Examen Teórico 2

1.- Las siglas RPC significan “Remote Procedure Call” (Llamada a procedimiento remoto). Se trata de una arquitectura mediante la cual, el programador, puede realizar llamadas a funciones remotas (que se encuentran programadas en otras máquinas conectadas a la Red) de igual modo que si se tratase de una función local.

2.- El programador invoca a una función que no se encuentra en la misma máquina sino que está programada en otra máquina diferente, la cual ha de estar conectada a la red, como es lógico.

Toda la complejidad que entraña el envío y recepción de los datos pertinentes a través de la red, así como la traducción del espacio de direcciones de memoria de un ordenador a otro (lo cual presenta el principal problema) es algo de lo que se encarga el software y, por lo tanto, es transparente para el programador.

3.- Sin embargo, este paradigma presenta algunas desventajas que menciono a continuación:

· Al tener que lidiar el software con toda la complejidad antes mencionada (comunicación a través de la red y direcciones de memoria, sobre todo), dicho software resulta frágil.

· Las llamadas remotas no ofrecen la fiabilidad de las llamadas a funciones locales. A menudo fallan con demasiada frecuencia.

· También presentan un problema las altas latencias que puede haber entre las máquinas.

· Las tecnologías RPC son muy heterogéneas.

· Fuerte acople con el lenguaje de programación. Es necesario que cliente y servidor estén escritos en el mismo lenguaje, debido a lo mencionado en el punto anterior. La implementación de las RPC difieren para distintos lenguajes. Por ejemplo, en Java se usa la “Java RMI” (Remote Method Invocation de Java).

· Mala integración con Cortafuegos.

4.- Los Web Services presentan varias ventajas sobres las RPC, que menciono a continuación:

· Al basarse en los protocolo **Estándar** de comunicación HTTP, y XML para el envío de datos, no existe la heterogeneidad que presentaban las RPC, lo cual simplifica notablemente la arquitectura.

· Se reduce significativamente el fuerte acoplamiento con el lenguaje de programación y el Sistema Operativo.

· Su integración con cortafuegos resulta más sencilla.

5.- SOAP es un protocolo que permite la comunicación entre máquinas mediante llamadas a métodos. Fue desarrollado en 1998 con el respaldo de Microsoft. De hecho, al ser privativo, nació un protocolo similar pero de código abierto denominado “XML-RPC”.

SOAP, normalmente, emplea HTTP, aunque también soporta otros protocolos como SMTP o RSS.

Su funcionamiento es similar al de una RPC pero usando el protocolo HTTP, que es estándar. Además, para el intercambio de documentos, se utiliza XML, que también es estándar.

Es muy importante el concepto de utilizar protocolos y procedimiento estándar, porque es la principal razón del desacoplo con el lenguaje de programación, además, de que simplifica en gran medida el desarrollo de las aplicaciones. En esto se basan los Web Services, como mencioné anteriormente.

Además, para garantizar la estabilidad y la buena integración entre aplicaciones SOAP, este protocolo especifica cómo han de codificarse las cabeceras HTTP, así como los documentos XML.

6.- Un mensaje SOAP es un documento XML que contiene:

· Un sobre (envelop) que lo identifica como mensaje SOAP.
· Opcionalmente, una cabecera.
· El cuerpo, con la información sobre las llamadas y las respuestas.
· Opcionalmente, un elemento “fault” con información sobre errores.

7.- Las siglas REST significan “Representation State Transfer”.

8.- Características REST:
· Ha de seguir la estructura Cliente-Servidor.
· Sin estado.
· Admitir el uso de cachés.
· Ha de ser un sistema basado en capas.
· Opcionalmente, soportar la generación de código bajo demanda.
· Uso de interfaces uniformes.

9.- El servidor NO almacena contexto acerca de las peticiones del cliente. Cada petición del cliente contiene toda la información necesaria para que el servidor responda.

