1. [2,5 puntos]

a) Explica el paradigma de computación distribuida denominado MOM y sus tipos (1).

El MOM (Middlewire Orientado a Mensajes) es una evolución arquitectónica del *paso de mensajes*. Se caracteriza por un mayor **desacoplamiento, a sincronización** (no necesita respuesta inmediata) y la introducción de un **intermediario** en el proceso de comunicación.

Participan tres elementos: el **emisor**, que emite el mensaje; el proceso **intermedio**, que es el que se encarga de almacenar los mensajes enviados por el emisor, que tiene implementados ya características dependiendo del objetivo final del sistema; y un **receptor**, que se conecta al proceso *intermediario* para obtener los mensajes.

De éstos existen dos tipos:

Modelo punto a punto, el cual se caracteriza por que el mensaje es enviado por un único emisor y recibido únicamente por un receptor. En el proceso intermediario, se borra una vez que se ha enviado al receptor.

Modelo publicación/suscripción: Un emisor publica un mensaje en el *intermediario* y es procesado por todos los receptores que se hayan suscrito al proceso *intermediario.*

b) Enumera de forma general qué diferencias y coincidencias hay entre MOM y SOA (0,5).

Coincidencias:

1. Ambos son modelos **arquitectónicos**.
2. **Composición**: tanto en MOM como en SOA, un nodo puede ser emisor/receptor.

Diferencias:

1. **Asincronismo**: Las diferencias son que **MOM** se trata de una comunicación asíncrona y **SOA** es **petición/respuesta**.
2. **Desacoplamiento: SOA** sólo **desacopla** la **localización**, mientras que **MOM desacopla todo**, es decir, en ***MOM todo pasa por el intermediario*** (En *SOA una vez se ha descubierto el servicio ya son los nodos los que se comunican entre ellos*)
3. **Interoperabilidad**: **SOA** es **interoperable**, pero en **MOM** todo ha de estar en el **mismo lenguaje**.
4. **Localización:** en **MOM** en todo momento debemos saber a qué receptor vamos a enviar el mensaje. Sin embargo en **SOA** es el propio sistema *intermediario,* el que nos dice a ***qué*** cliente enviarle el mensaje.

c) Compara las funcionalidades del intermediario en cada una de ellas (0,5).

En MOM se encarga de almacenar el mensaje que ha emitido el emisor, hasta que el receptor esté disponible o envíe una petición al *intermediario* para ello. Una vez enviado, borra el mensaje.

En SOA, se encarga de comunicar **una vez** el emisor con el receptor. De ésta manera, sólo desacoplamos la comunicación una vez y aumenta la independencia del emisor y receptor con el intermediario. Nos provee de un mecanismo de **descubrimiento.**

d) ¿Qué mecanismo de comunicación distribuida (RPC, RMI, ORB, Servicios Web) utilizarías para implementar MOM. Justifica tu respuesta (0,5).

Dado que MOM requiere de un intermediario para la gestión de mensajes, una comunicación adecuada sería **ORB** ya que es una agente intermediario para la gestión de objetos, y es común a todos los nodos. Sin embargo **Servicios Web** abarca muchas herramientas y componentes que nos permitiría también implementar un sistema **MOM**.

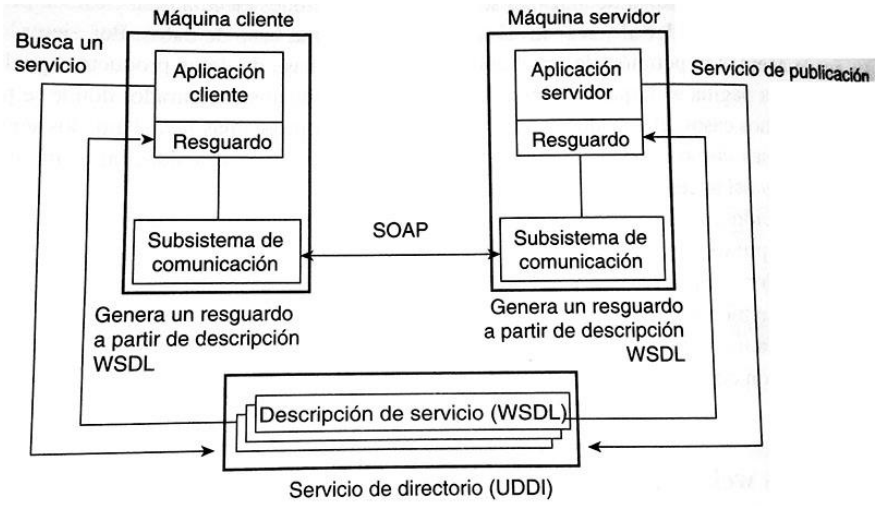
2. Explica y describe la funcionalidad dentro de la tecnología de Servicios Web de WSDL, UDDI y SOAP. ¿Qué relación hay entre WSDL, UDDI y SOAP? [2 puntos]

