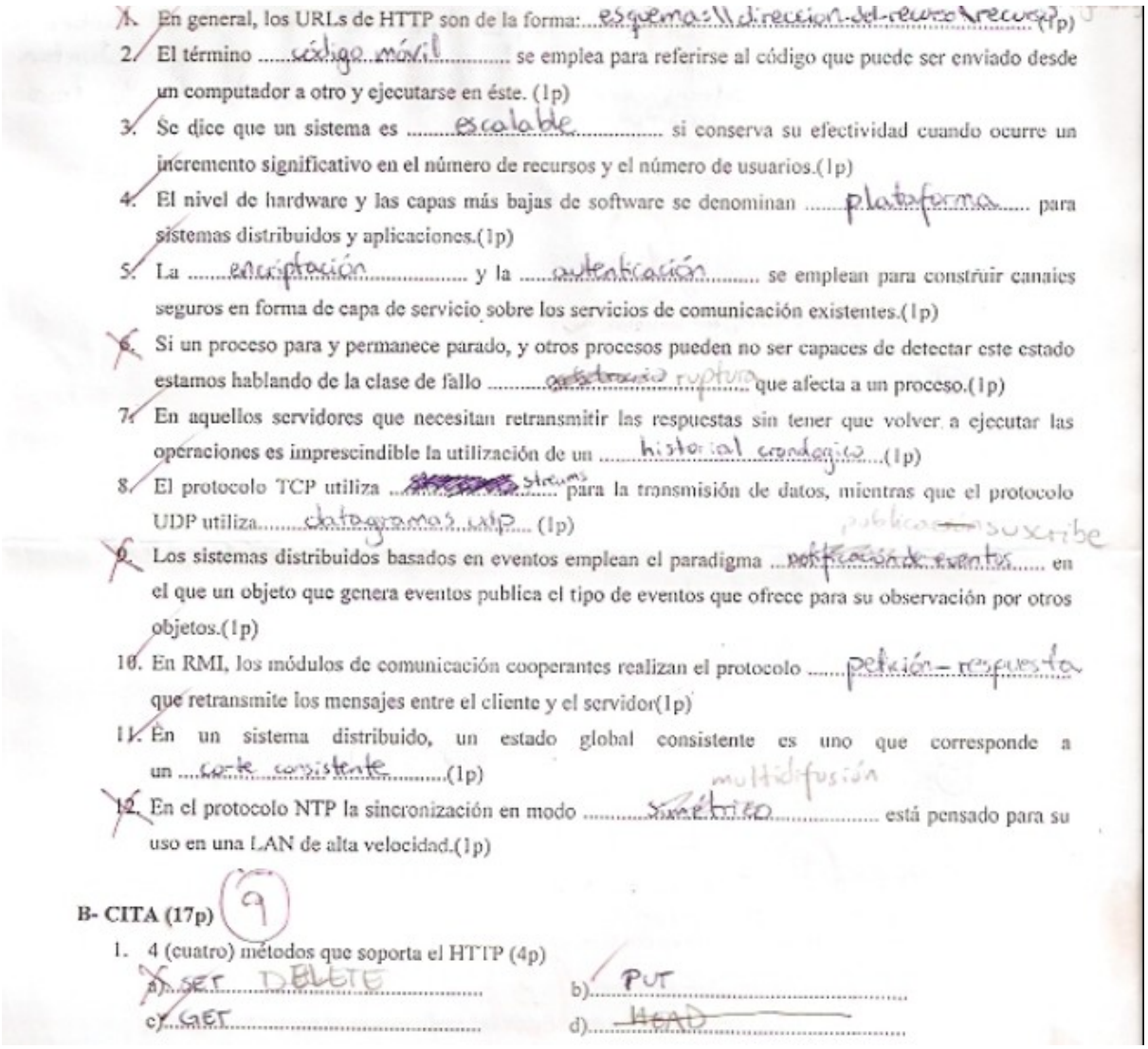
GIT: es un software libre y de código abierto distribuido de control de versiones diseñado para administrar desde pequeños proyectos a muy grandes, con rapidez y eficacia.
Explique git push
git push repositorio rama

git push es un comando que sube los cambios hechos en tu ambiente de trabajo a una rama de trabajo tuya y/o de tu equipo remota. Commit identifica los cambios hechos en dicho ambiente de trabajo. Si tu no haces un push de tus cambios, estos jamas se veran reflejados en tu repositorio remoto.

A nivel de trabajo git push trabaja a nivel de repositorio, es decir con tu repositorio remoto,

mientras que git commit trabaja en tu repositorio local.

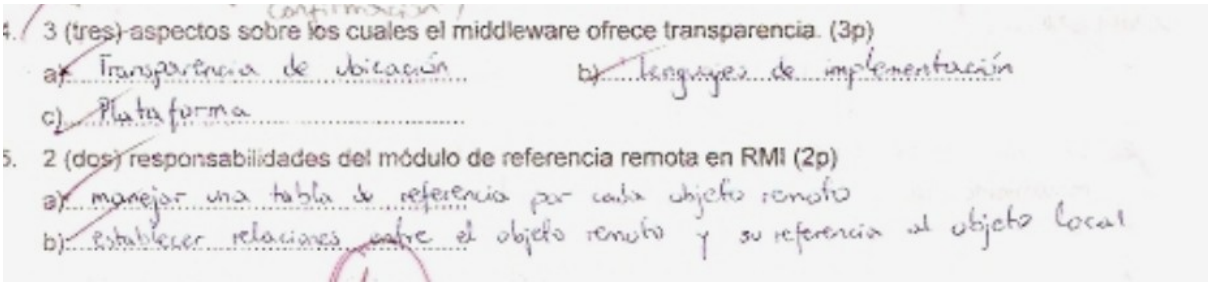
Cuando ocupas el comando git status y anteriormente hiciste un commit sin haber hecho git push (sin haber aplicado los cambios en tu repositorio remoto) puedes verificar los archivos que localmente modificaste. A esto me refería que "identifica" un commit.



1. En general, los URLs de HTTP son de la forma: esquema:\\dirección-del-recurso\recurso (1p)
2. El término código móvil se emplea para referirse al código que puede ser enviado desde un computador a otro y ejecutarse en éste. (1p)
3. Se dice que un sistema es escalable si conserva su efectividad cuando ocurre un incremento significativo en el número de recursos y el número de usuarios.(1p)
4. El nivel de hardware y las capas más bajas de software se denominan plataforma para sistemas distribuidos y aplicaciones.(1p)
5. La encriptación y la autenticación se emplean para construir canales seguros en forma de capa de servicio sobre los servicios de comunicación existentes.(1p)
6. Si un proceso para y permanece parado, y otros procesos pueden no ser capaces de detectar este estado estamos hablando de la clase de fallo caída o ruptura que afecta a un proceso.(1p)
7. En aquellos servidores que necesitan retransmitir las respuestas sin tener que volver a ejecutar las operaciones es imprescindible la utilización de un historial con bajo (1p)
8. El protocolo TCP utiliza streams para la transmisión de datos, mientras que el protocolo UDP utiliza datagramas UDP (1p)
9. Los sistemas distribuidos basados en eventos emplean el paradigma notificación de eventos en el que un objeto que genera eventos publica el tipo de eventos que ofrece para su observación por otros objetos.(1p)
10. En RMI, los módulos de comunicación cooperantes realizan el protocolo petición – respuesta que retransmite los mensajes entre el cliente y el servidor(1p)
11. En un sistema distribuido, un estado global consistente es uno que corresponde a un corte consistente (1p)
12. En el protocolo NTP la sincronización en modo simétrico está pensado para su uso en una LAN de alta velocidad.(1p)

B- CITA (17p) 9

1. 4 (cuatro) métodos que soporta el HTTP (4p)
   a) SET DELETE
   b) PUT
   c) GET
   d) HEAD



Cap 2

El API Java para los streams TCP . La interfaz Java para los streams TCP está constituida por las **clases ServerSocket y Socket.**

ServerSocket: está diseñada para ser utilizada por un servidor para crear un conector en el puerto de servidor que escucha las peticiones de conexión de los clientes. Su método accept toma una petición connect de la cola, o si la cola está vacía, se bloquea hasta que llega una petición. El resultado de ejecutar accept es una instancia de Socket, un conector qué da acceso a streams para comunicarse con el cliente.

