

INTRODUCCIÓN A LOS SISTEMAS DISTRIBUIDOS

Sistemas informáticos I

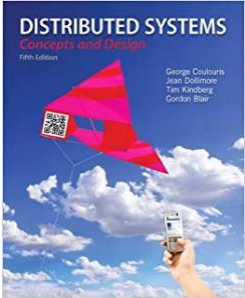
ÍNDICE

- Introducción a los sistemas distribuidos
- Arquitectura de los sistemas distribuidos
- Características de los sistemas cliente-servidor
- Arquitectura de las aplicaciones WWW
- Introducción a la computación en la nube

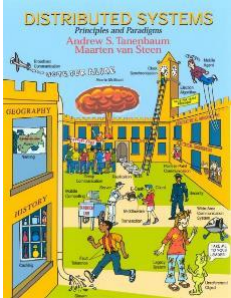


REFERENCIAS BIBLIOGRÁFICAS

COULOURIS, G., DOLLIMORE, J. y
KINDBERG, T., Sistemas distribuidos.
Conceptos y diseño, Addison-Wesley



TANENBAUM, A. y VAN STEEN, M.,
Distributed Systems. Principles and
paradigms, Prentice Hall



SISTEMAS DISTRIBUIDOS. DEFINICIONES

Un sistema distribuido es una colección de entidades independientes que cooperan entre sí para resolver un problema que no se puede resolver de forma individual → sistema complejo

En el contexto de la asignatura:

- **Sistema centralizado:** ordenador central y red de terminales sin capacidad de proceso

VS.

- **Sistema distribuido:**
 - Conjunto de elementos de proceso computacional autónomos,
 - no necesariamente homogéneos,
 - que están interconectados por una red de comunicaciones de cualquier tipo,
 - y que cooperan mediante el envío de mensajes para realizar las tareas que tienen asignadas



MOTIVACIÓN DE LOS SISTEMAS DISTRIBUIDOS

- Distribución inherente de algunas aplicaciones
- Compartición de recursos
- Acceso a recursos remotos
- Economía
 - Ley de Grosh (~70s): la potencia computacional de una CPU es proporcional al cuadrado de su precio
 - Con el paso de los años, la ley de Grosh deja de ser válida...
 - ... ni que decir tiene a día de hoy con la computación en la nube
- Incremento de la potencia computacional y la velocidad de cálculo
 - Procesamiento paralelo
 - Escalabilidad
- Flexibilidad y modularidad



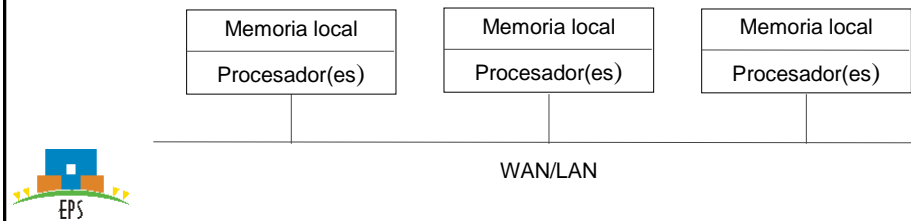
DESVENTAJAS DE LOS SISTEMAS DISTRIBUIDOS

- Aumenta la complejidad
- Las comunicaciones son fuente de problemas/error:
 - Pérdida de mensajes
 - Saturación
 - Latencia
- Seguridad
- Confidencialidad



TIPOS DE SISTEMAS DISTRIBUIDOS

- Existen diversas clasificaciones atendiendo a distintos criterios. Aquí vamos a distinguir dos clasificaciones principales
- Atendiendo a su grado de acoplamiento (HW):
 - **Fuertemente acoplados:** Procesadores que comparten memoria o buses de entrada/salida. Aplicaciones multiprocesador
 - **Débilmente acoplados:** Procesadores autónomos interconectados por sistemas de comunicaciones



TIPOS DE SISTEMAS DISTRIBUIDOS

- Atendiendo a su arquitectura software típicamente distinguimos dos tipos:
 - **Igual a igual** (peer to peer, p2p):
 - Sistema simétrico
 - Todos los procesos desempeñan tareas semejantes
 - Interactúan para realizar una actividad distribuida
 - **Cliente-servidor:**
 - Sistema asimétrico
 - Procesos clientes solicitan servicios
 - Procesos servidores los ejecutan y devuelven los resultados
- En este curso nos centraremos principalmente en sistemas cliente-servidor débilmente acoplados



¿QUÉ DISTRIBUIR EN UN SI?