Un ***servicio web*** no es más que un servicio tradicional puesto a disposición de todo el mundo en internet. Lo que hace especial al servicio web es que se apega a un conjunto de estándares que permiten ser ***descubierto*** y ***accedido*** a través de la red por **aplicaciones** cliente que también se apegan a dichos estándares.

UDDI: Un servicio web está formado por un servidor de directorio que guarda **descripciones de servicios.** Este servicio se adhiere al estándar **DESCRIPCIÓN, DESCUBRIMIENTO E INTEGRACIÓN UNIVERSALES (UDDI),** y es una 'base de datos' que contiene descripciones de servicios. Esto permite descubrir con quién comunicarse y dónde. Los registros que se almacenen aquí, apuntan a una hoja **WSDL.**

WSDL: Los servicios anteriormente nombrados, se describen por medio del **LENGUAJE DE DEFINICIÓN DE SERVICIOS WEB (WSDL)**, el cual es un lenguaje formal para soportar la comunicación basada en RPC. Un WSDL contiene definiciones de las interfaces provistas por un servicio (especificación, tipos de datos, lógica…). Un aspecto importante de éstos es que puede ser transformado automáticamente en Resguardos del lado del cliente y en resguardos del lado del servidor.

SOAP: Es la especificación de como ocurre la **comunicación.** Para ello se utiliza el **PROTOCOLO DE ACCESO A UN OBJETO SIMPLE (SOAP)**, que es una estructura donde una gran parte de la comunicación entre dos procesos puede ser estandarizada. Y actúa sobre HTTP.



Cómo podemos comprobar en la imagen, **UDDI** es el servidor de directorio que contiene los documentos **WSDL**, que una vez se han generado resguardos a partir de éstos, se utiliza **SOAP** para la comunicación entre cliente y servidor.

b) Explica y describe las partes de un documento WSDL.

Un documento WSDL se compone por:

- **Definitions:** contiene la definición de uno o más servicios.
- **Message:** una definición abstracta de datos que están siendo transmitidos.
- **PortType:** conjunto de operaciones abstractas. Hace referencia a un mensaje de entrada y otro de salida.
- **Binding:** especifica un protocolo concreto y las especificaciones del formato de los datos y los mensajes definidos por un portType.
- **Service:** para unir un conjunto de puertos relacionados
- **Documentation:** documentación relacionada.

3. [2 puntos]

a) Explica qué es un servicio de nombres y qué es un servicio de directorios y cuáles son sus principales características.

El **servicio de nombres**, es un servicio que provee de un traductor de identificador de recursos a nombre legible por el ser humano, es decir es una base de datos con registros que contienen una relación ***nombre lógico*/*atributo***. Puede usarse para referenciar recursos o usuarios, mediante objetos, archivos, máquinas...

Es independiente de la **localización,** lo que nos da una mayor flexibilidad a la hora de realizar peticiones al servicio. Y es independiente de cualquier otro servicio, por tanto, nos da una mayor **escalabilidad** al no depender de otros.

Es una evolución, también, del paradigma **cliente/servidor**

El servicio simplemente nos responde con la localización del recurso, *no con el recurso* en sí mismo. Además, existen ***convenciones*** de ***nombrado*** para la difusión de estas técnicas.

Finalmente, se debe añadir que se caracteriza por su **interoperabilidad**, ya que es visible y utilizado por casi todos los sistemas.

Un **servidor de directorio** tiene información almacenada acerca de objetos relacionados, como recursos de red, usuarios o departamentos. Si el servidor de nombres actúa como un

análogo a las *páginas blancas* de toda la vida, el servidor de directorio se equipararía a unas *páginas amarillas,* y corresponde a un **ámbito más general,** ya que desarrolla un sistema para traducir cualquier servicio.