Socket: es utilizada por el par de procesos de una conexión. El cliente utiliza un constructor para crear un conector, especificando el nombre DNS de host y el puerto del servidor. Este constructor no sólo crea el conector asociado con el puerto local, sino que también se conecta con el computador remoto especificado en el puerto indicado.

Agentes móviles. es un programa en ejecución (lo que incluye tanto código como datos) que se traslada de un computador a otro en la red realizando una tarea para alguien; por ejemplo, recolectando información, y retornando eventualmente con los resultados.

Sockets

Designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada.

Los sockets de Internet constituyen el mecanismo para la entrega de paquetes de datos provenientes de la tarjeta de red a los procesos o hilos apropiados. Un socket queda definido por un par de direcciones IP local y remota, un protocolo de transporte y un par de números de puerto local y remoto.

Protocolos de Transporte

« UDP (User Datagram Protocol): Es un protocolo no orientado a conexión. Es decir cuando una maquina A envía paquetes a una maquina B, el flujo es unidireccional. La transferencia de datos es realizada sin haber realizado previamente una conexión con la máquina de destino (maquina B), y el destinatario recibirá los datos sin enviar una confirmación al emisor (la maquina A). Esto es debido a que la encapsulación de datos enviada por el protocolo UDP no permite transmitir la información relacionada al emisor. Por ello el destinatario no conocerá al emisor de los datos excepto su IP.

« TCP (Transmission Control Protocol): Contrariamente a UDP, el protocolo TCP está orientado a conexión. Cuando una máquina A envía datos a una máquina B, la máquina B es informada de la llegada de datos, y confirma su buena recepción. Aquí interviene el control CRC de datos que se basa en una ecuación matemática que permite verificar la integridad de los datos transmitidos. De este modo, si los datos recibidos son corruptos, el protocolo TCP permite que los destinatarios soliciten al emisor que vuelvan a enviar los datos corruptos.

Capítulo 4 -Resumen

Aplicaciones Distribuidas: Aplicaciones que se componen de programas cooperantes, corriendo procesos distintos, se aplica mediante:

Llamada a procedimiento remoto (RPC): permite a programas clientes llamar a procedimientos de programas de servidores lanzados en procesos separados.

Invocación de método remoto (RMI): permite que los objetos de diferentes procesos se comuniquen.

Modelo de programación basado en eventos: permite a los objetos recibir notificaciones de los eventos que ocurren en otros objetos.

Protocolo de petición-respuesta

Permite el paso de mensajes

Comunicación entre cliente-servidor

Soporte para la ejecución remota de RPC y RMI

Middleware: Software que provee abstracción de programación, y encapsulamiento de la heterogeneidad subyacente en redes, hardware, sistemas operativos y lenguajes de programación.

Proporciona:

a) transparencia de ubicación

RPC: El programa cliente que llama al procedimiento no sabe si el procedimiento se ejecutará en el mismo proceso o si en otro diferente, tampoco es necesario conocer la ubicación del servidor.

RMI: El objeto que invoca al objeto de otro proceso no necesita saber si es local o no, y tampoco saber su ubicación

Procesos distribuidos basados en eventos: los objetos tampoco necesitan conocer la ubicación de otros objetos.

b) independencia de detalles de protocolos de comunicación

Son protocolos independientes, y el protocolo request-reply puede estar implementado sobre TCP o UDP.

c) independencia de los sistemas operativos

d) independencia el hardware de los computadores.

Se puede tener implementaciones en lenguajes diferentes

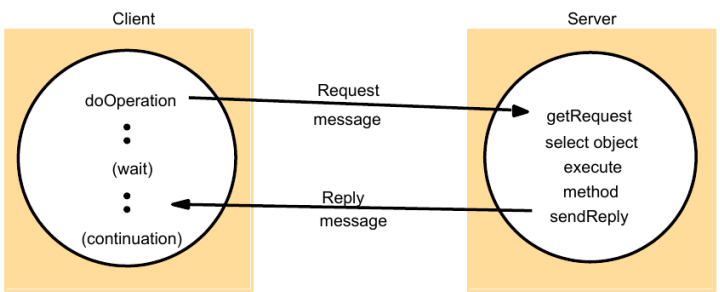
Permitir que las aplicaciones distribuidas estén escritas en más de un lenguaje de programación ej CORBA.

La comunicación **request-reply es:**

síncrona, porque el cliente se bloquea hasta que llegue la respuesta del servidor.

fiable, porque la respuesta del servidor funciona como acuse de recibo para el cliente.

Comunicación cliente-servidor



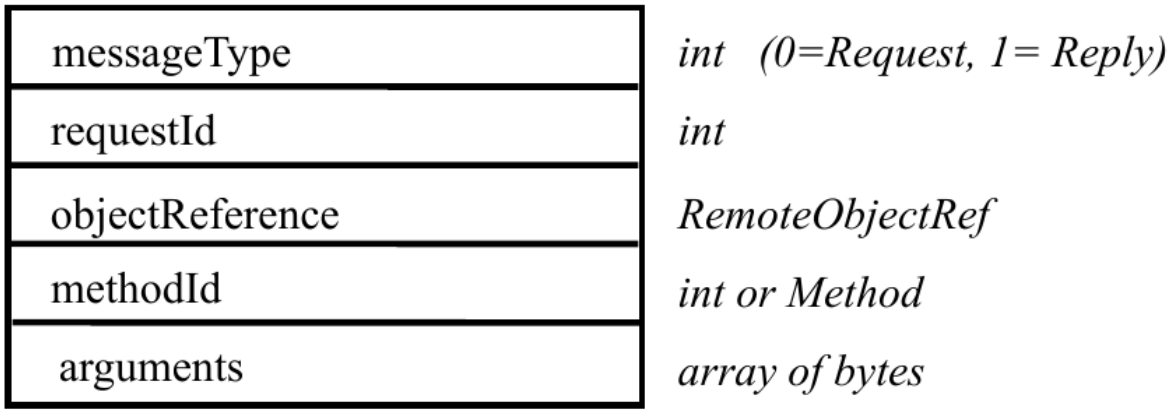
Operaciones del protocolo Petición-Respuesta

doOperation

getRequest

sendReply

Estructura de un mensaje petición-respuesta



Modelos de fallos del protocolo petición-respuesta

UDP: Si las 3 primitivas doOperation, getRequest,sendReply se implementan bajo datagramas udp sufrieran los mismos fallos que sufre ese protocolo.

fallos de omisión y no se garantiza que lleguen en orden correcto de emisión

Eliminación de mensajes de petición duplicados: En los casos que el mensaje de petición se retransmite y el servidor puede recibir más de uno.

Para que el servidor no ejecute una operación más de una vez para una petición se desarrolló un protocolo en donde se reconoce sucesivos mensajes de petición del mismo cliente porque cuenta con id_cliente#id_peticion.

Pérdida de mensaje de Respuesta: Si el servidor ya ha enviado un reply, cuando recibe una petición duplicada deberá ejecutar de nuevo la operación para obtener el resultado.

Operación Idempotente: puede realizarse repetidamente, y es como si hubiera sido ejecutada una sola vez.