- Lógica de proceso
- Funciones
- Datos
- Control
- ...



ARQUITECTURA DE LOS SISTEMAS DISTRIBUIDOS


- La organización lógica de los componentes de un sistema distribuido varía con cada aplicación, pero existen patrones que se repiten habitualmente
- Típicamente se consideran dos arquitecturas SW básicas para los sistemas distribuidos:
 - Igual a igual (peer to peer, p2p)
 - Cliente-Servidor
 - Históricamente es la arquitectura más importante
 - Continúa siendo la más ampliamente utilizada (o alguna de sus variantes)

Pero antes...




ARQUITECTURA SOFTWARE

- La arquitectura software de un SI define el sistema en términos de componentes computacionales e interacciones entre ellos
- Identifica:
 - Los componentes SW del sistema
 - La ubicación de cada componente en la red
 - Las interrelaciones entre componentes
 - La distribución de datos y tareas computacionales entre los nodos físicos de la red
- Evalúa el rendimiento, confiabilidad, escalabilidad y otras propiedades del sistema de software




Estructura y Topología



ARQUITECTURA SOFTWARE

- Componentes:
 - clientes
 - servidores
 - bases de datos
 - ...
- Interacciones:
 - llamadas a procedimientos
 - compartición de variables
 - protocolos cliente/servidor
 - protocolos de acceso a BB.DD.
 - ...



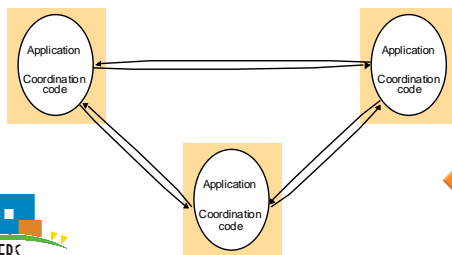
¿POR QUÉ ES IMPORTANTE LA ARQUITECTURA SOFTWARE DE UN SI?

- Forma la columna vertebral para construir un sistema de software
- Es en gran medida responsable de permitir o no ciertos atributos de calidad del sistema (p.ej., confiabilidad y rendimiento)
- Es un modelo abstracto reutilizable
 - Puede transferirse de un sistema a otro
 - Representa una inversión
- Representa un medio de comunicación y discusión entre participantes del proyecto



ARQUITECTURA DE LOS SISTEMAS DISTRIBUIDOS

- La organización lógica de los componentes de un sistema distribuido varía con cada aplicación, pero existen patrones que se repiten habitualmente
- Típicamente se consideran dos arquitecturas SW básicas para los sistemas distribuidos:
 - Igual a igual (peer to peer, p2p)
 - Cliente-Servidor
 - Históricamente es la arquitectura más importante
 - Continúa siendo la más ampliamente utilizada (o alguna de sus variantes)



Volvemos...

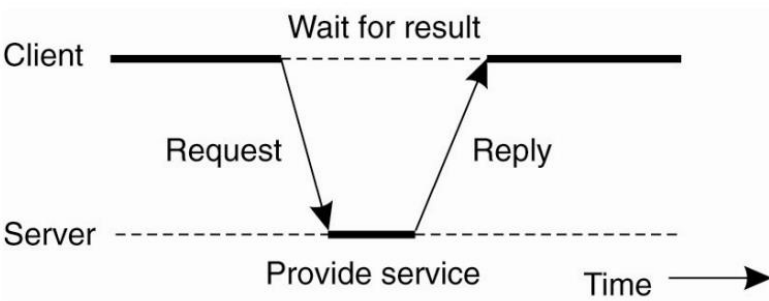


ARQUITECTURA CLIENTE-SERVIDOR

- Objetivo principal: proveer una arquitectura escalable
- Relación asimétrica en cada interacción entre componentes
- Servidor:
 - Pasivo
 - Espera peticiones
 - Cuando recibe una petición, la procesa y envía respuesta
 - Puede conservar o no el estado de la comunicación
- Cliente:
 - Activo
 - Envía peticiones
 - Espera hasta que llega la respuesta