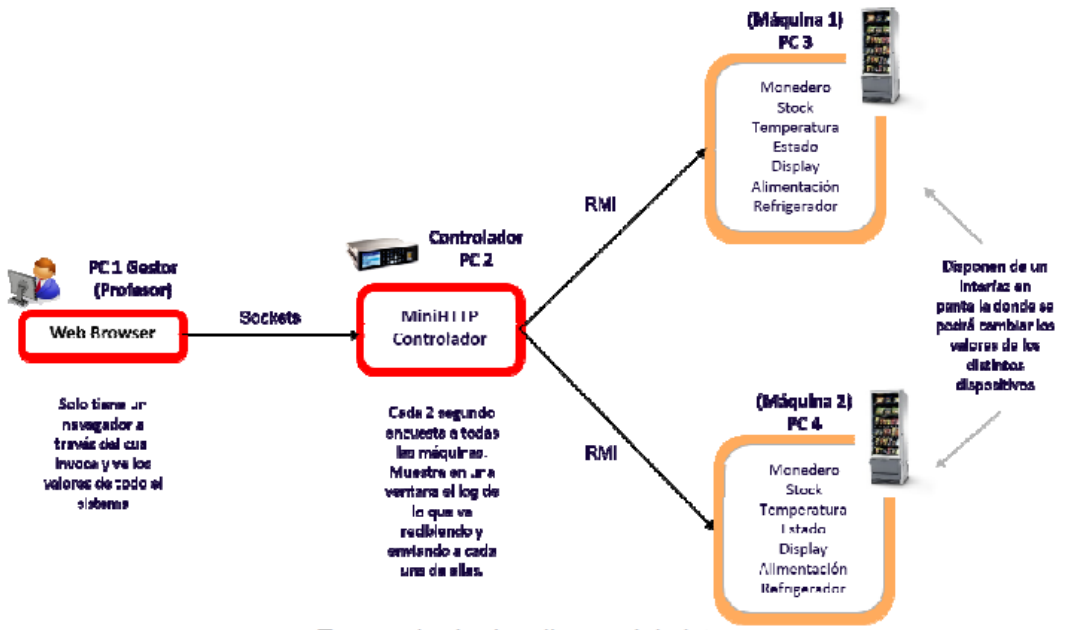
b) De forma breve, clasifica y separa DNS, LDAP y UDDI en servicios de nombre o directorio y si son de propósito general o específico. Justifica la respuesta.

DNS pertenece a **servidor de nombre**. Y LDAP, es un protocolo ligero para el acceso a servicio de directorio, así que lo clasificaremos, junto a UDDI en **servicio de directorio.** DNS y LDAP son de ***propósito general***, ya que se realiza una tarea con cada uno de ellos, que abarca un resultado final. Sin embargo UDDI, pertenece a un conjunto de mecanismos, para la correcta comunicación con un servicio de directorio, así que se deduce que es de ***propósito específico***.

4. Respecto a la práctica no guiada de sockets y RMI que habéis realizado, explica esquemáticamente la arquitectura desde el navegador hasta el controlador. ¿Qué importancia tiene el controlador? [1 punto]

El controlador es el encargado de gestionar la comunicación de las máquinas, y de proveer al servidor http, de la información necesaria para que sea reflejada en el navegador. Controla tanto las peticiones cada dos segundos a las máquinas de Vending (mediante protocolo RMI) y responde a las peticiones que le llegan desde el servidor HTTP con la información correspondiente.

la incidencia.



Escenario de despliegue del sistema

5. Explica el modelo arquitectónico denominado middleware orientado a mensajes (MOM). Enumera y describe sus elementos. ¿Cuáles son las principales similitudes y diferencias entre MOM y el modelo cliente-servidor convencional? [2 puntos]

El MOM (Middlewire Orientado a Mensajes) es una evolución arquitectónica del *paso de mensajes*. Se caracteriza por un mayor **desacoplamiento, a sincronización** (no necesita respuesta inmediata) y la introducción de un **intermediario** en el proceso de comunicación. Gracias a esto último, no es necesario que tanto el emisor como el receptor estén conectados; cuando el receptor se conecte, aunque el emisor no esté conectado, éste recibirá el mensaje si lo pide.

Participan tres elementos: el **emisor**, que emite el mensaje; el proceso **intermedio**, que es el que se encarga de almacenar los mensajes enviados por el emisor, que tiene implementados ya características dependiendo del objetivo final del sistema; y un **receptor**, que se conecta al proceso *intermediario* para obtener los mensajes.

La principal diferencia con un **cliente-servidor** convencional, es la aparición del nodo intermediario, que es el que almacenará el mensaje u objeto, hasta que el receptor se lo solicite al intermediario. (Similitud de cliente-servidor con MOM punto a punto).