Historial: aquellos servidores que necesiten retransmitir las respuestas sin tener qué ejecutar de nuevo ninguna operación, deben utilizar un historial, qué es la estructura que contiene el registro de los mensajes reply qué han sido transmitidos: id_request, message, id_client, su desventaja es el costo de almacenamiento, el servidor es quien decide cuando ya no es necesario almacenar los datos.

Protocolo de intercambio RPC

En la implementación de los distintos tipos de RPC se utilizan tres protocolos

Protocolo de Petición R- Request: el procedimiento no devuelve ningún valor y el cliente no necesita confirmación.

Protocolo Request-Reply RR: no se necesitan mensajes de acuses de recibo, el reply del servidor funciona como uno.

Protocolo Petición-Respuesta-Confirmación RRA: el cliente envía un ack

HTTP:

está basado en el protocolo request-reply

Cliente: Navegador

Servidor: WebServer

Permite la

Negociación de contenido: Las peticiones de los clientes definen qué tipos de datos y con qué estructuras están dispuestos a aceptar.

La autenticación mediante tipos de claves de acceso

Implementado sobre TCP

Tiene un conjunto de métodos:

GET

HEAD

PUT

POST

TRACE

DELETE

OPTIONS

Semántica al menos una vez

SUN RPC

Lenguaje de definición de Interfaces: permite que los objetos implementados en lenguajes diferentes se invoquen unos a otros.

Semánticas de invocación

Medidas de tolerancia a fallos			*Semánticas De Invocación*
Retransmisión de mensajes	*Filtrado De duplicados*	*Reejecución del procedimiento O retransmisión de la respuesta*	
No	No procede	No procede	*Pudiera ser*
Si	No	Reejecutar proced.	*Al menos una vez*
Si	Si	Retransmitir resp.	*Como máximo una vez*

Semánticas de la invocación RPC

Opciones principales

Reintento del mensaje de petición: se retransmite el mensaje request hasta qué se asume qué el servidor ha fallado.

filtrado de duplicados: cuando se realizan retransmisiones se descartan peticiones duplicadas.

Retransmisión de respuesta: se usa el registro historial para hacer las transmisiones de los replays

Semánticas de invocación "pudiera ser"

el qué invoca no puede decir si un metodo se ha ejecutado una vez, o ninguno en absoluto.

Segundo Parcial 2015

Tema 1 Defina:

a) WSDL: WSDL (Web Services Description Language) es un lenguaje para describir servicios Web. Es un lenguaje de interfaz pública para los servicios Web de los requisitos funcionales necesarios

para establecer una comunicación con los servicios Web. Describe servicios Web a partir de los mensajes que se intercambian entre los agentes del request (solicitante) y el provider (proveedor). El mensaje en sí se describe de forma abstracta y luego relacionado a un protocolo de red y formato de mensajes concreto.

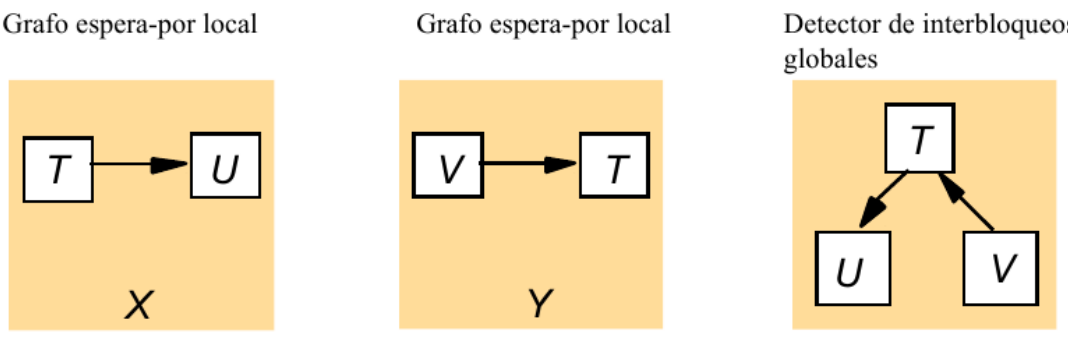
b) Control optimista de la concurrencia: se basa en la observación de que, en la mayoría de las aplicaciones, la similitud entre las transacciones de dos clientes que acceden al mismo objeto es baja. Se permite que las transacciones procedan como si no hubiera posibilidad de conflicto con otras transacciones hasta que el cliente complete su tarea y publique una petición cierraTransacción. Cuando aparece un conflicto, habitualmente se abortará alguna transacción y se necesitará reinicializar el cliente.

c) Equivalencia secuencial: Si se sabe que cada una de las distintas transacciones tiene el efecto correcto cuando se realiza ella sola, podemos inferir que si estas transacciones se realizan una cada veza en el mismo orden, el efecto combinado también será correcto.

d) Qué es y cómo se produce un interbloqueo fantasma?

a) Un interbloqueo que se «detecta» pero que realmente no lo es se conoce como interbloqueo fantasma. En la detección de interbloqueos distribuidos, la información acerca de las relaciones espera por entre las transacciones se transmite de un servidor a otro. Si existe un interbloqueo, la información necesaria se recogerá, al final, en algún lugar y se detectará el ciclo. Ya que este procedimiento tarda un cierto tiempo, existe la posibilidad de tanto, una de las transacciones que mantiene un bloqueo se haya liberado, en cuyo caso ya no existe.

b) Considérese el caso de un detector de interbloqueo global, que recibe los grafos espera por locales de los servidores X e Y, como se muestra en la Figura 9.14. Supóngase que la transacción U a continuación libera un objeto en el servidor X y solicita el que mantene V en el servidor Y. Supóngase además que el detector global recibe el grafo local del servidor Y antes que el del servidor X. En este caso, detectaría un ciclo T- U - V - T, aunque el arco T- U ya no existe. Este es un ejemplo de interbloqueo fantasma.



2 Cite y explique brevemente:

a) Tres métodos de sincronización utilizados en los servidores NTP.

Los servidores NTP se sincronizan entre sí en uno de estos tres modos:

1. **Multidifusión:** está pensado para su uso en una LAN de alta velocidad. Uno o más servidores reparten periódicamente el tiempo a los servidores que se ejecutan en otros computadores conectados en la LAN, que fijan sus relojes suponiendo un pequeño retardo. Este modo puede alcanzar sólo precisiones relativamente bajas, pero que son consideradas suficientes para muchos propósitos.
2. **Llamada a procedimiento:** es similar al funcionamiento del algoritmo de Cristian. Un servidor acepta solicitudes de otros computadores, que el procesa respondiendo con su marca de tiempo (lectura actual del reloj). Este modo es adecuado donde se requieren precisiones más altas que las que se pueden conseguir con multidifusión, o donde la multidifusión no viene soportada por hardware.
3. **Simétrico:** está pensado para su utilización por servidores que proporcionan información del tiempo en LANs y por los niveles más altos (estratos más bajos) de la subred de sincronización, donde se deben obtener las precisiones más altas.

b) Cuatro Propiedades deseables de las transacciones.

1. **Atomicidad (Atomicity):** Una transacción es una unidad atómica de procesamiento; es realizada enteramente o no es realizada en nada.
2. **Consistencia (Consistency):** La ejecución de la transacción debe pasar la BD desde un estado consistente a otro también consistente.
3. **Aislamiento (Isolation):** Una transacción no deberá hacer visible sus modificaciones a otras transacciones hasta que esté confirmada.
4. **Persistencia (Durability):** Cuando una transacción cambia la BD y los cambios son confirmados, estos cambios no deben perderse por fallos posteriores.

c) Tres pasos de los algoritmos de captura de arcos o caza de arcos.