La sesión y el estado del cliente SÓLO se guardan en el cliente.

Todas estas cuestiones, hacen que se simplifique notablemente la lógica del servidor. Además, garantiza que no se llegará a estados inconsistentes y facilita el escalado de la arquitectura.

10.- Que REST admita un sistema basado en capas significa que entre cliente y servidor, en la red, puede hacer Proxys, cachés (de http, por ejemplo) o distintos Gateways, de forma totalmente transparente a la comunicación y a la arquitectura (tanto para el cliente como para el servidor).

11.- Permite **sistemas débilmente acoplados**. Un sistema débilmente acoplado se trata de un sistema distribuido basado en la compartición de recursos entre distintas máquinas y, quizá también, por distintos Sistemas Operativos, comunicados entre sí a través de la red, para proporcionar un servicio.

12.-

1º Identificación de recursos: Los recursos individuales se identifican en las peticiones, normalmente mediante URI's. En necesario tener en cuenta que el recurso y la representación del mismo son cosas distintas. Por ejemplo, un recurso como puede ser un fichero, puede representarse en formato XML, JSON, etc.

2º Manipulación de recursos mediante representaciones: Como he mencionado anteriormente, es necesario diferenciar entre el recurso y la representación del mismo. Sin embargo, no hay una URI distinta para cada formato de un mismo recurso. Un recurso sólo puede tener una URI asociada, otra cosa es cómo se represente.

3º Mensajes autodescriptivos: Cada mensaje contiene toda la información necesaria para ser procesado y contestado por el servidor.

4º Hipermedia como motor del estado de la aplicación: Hace alusión a que las transiciones entre unos estados y otros, en el cliente, se especifican en un documento HTML que el servidor devuelve a éste. Dado el estado actual del cliente y el HTML que recibe del servidor, descubre a qué estados puede pasar. Al especificarse a través de un documento HTML, no es necesaria la utilización de instrucciones adicionales.

13.- Las siglas ROA significan "Resource-Oriented Architecture".

14.- Se dice que una arquitectura es RESTful cuando cumple rigurosamente con las especificaciones de la arquitectura REST. En la mayoría de los casos, se denominan RESTful arquitecturas que obedecen a las especificaciones ROA (un tipo de arquitectura REST concreta).

15.- Pautas para hacer servicios RESTfull según Richardson y Ruby:

· La arquitectura ha de estar basada en recursos.

· Todo recurso tiene que poseer una URI asociada.

· Las aplicaciones han de ser direccionables, es decir, que todos sus datos relevantes estén disponibles como recursos.

· Las aplicaciones deben usar una interfaz uniforme **concreta** (En el caso de REST no lo era): GET, PUT, DELETE.

· El servidor no presenta estado/s.

16.- AJAX (Asynchronous JavaScript And XML) es un conjunto de técnicas para enviar información, de manera asíncrona, entre un servidor web y el cliente. El modelo de AJAX actual fue creado en 2004 por Google.

Lo que AJAX proporciona es evitar que el usuario tenga que pulsar “enviar” para llevar a cabo un envío de datos al servidor, sino que éste reciba la información continuamente, lo que ofrece al usuario una experiencia similar a la de una aplicación de escritorio.

Procedo a explicar su funcionamiento:

En primer lugar, el usuario solicita una URL desde el navegador, a lo que el servidor web responde devolviendo la página correspondiente, la cual contiene un **script**. Por lo tanto, el navegador presenta la página al usuario y ejecuta el script (por lo que este script se ejecuta en la máquina cliente).

La función de dicho script es hacer peticiones HTTP asíncronas, es decir, sin que el usuario intervenga. Además, este script, recibe las respuestas HTTP del servidor y va modificando el documento HTML que se presenta al usuario (también de manera asíncrona).

Estos Scripts que se ejecutan en segundo plano en la máquina cliente son programas escritos en lenguaje JavaScript