6. Uno de los principales problemas en la comunicación entre entidades de un sistema distribuido es la representación de la información. Enumera y describe el funcionamiento de las tres posibles tácticas que tratan de resolver esta problemática [2 puntos]

La representación de la información para continuar con la heterogeneidad de los datos, se puede resolver de tres posibles maneras:

- **Emisor se adapta al receptor** (menos usado): tiene una gran latencia por preguntar y traducir.
- **Receptor se adapta al emisor** (bastante utilizado): el mensaje inicial contiene información para que el receptor pueda interpretar la información. Por ejemplo, el protocolo del correo electrónico.
- **Representación externa común** (el más utilizado): Seguro que lo entenderá porque ambos hablan ese lenguaje. Por ejemplo, Corba, servicios web con XML o RMI para Java.

7. Comenta brevemente el modelo CGI indicando sus ventajas y desventajas. Justifica su aparición partiendo del modelo Web básico [2 puntos]

El **CGI** aparece por la necesidad de ampliar el modelo HTTP a contenido web generado en tiempo de ejecución (respuesta dinámica). Dado que el modelo web básico no creaba nada en tiempo de ejecución, las necesidades de la web, requerían de un sistema que interactuara de manera dinámica con el usuario. Por tanto, nace como necesidad a responder de manera **dinámica** a las peticiones de los navegadores web.

Ventajas:

Ofrece capacidad de **respuesta dinámica**

Libertad de elección en el lenguaje de programación

Desventajas:

No existe relación entre programa CGI y servidor web, por tanto no existe **control** sobre la ejecución y respuesta.

Nueva instanciación del programa CGI por cada solicitud, lo que produce una **sobrecarga** en los sistemas.

8. Justifica la gestión distribuida de LDAP, define las ventajas sobre otros sistemas de información tradicionales y enumera y describe los motivos que pueden llevar a esta gestión distribuida. ¿Qué técnica es utilizada para relacionar las diferentes partes del espacio de nombres cuando es distribuido? [2 puntos]

La gestión distribuida de **LDAP** se produce porque de esa forma puede dividirse en subárboles por motivos de rendimiento, localización y por cuestiones administrativas.

Otros **protocolos** de servidor de directorio, como puede ser **X500** es mucho más pesado, y lo forma una serie de directorios locales para formar uno global. Es previo a **LDAP** y se usaba como páginas blancas (servidor de nombres). Además fue un intento de impulsar la construcción de un servicio de Directorio a nivel mundial, totalmente distribuido.

LDAP surge como alternativa a otros sistemas de información, para subsanar características de autenticación, **seguridad** y otras especificaciones técnicas. Además es una alternativa ligera a X500, ya que trabaja sobre TCP y tiene un conjunto de operaciones reducidas. Además la comunicación de este método es **asincrona**, por lo que nos permite una mayor flexibilidad a la hora de utilizarlo.

9. Explicar la arquitectura SOA

La arquitectura **SOA** se fundamentó en la abstracción a actividades de negocio denominadas *servicios.* Es una evolución del MOM, y se diferencia de ésta en el desacoplamiento único de la comunicación. El funcionamiento es el siguiente:
El **consumidor** envía una petición de recursos a un registro **intermediario**, y es este registro el encargado de enviarle al cliente la localización del recurso. Así, se conectan el consumidor y el **proveedor**, sin necesidad de establecer comunicación directa continua con el registro intermediario. Podríamos dar una analogía similar a las *páginas amarillas* convencionales.
Sus características principales son*: Localización y descubrimiento; interoperabilidad, composición, autonomía, reusabilidad, desacoplamiento y la ausencia de estado.*

10. Explicar los Servlets, sus características y compararlos con los CGI y las páginas dinámicas.

Los **servlets** *son* programas escritos en Java, se pueden invocar desde un cliente, de forma similar a como se invocan los programas CGI; se *ejecutan* en el seno del servidor Web, que actúa como un **contenedor de componentes**; *realizan* una determinada tarea; y generan una salida (página Web dinámica) que es recogida por el servidor Web y posteriormente reenviada al **navegador** que realizó la petición de este componente.
Teniendo encuentra que un ***servlet*** es un objeto software que se invoca externamente, representa un modelo mucho más cercano al *modelo CGI* que al *modelo de páginas activas*, donde este objeto se encuentra incrustado dentro de código HTML estático. Sin embargo, la diferencia fundamental con respecto al modelo CGI clásico se encuentra en que un servlet se ejecuta dentro del contexto, y por lo tanto del control, del servidor o, ahora, contenedor Web.