1) iniciación
2) detección
3) resolución

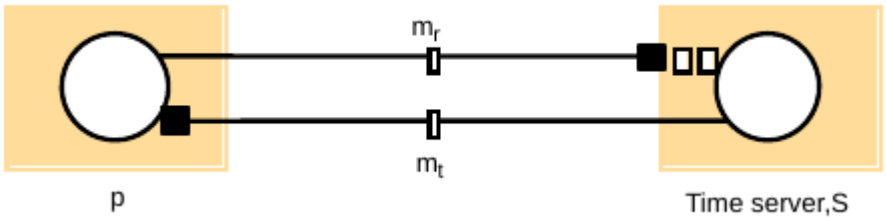
Tema 3 Resuelva:

a) Explique el método de Cristian para sincronizar relojes. Incluir fórmulas.

Cristian [1989] sugirió la utilización de un servidor de tiempo, conectado a un dispositivo que recibe señales de una fuente de UTC, para sincronizar computadores externamente. Bajo solicitud, el proceso servidor S proporciona el tiempo de acuerdo con su reloj.

El describe el algoritmo como probabilístico: el método consigue sincronización sólo si los tiempos de ida y vuelta entre el cliente y el servidor son suficientemente cortos comparados con la precisión requerida.

- Un proceso p solicita el tiempo en un mensaje mr y recibe el valor del tiempo t en un mensaje mt.
- El proceso p registra el tiempo total de ida y vuelta Tround tomado para enviar la solicitud mr y recibir la respuesta mt . Se puede medir este tiempo con precisión razonable si su ritmo de deriva de reloj es pequeño.
- Una estimación sencilla del tiempo al que p debe fijar su reloj es t + Tround/2, qué supone que el tiempo transcurrido se desdoble igualmente antes y después de qué S coloque t en mt.

b) 

Escriba, en el lenguaje Java, un WebService que contenga 2 métodos web. Además escriba la representación XML del request y response para la invocación de uno de sus métodos.

```java
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;

/**
 *
 * @author l
 */
@WebService(serviceName = "WsSuma")
public class WsSuma {

    /**
     * This is a sample web service operation
     */
    @WebMethod(operationName = "hello")
    public String hello(@WebParam(name = "name") String txt) {
        return "Hello " + txt + " !";
    }

    /**
     * Web service operation
     */
    @WebMethod(operationName = "sumar")
    public int sumar(@WebParam(name = "a") int a, @WebParam(name = "b") int b) {
        //TODO write your implementation code here:
        return a + b;
    }

    @WebMethod
    public int restar(int a, int b) {
        return 0;
    }
}
```

import java.jws.WebService;

import java.jws.WebMethod;

import java.jws.WebParam;

import java.jws.rs.Consumes

```
import java.jws.rs.PathParam
import java.jws.rs.Produces
import java.jws.rs.core.MediaType
import com.thoughtworks.xstream.XStream;



@WebService(serviceName=”WsOperaciones”)
@Path(“/operaciones”)
public class WsSuma{
        @Path(/sumar)
        @WebMethod(operationsName = “sumar”)
        @GET
        @Consumes(MediaType.APPLICATION_XML)
        @Produces(MediaType.APPLICATION_XML))
        public Response sumar(Numeros s) {
                XStream xstream = new XStream();
                return xstream.toXML( s.a + s.b);
        }
        @Path(/restar)
        @WebMethod(operationsName = “restar”)
        @GET
        @Consumes(MediaType.APPLICATION_XML)
        @Produces(MediaType.APPLICATION_XML))
        public Response restar(Numeros s) {
                XStream xstream = new XStream();
                return xstream.toXML(s.a - s.b);

        }

}

public class Numeros{
        int a;
        int b;
        public Numeros() {
        }
}
```

Para el método “hello”

Entrada: Melisa

Respuesta: Hello Melisa !

SOAP Request:

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header/>
```

<S:Body>
<ns2:hello xmlns:ns2="http://pol.py/">
<name>Melisa</name>
</ns2:hello>
</S:Body>
</S:Envelope>

SOAP Response:

<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<S:Body>
<ns2:helloResponse xmlns:ns2="http://pol.py/">
<return>Hello Melisa !</return>
</ns2:helloResponse>
</S:Body>
</S:Envelope>

Para el metodo “sumar

Entradas: 1 y 2

Respuesta: 4

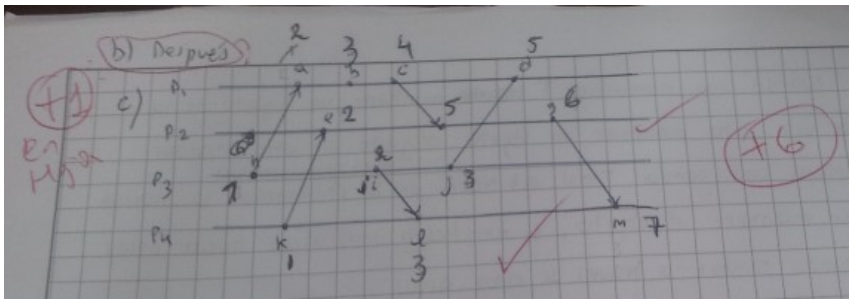
SOAP Request

<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<S:Body>
<ns2:sumar xmlns:ns2="http://pol.py/">
<a>1
2
</ns2:sumar>
</S:Body>
</S:Envelope>

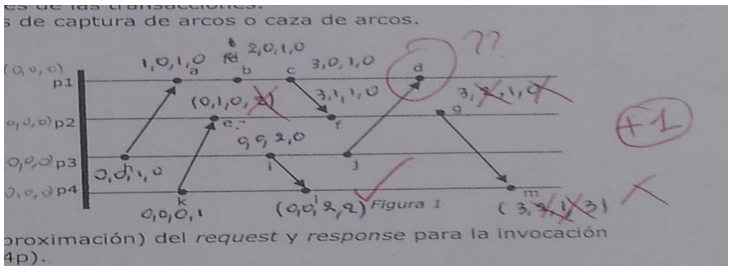
SOAP Response

<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<S:Body>
<ns2:sumarResponse xmlns:ns2="http://pol.py/">
<return>3</return>
</ns2:sumarResponse>
</S:Body>
</S:Envelope>

c) Identifique y complete las marcas temporales de Lamport de los eventos de la figura 1.



d) Identifique los relojes vectoriales en la figura 1 para los eventos d,e,g,i y m.



e) Explique e indique el nombre de la primera fase del "protocolo de consumación en dos fases".

puedeConsumar?(trans) → *Sí / No*
Llamada desde el coordinador al participante para preguntar si puede consumar una transacción. El participante responde con su voto.

Consuma(trans)
Llamada desde el coordinador al participante para decirle que consume su parte de una transacción.

Aborta(trans)
Llamada desde el coordinador al participante para decirle que aborte su parte de una transacción.

heConsumado(trans, participante)
Llamada desde el participante al coordinador para confirmar que ha consumado la transacción.

dameDecisión(trans) → *Sí / No*
Llamada desde el participante al coordinador para preguntar por la decisión sobre una transacción tras haber votado *Sí* aunque no ha obtenido respuesta tras cierto tiempo. Se utiliza para recuperarse de la caída de un servidor o de mensajes con retraso.

Fase 1 (fase de votación):

1. El coordinador envía una petición *puedeConsumar?* a cada participante en la transacción.
2. Cuando un participante recibe una petición *puedeConsumar?*, responde al coordinador con su voto (*Sí* o *No*). Antes de votar *Sí*, se prepara para consumar, guardando los objetos en un dispositivo de almacenamiento permanente. Si el voto es *No*, el participante aborta inmediatamente.

Fase 2 (finalización en función del resultado de la votación)

3. El coordinador recoge los votos (incluyendo el propio).
   (a) Si no hay fallos y todos los votos son *Sí*, el coordinador decide consumar la transacción y envía peticiones de *Consuma* a cada uno de los participantes.
   (b) En otro caso, el coordinador decide abortar la transacción y envía peticiones *Aborta* a todos los participantes que votaron *Sí*.
4. Los participantes que han votado *Sí* están esperando por una petición *Consuma* o *Aborta* por parte del coordinador. Cuando un participante recibe uno de estos mensajes, actúa en función de ellos, y en el caso de *Consuma*, realiza una llamada de *heConsumado* como confirmación hacia el coordinador.

Tema 4 Complete:

a) En JavaRMI, el método **rebind** del Registry es utilizado para publicar/disponibilizar un objeto remoto desde el servidor; y el método **lookup** también del Registry es utilizado en el cliente para encontrar el objeto remoto.

b) Uno de los problemas que evita el control de concurrencia es **el problema de las actualizaciones perdidas.**

c) Una forma de detectar un bloqueo indefinido es **utilizado el grafo espera por**

d) Al implementar mecanismos de bloqueo, si una transacción T ha realizado ya una operación de **lectura** en un objeto particular, entonces una transacción concurrente U no debe **escribir** ese objeto hasta la consumación de T, o que aborte.

Primer Parcial 2016

Tema 1 Explique Brevemente:

a) NTP: El Protocolo de Tiempo de Red (Network Time Protocol, NTP) define una arquitectura para

un servicio de tiempo y un protocolo para distribuir la información sobre Internet.

b) Session beans: Los session beans son objetos no persistentes que implementan la lógica del negocio que se ejecuta en el servidor. Es posible pensar que un session bean es una extensión lógica del programa cliente que se ejecuta en el servidor y contiene información específica al cliente. El tiempo de vida es limitado por la duración de la sesión del cliente, por lo cual no son persistentes en el tiempo. Cada session bean mantiene una interacción con un cliente que se desarrolla a través de la ejecución de los distintos métodos que provee. Existen dos tipos de session beans que varían en la forma de modelar esta interacción: stateless y stateful. Debido a su tiempo de vida reducido y a la no persistencia de sus datos, un session bean -ya sea stateless o stateful- no sobrevive a fallas en el container o en el servidor. En este caso el bean es eliminado de la memoria, siendo necesario que el cliente vuelva a iniciar una conexión para poder continuar con su uso.

c) Session beans (incluir tipos)

Los session beans son objetos no persistentes que implementan la lógica del negocio que se ejecuta en el servidor. Es posible pensar que un session bean es una extensión lógica del programa cliente que se ejecuta en el servidor y contiene información específica al cliente. El tiempo de vida es limitado por la duración de la sesión del cliente, por lo cual no son persistentes en el tiempo. Cada session bean mantiene una interacción con un cliente que se desarrolla a través de la ejecución de los distintos métodos que provee. Existen dos tipos de session beans que varían en la forma de modelar esta interacción: stateless y stateful. Debido a su tiempo de vida reducido y a la no persistencia de sus datos, un session bean -ya sea stateless o stateful- no sobrevive a fallas en el container o en el servidor. En este caso el bean es eliminado de la memoria, siendo necesario que el cliente vuelva a iniciar una conexión para poder continuar con su uso.

c) Web Service: Los servicios web o Web Services por sus siglas en inglés, son interfaces de programación disponibles para la comunicación de aplicaciones en la World Wide Web.

Tema 2 Cite: Atributos de las transacciones en EJB:

a)Required

b)RequiresNew

c)Mandatory, NotSupported, Supports, Never

Tema 3: Complete

a) El lenguaje de definición de servicios web está representado mediante un URL, cuyo contenido tiene el formato de un documento **XML**.

b) El modo **Llamada a procedimiento** del protocolo NTP es similar al funcionamiento del algoritmo de Cristian. Un servidor acepta solicitudes de otros computadores, que el procesa respondiendo su marca de tiempo (lectura actual del reloj).

c) La **tasa de deriva** del reloj alude a la proporción en que el reloj de un computador difiere del reloj de referencia perfecto.

Segundo Parcial 2016

Tema 1 Explique Brevemente:

a) Transacción: define una secuencia de operaciones que se realiza por el servidor y se garantiza por el mismo que es atómica, ya sea en presencia de múltiples usuarios e incluso de caídas del servidor. Los servidores deben garantizar que se realiza completamente y qué los resultados se almacenan en memoria permanente o no.

b) Interbloqueo fantasma: Un interbloqueo que se «detecta» pero que realmente no lo es se conoce como interbloqueo fantasma. En la detección de interbloqueos distribuidos, la información acerca de las relaciones espera por entre las transacciones se transmite de un servidor a otro. Si existe un interbloqueo, la información necesaria se recogerá, al final, en algún lugar y se detectará el ciclo. Ya que este procedimiento tarda un cierto tiempo, existe la posibilidad de tanto, una de las transacciones que mantiene un bloqueo se haya liberado,
en cuyo caso ya no existe.

c) Propiedad de Aislamiento de las transacciones: Cada transacción debe ser realizada sin interferencia de otras transacciones, es decir, los efectos intermedios de una transacción no deben ser visibles para los demás.

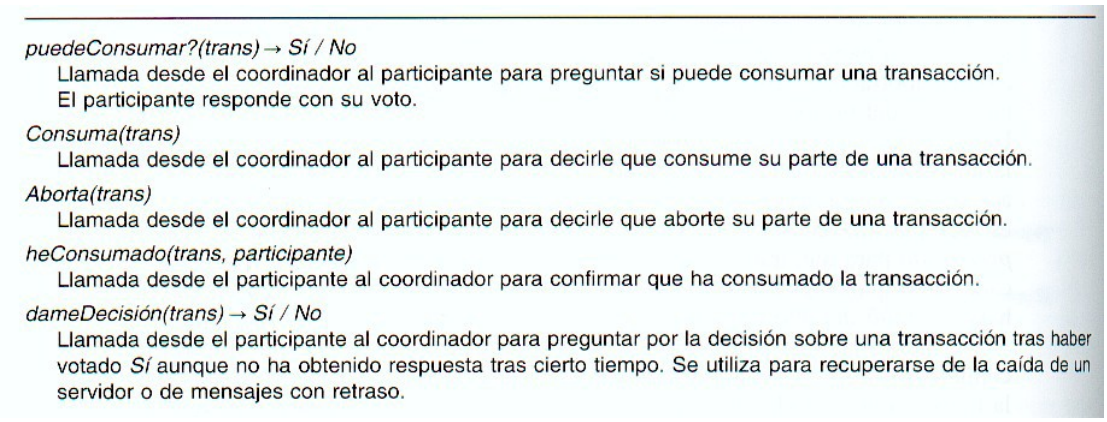
d) Registro Histórico: En la técnica de registro histórico, el archivo de recuperación representa a un registro que contiene el historial de todas las transacciones realizadas por el servidor.

Tema 2 Cite:

a) Propiedades de las transacciones:

1. **Atomicidad (Atomicity):** Una transacción es una unidad atómica de procesamiento; es realizada enteramente o no es realizada en nada.
2. **Consistencia (Consistency):** La ejecución de la transacción debe pasar la BD desde un estado consistente a otro también consistente.
3. **Aislamiento (Isolation):** Una transacción no deberá hacer visible sus modificaciones a otras transacciones hasta que esté confirmada.
4. **Persistencia (Durability):** Cuando una transacción cambia la BD y los cambios son confirmados, estos cambios no deben perderse por fallos posteriores.

b) Operaciones dentro del protocolo de consumación en dos fases



puedeConsumar?(trans) → *Sí / No*
Llamada desde el coordinador al participante para preguntar si puede consumar una transacción. El participante responde con su voto.

Consuma(trans)
Llamada desde el coordinador al participante para decirle que consume su parte de una transacción.

Aborta(trans)
Llamada desde el coordinador al participante para decirle que aborte su parte de una transacción.

heConsumado(trans, participante)
Llamada desde el participante al coordinador para confirmar que ha consumado la transacción.

dameDecisión(trans) → *Sí / No*
Llamada desde el participante al coordinador para preguntar por la decisión sobre una transacción tras haber votado *Sí* aunque no ha obtenido respuesta tras cierto tiempo. Se utiliza para recuperarse de la caída de un servidor o de mensajes con retraso.

c) Pasos en los algoritmos de caza de arcos:

Tema 3 Complete:

a) Cuando decimos que un par de operaciones en transacciones tienen conflictos queremos decir que su efecto combinado depende del **orden en el que se ejecutan**.

b) Cuando una subtransacción finaliza, hace una decisión independiente sobre si consumarse provisionalmente o abortar. Su decisión de **abortar** es final.

c) Si una transacción T ha realizado ya una operación de **lectura** en un objeto particular, entonces una transacción concurrente U no debe **escribir** ese objeto hasta la consumación de T, o que aborte.

d) Un método para resolver los bloqueos indefinidos es el uso de **los timeouts de bloqueo**.

Tema 4 Resuelva:

a) Escriba 2 transacciones cada una con operaciones sobre objetos "a", "b" y "c", posteriormente re-escriba dichas transacciones para un solapamiento secuencialmente equivalentes. Obs: Graficar cada transacción en una columna, tal como fue analizada en clase.

Transaction *T*: *balance = b.getBalance()* *b.setBalance(balance*1.1)* *a.withdraw(balance/10)*		**Transaction *U*:** *balance = b.getBalance()* *b.setBalance(balance*1.1)* *c.withdraw(balance/10)*	
balance = b.getBalance()	$200		
*b.setBalance(balance*1.1)*	$220		
		balance = b.getBalance()	$220
		*b.setBalance(balance*1.1)*	$242
a.withdraw(balance/10)	$80		
		c.withdraw(balance/10)	$278

La figura muestra dicho solapamiento en el que las operaciones que afectan a la cuenta compartida B, son realmente secuenciales, puesto que la transacción T realiza todas las operaciones antes que las realice la transacción U. Otro solapamiento de T y U que tenga esta propiedad es uno en el que la transacción U finaliza sus operaciones en la cuenta B antes de que comience la transacción T.

b) Explique el método de "Versiones Provisionales".

Las operaciones de actualización durante una transacción se hacen sobre versiones provisionales de los objetos en memoria volátil, estas almacenan los valores de los objetos en el propio conjunto privado de la transacción si es posible o fallan

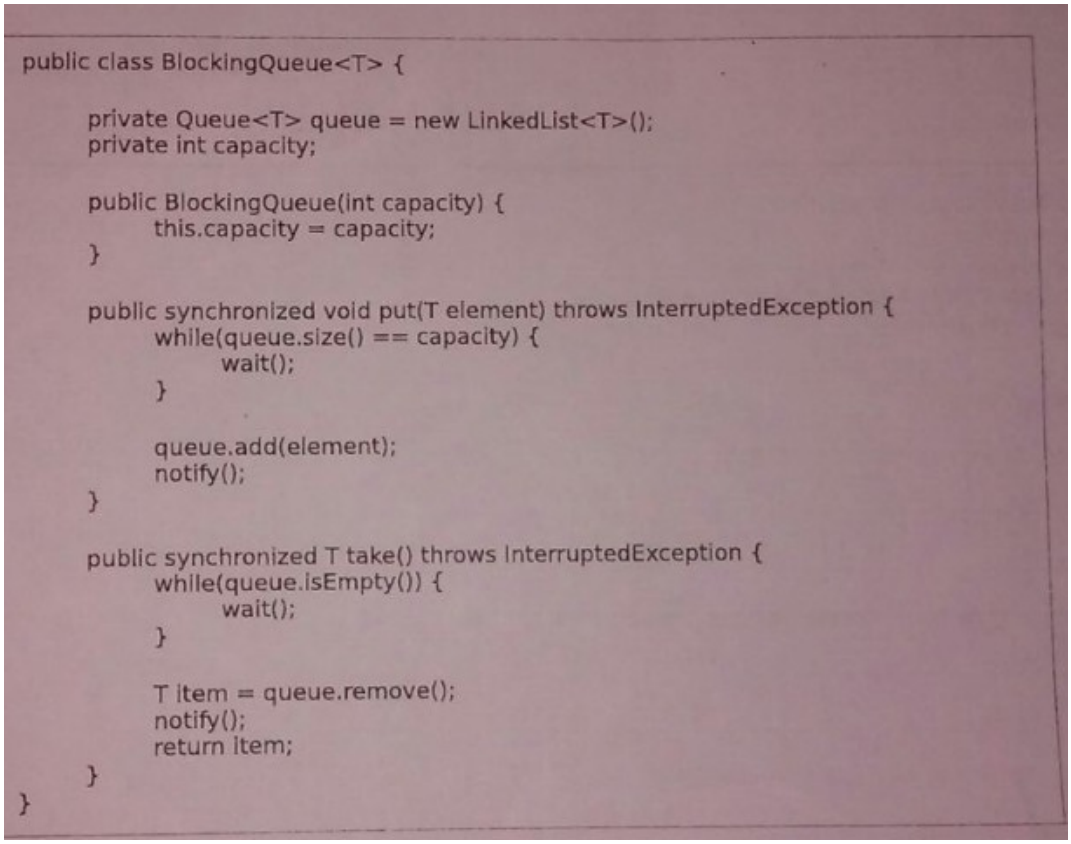
Las versiones provisionales son transferidas a los objetos sólo cuando la transacción se consuma entonces quedarán registradas en la memoria permanente, si una transacción aborta sus versiones provisionales se borran.

c) Explique la fase 1 del protocolo de consumación en dos fases referente a transacciones distribuidas.

1- El coordinador envía una petición puedeConsumar? a cada participante en la transacción

2- Cuando un participante recibe una petición puedeConsumar?, responde al coordinador con su voto (Si o no). Antes de votar si, se prepara para consumar, guardando los objetos en un dispositivo de almacenamiento. Si el voto es No, el participante aborta inmediatamente.

d) En la figura 1, explique las transacciones wait y notify dentro del contexto del método put



```
public class BlockingQueue<T> {

	private Queue<T> queue = new LinkedList<T>();
	private int capacity;

	public BlockingQueue(int capacity) {
		this.capacity = capacity;
	}

	public synchronized void put(T element) throws InterruptedException {
		while(queue.size() == capacity) {
			wait();
		}

		queue.add(element);
		notify();
	}

	public synchronized T take() throws InterruptedException {
		while(queue.isEmpty()) {
			wait();
		}

		T item = queue.remove();
		notify();
		return item;
	}
}
```

Los métodos wait (espera) y notify (notifica) de Java permiten que los hilos se comuniquen con los otros de una manera que resuelva los problemas anteriores.

Un hilo llama a wait en un objeto para suspenderse él mismo y permitir a otro hilo ejecutar un método en ese objeto. Un hilo llama a notify para informar que cualquier hilo que esté esperando en el objeto que ha cambiado algunos de sus datos.

El acceso a un objeto es todavía atómico cuando un hilo espera por otro: un hilo que llama a wait activa su bloqueo y se suspende como una acción atómica, cuando el hilo es activado después de ser notificado y adquiere un nuevo bloqueo en el objeto y recupera la ejecución del wait.

wait es para esperar a que alguien realice una acción sobre ese objeto y notify es para avisar que se realizó alguna acción o modificación

Dentro del método put, mientras la cola esté al límite de su capacidad, con **wait()** el hilo espera a que se realice alguna modificación sobre la cola (o sea, el mismo objeto instanciado **this**). Luego de salir del mientras, se agrega un objeto en la cola y con **notify()** se notifica al primer hilo en la cola de espera, que se realizó una modificación sobre la cola (o sea, el mismo objeto instanciado **this**) y finaliza la ejecución del método.

1er Final 2016

Tema 2 Cite:

a) Dos desventajas de utilizar bloqueos para el control de concurrencia.

1. Mantener el bloqueo representa sobrecarga que no está presente en los sistemas que no soportan acceso concurrente a los datos compartidos.
2. Puede producir un bloqueo indefinido. Su prevención reduce la concurrencia de forma severa, y por tanto las situaciones de bloqueo indefinido deben ser resueltas o por el uso de timeouts o por la detección de bloqueo indefinido. Ninguna de ellas es totalmente satisfactoria para uso en programas interactivos.
3. Para impedir abortos en cascada, los bloqueos no pueden ser liberados hasta el final de la transacción. Esto puede reducir significativamente el potencial de concurrencia.

Tema 3 Complete:

a) Los servidores NTP se sincronizan entre sí en uno de estos tres modos **Multidifusión**, **llamada a procedimiento** y modo simétrico.

Tema 4 Explique:

a) Algoritmo de instantánea de Chandy y Lamport.

Chandy y Lamport [1985] describen un algoritmo de ***instantánea*** para determinar estados globales de sistemas distribuidos.

El objetivo es registrar un conjunto de estados de procesos y canales (una *instantánea)* para un conjunto de procesos *pi(i* = 1, 2, ..., *N)* tal que, incluso a través de una combinación de estados registrados que nunca podrían haber ocurrido al mismo tiempo, el estado global registrado sea consistente.

Veremos que el estado *que registra el algoritmo* de *instantánea* tiene propiedades convenientes para evaluar predicados del estado global.

El algoritmo registra el estado localmente en los procesos; no proporciona un método para recoger el estado global en un sitio.

b) Complete el reloj vectorial correspondiente a la figura.

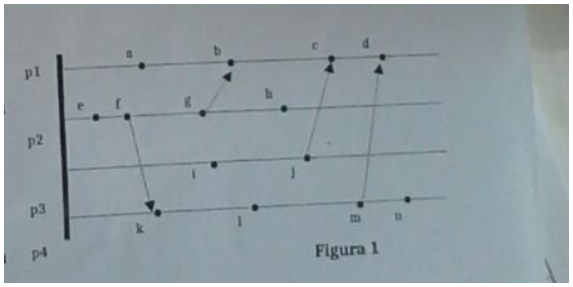


Figura 1

Segundo Parcial 2017

Tema 1 Explique brevemente

a) Transacción distribuida. Indicar elementos involucrados.

Transacción plana o anidada que accede a objetos gestIonados por múltIples servidores. **Involucran** más de un servidor. Cuando una transacción llega a su fin, la propiedad de atomicidad requiere que, todos los servidores completen su transacción o que todos aborten. Uno de los servidores asume de coordinador que debe asegurar el mismo resultado en todos los servidores.

b) OAuth. Definición y funcionamiento

OAuth2 es un protocolo de autorización que permite a terceros (clientes) acceder a contenidos propiedad de un usuario (alojados en aplicaciones de confianza, servidor de recursos) sin que éstos tengan que manejar ni conocer las credenciales del usuario. Es decir, aplicaciones de terceros pueden acceder a contenidos propiedad del usuario, pero estas aplicaciones no conocen las credenciales de autenticación .

c) REST (Representational State Transfer) : REST [Fielding 2000] es un enfoque con un estilo de operación muy limitado, en el que los clientes usan URL y operaciones HTTP: GET, PUT, DELETE y POST para manipular recursos representados en XML.

Tema 2 Cite:

Propiedades de las transacciones

1. Atomicidad
2. Consistencia
3. Aislamiento
4. Persistencia

Protocolos y elementos involucrados en el desarrollo e implementación de Web Services

1. SOAP
2. WSDL
3. HTTP

Tema 3 Complete con la expresión adecuada:

a) En el protocolo de consumación en dos fases, la operación “puede Consumar ?” es llamada desde el **coordinador** a **participante** para **preguntar si se puede consumar una**

transacción.

b) El protocolo **SOAP (Simple Object Access Protocol)**define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos representados en formato **XML.**

c) Cuando una sustracción finaliza, hace una decisión independiente sobre si consumarse provisionalmente o abortar. Su decisión de **Abortar** es final.

d) En Java los métodos "synchronized" asegura que **la concurrencia de acceso a objetos y reservada para una transacción para así poder implementar la propiedad de atomicidad y almacenamiento.**

Tema 4 Resuelva

En lenguaje Java, escriba las clases necesarias para realizar una votación online según lo siguiente:

El servidor debe:

i. Retornar los candidatos habilitados, con un código de candidato.

ii. Recibir las votaciones de los ciudadanos.

iii. Debe retornar un objeto compuesto como resultado de cada método web.

El cliente debe:

i. Listar los candidatos habilitados

ii. Elegir el candidato

iii. Enviar su votación

iv. Para cada invocación, utilizar los atributos del objeto respuesta que indiquen el éxito o error de dicha invocación.

a) Explique la situación presentada en la segunda fila y su un solapamiento secuencialmente equivalente. Re-escriba y realice un solapamiento de operaciones secuencialmente equivalente para las transacciones T y U. La solución no debe ser completamente secuencial.

Transacción T:	Transacción U:

balance = b.getBalance();
b.setBalance(balance * 1,1);
a.extraer(balance/10);
	balance = b.getBalance();
	b.setBalance(balance * 1,1);
	c.extraer(balance/10);

balance = b.getBalance();
 	
b.setBalance(balance * 1,1);
 	
a.extraer(balance/10);
	balance = b.getBalance();
	
	b.setBalance(balane * 1,1);
	
	c.extraer(balance/10);

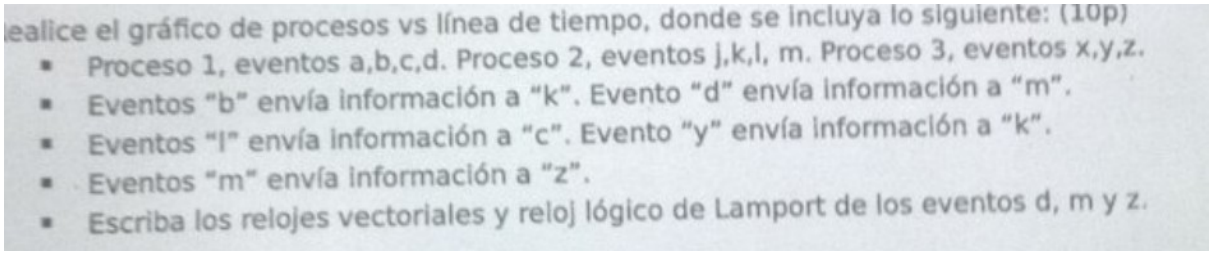
```
import java.jws.WebService;
import java.jws.WebMethod;
import java.jws.WebParam;
import java.jws.rs.Consumes
import java.jws.rs.PathParam
import java.jws.rs.Produces
import java.jws.rs.core.MediaType
import com.thoughtworks.xstream.XStream;



@WebService(serviceName="WsElecciones")
@Path("/elecciones")
public class Servidor{
        @Path(/canditatos)
        @WebMethod(operationsName = "candidatos")
        @GET
        @Produces(MediaType.APPLICATION_JSON))
        public ArrayList<Candidato> listar() {
                return obtenerListaCandidato();
        }

        @Path(/votar)
        @WebMethod(operationsName = "votar")
        @POST
        @Consumes(MediaType.APPLICATION_JSON)
        @Produces(MediaType.APPLICATION_JSON))
        public Response votar(Votacion s)  {
                JSONObject respuesta = 0;
                try{
                        respuesta = votar(s.candidato_id, s.votante_id);
                }catch(Exception e){
                }
                return respuesta;
        }

}

public class Cliente {
        public static void main (args[]) {
                Arraylist<Candidato> lista =  listar();
                Votacion v = votar(lista);
                enviarVoto(v);
        }

        public static ArrayList<Candidatos> listar (){
```

```
            try {
                    URL url = new URL("http://www.example.com/elecciones/candidatos");
                    HttpURLConnection connection = (HttpURLConnection) url.openConnection();
                    connection.setDoOutput(true);
                    connection.setRequestMethod("GET");
                    connection.setRequestProperty("Content-Type", "application/json");
                    InputStream in = new BufferedInputStream(urlConnection.getInputStream());
                    connection.disconnect();
            } catch(Exception e) {
                    throw new RuntimeException(e);
            }
            return JSONObject.toJSON(in).toObject();
        }


            public static void  enviarVoto (Votacion v ){
            try {
                    URL url = new URL("http://www.example.com/elecciones/candidatos");
                    HttpURLConnection connection = (HttpURLConnection) url.openConnection();
                    connection.setDoOutput(true);
                    connection.setRequestMethod("POST");
                    connection.setRequestProperty("Content-Type", "application/json");
                    OutputStreamWriter out = new    OutputStreamWriter(urlConnection.getOutputStream());
                    out.write(new JSONObject(v));
                    out.close();
                    connection.getResponseCode();
                     connection.disconnect();
            } catch(Exception e) {
                    throw new RuntimeException(e);
            }
        }
}

public class Candidato {
            long id;
            String nombre;
            public Candidato() {
            }
}

public class Votacion {
            long id;
            long candidato_id;
            long votante_id;
}
```

Primer Final 2017

Tema 2. Resuelva lo siguiente:

A. **Cite y explique dos annotations java utilizadas en desarrollo de Web Services**
B. **Considerar un servidor git alojado en un servidor con nombre DNS "git.alumnos.pol.una.py", y un proyecto llamado "supertarea". Escriba y explique los comandos git para:**
    a. **Realizar la clonación, el checkout del proyecto via SSH a la rama "developer".**
    b. **Enviar al repositorio los archivos "hola.java" y "hola.php" pasando primeramente por el Stage y HEAD.**
C. **Realizar el gráfico de procesos vs linea de tiempo, donde se incluya lo siguiente**


ealice el gráfico de procesos vs línea de tiempo, donde se incluya lo siguiente: (10p)
- Proceso 1, eventos a,b,c,d. Proceso 2, eventos j,k,l, m. Proceso 3, eventos x,y,z.
- Eventos "b" envía información a "k". Evento "d" envía información a "m".
- Eventos "l" envía información a "c". Evento "y" envía información a "k".
- Eventos "m" envía información a "z".
- Escriba los relojes vectoriales y reloj lógico de Lamport de los eventos d, m y z.

Tema 3. Complete con la expresión correcta

c) Un bloqueo se detecta cuando _____

d) La propiedad de **Aislamiento** de las transacciones establece que los efectos intermedios de una transacción no deben ser visibles para las demás transacciones.

e) El método de Cristian se utiliza para **Sincronización de relojes utilizando un servidor de tiempo**

Tema 4. Complete con V o F. Fundamente lo falso

c) El lenguaje de definición de servicios Web SOAP, está representado mediante un URL, cuyo contenido tiene el formato de un JSON.**(f) Tiene el formato XML.**

2do Final 2017

Tema 1: Defina

a) **Ritmo deriva del reloj:** Tasa de deriva del reloj: proporción en que el reloj de un computador difiere del reloj de referencia perfecto. Incluso, si los relojes de todos los computadores de un sistema distribuido se ponen en hora a la vez, a partir de un tiempo sus relojes variarán en una magnitud significativa a menos que se apliquen correcciones.

b) **Interbloqueo fantasma:** Un interbloqueo que se «detecta» pero que realmente no lo es se conoce como interbloqueo fantasma. En la detección de interbloqueos distribuidos, la información acerca de las relaciones espera por entre las transacciones se transmite de un servidor a otro. Si existe un interbloqueo, la información necesaria se recogerá, al final, en algún lugar y se detectará el ciclo. Ya que este procedimiento tarda un cierto tiempo, existe la posibilidad de tanto, una de las transacciones que mantiene un bloqueo se haya liberado, en cuyo caso ya no existe.

Tema 2: Cite y explique brevemente

a) **2 Operaciones dentro del protocolo de consumación en dos fases:**

puedeConsumar?(trans) → *Sí / No*
Llamada desde el coordinador al participante para preguntar si puede consumar una transacción. El participante responde con su voto.

Consuma(trans)
Llamada desde el coordinador al participante para decirle que consume su parte de una transacción.

Aborta(trans)
Llamada desde el coordinador al participante para decirle que aborte su parte de una transacción.

heConsumado(trans, participante)
Llamada desde el participante al coordinador para confirmar que ha consumado la transacción.

dameDecisión(trans) → *Sí / No*
Llamada desde el participante al coordinador para preguntar por la decisión sobre una transacción tras haber votado *Sí* aunque no ha obtenido respuesta tras cierto tiempo. Se utiliza para recuperarse de la caída de un servidor o de mensajes con retraso.

Tema 3. Resuelva lo siguiente

a) **En relación a los "estados globales", ejemplifique con un gráfico un corte consistente y un corte inconsistente. Posteriormente explique cada uno de los cortes.----> no entra esta parte**

b) **Escriba los relojes vectoriales correspondientes a los eventos de la Figura 1.**

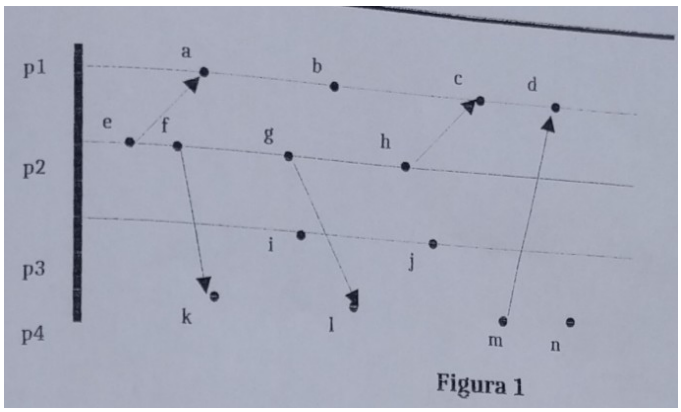


Figura 1

Tema 4. Complete con V o F. Justifique lo falso

a) El algoritmo de chandy y lamport establece que para sincronizar un reloj de un computador este debe colocar su tiempo a:"t +Tround/2". Siendo t el tiempo del computador con reloj de referencia y Tround el tiempo que tomo obtener el mensaje que contenía el tiempo t.**(F) el método de Cristian establece esa fórmula**

b) El método Cristian se utiliza para detectar interbloqueos en transacciones que gestionan los recursos**(F) Se utiliza para la sincronización de relojes utilizando un servidor de tiempo.**

c) La anotación @webservice se utiliza para indicar que el método de java referenciado será un web service**(V)**

CAP5

EVENTO: ocurrencia de una unica accion qué realiza un proceso a medida qué se ejecuta, puede ser una accion de comunicacion, o una transformacion de estados.

-La secuencia de sucesos de un proceso pi esta determinada por un unico orden qué es el siguiente

e->ie' si y solamente si e ocurre antes qué e'.

historial de un proceso pi: la secuencia de eventos de un unico proceso pi en un orden definido

RELOJ): un computador dispone de su propio reloj fisico

Un reloj es un dispositivo electronico, que cuenta las oscilaciones qué ocurre en un un cristal a una frecuencia definida. Normalmente esta cuenta se divide y el resultado se almacena en un registro contador.