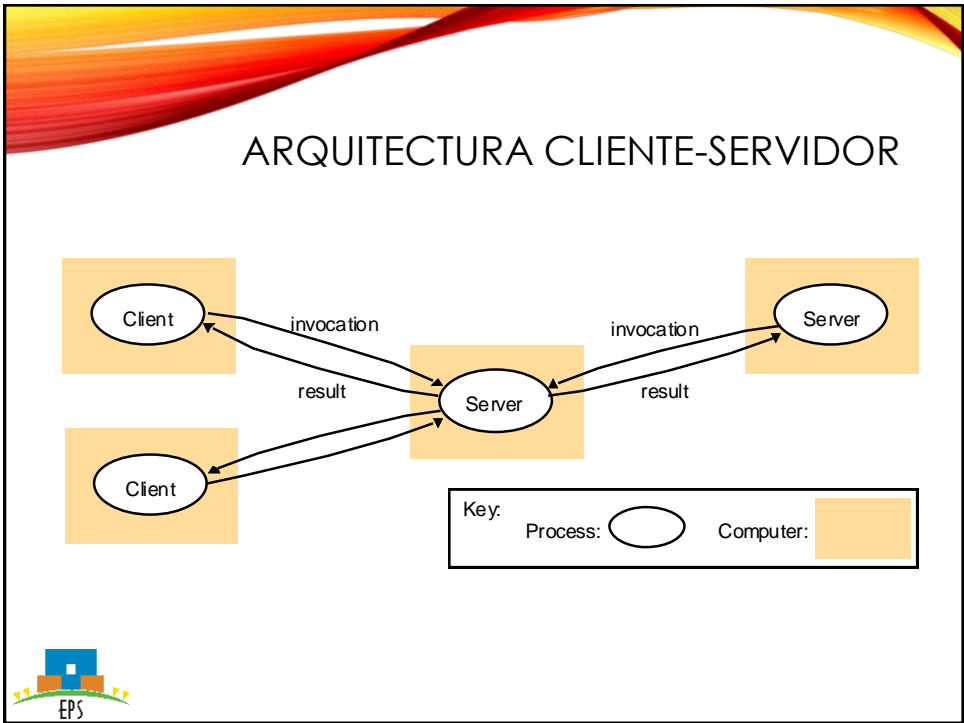
ARQUITECTURA CLIENTE-SERVIDOR



```
sequenceDiagram
    participant Client
    participant Server
    Note over Client: Wait for result
    Client->>Server: Request
    activate Server
    Note over Server: Provide service
    Server->>Client: Reply
    deactivate Server
    Note over Client: Wait for result
```

General interaction between a client and a server.





ARQUITECTURA CLIENTE-SERVIDOR

CARACTERÍSTICAS BÁSICAS

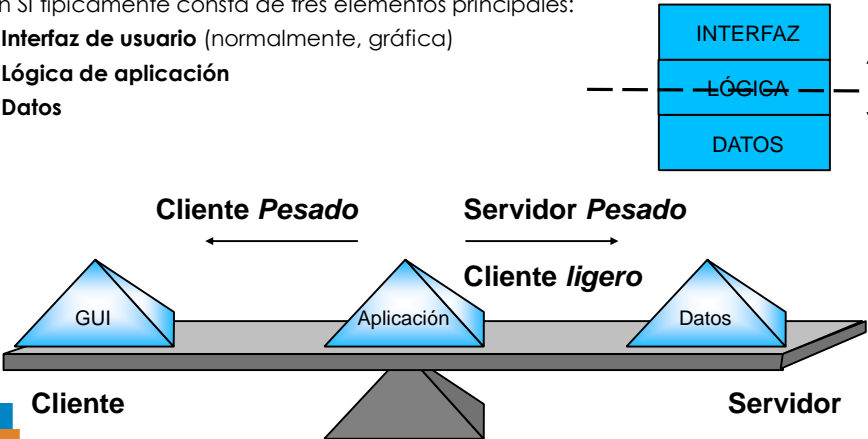
- **Sistemas débilmente acoplados.** Interacción basada en el envío de mensajes
- **Servicio.** Unidad básica de diseño. El servidor los proporciona y el cliente los utiliza
- **Encapsulamiento de servicios.** Los detalles de la implementación de un servicio son transparentes para el cliente
 - Lleva a las arquitecturas basadas en servicios (SOA)
- **Recursos compartidos.** Muchos clientes utilizan los mismos servidores y, a través de ellos, comparten recursos lógicos o físicos
- **Protocolos de aplicación asimétricos.** En su esquema básico, relación 1-n
 - El servidor es un cuello de botella → activo-pasivo y/o escalado horizontal
- **Transparencia.** El sistema aparece como una única unidad de proceso. Independencia de la plataforma HW y SW que se emplee
- **Escalabilidad,** horizontal (requiere balanceo de carga) y vertical
- Datos y programas centralizados en servidores facilitan su **integridad y mantenimiento**




CLASIFICACIÓN DE LOS SISTEMAS CLIENTE-SERVIDOR

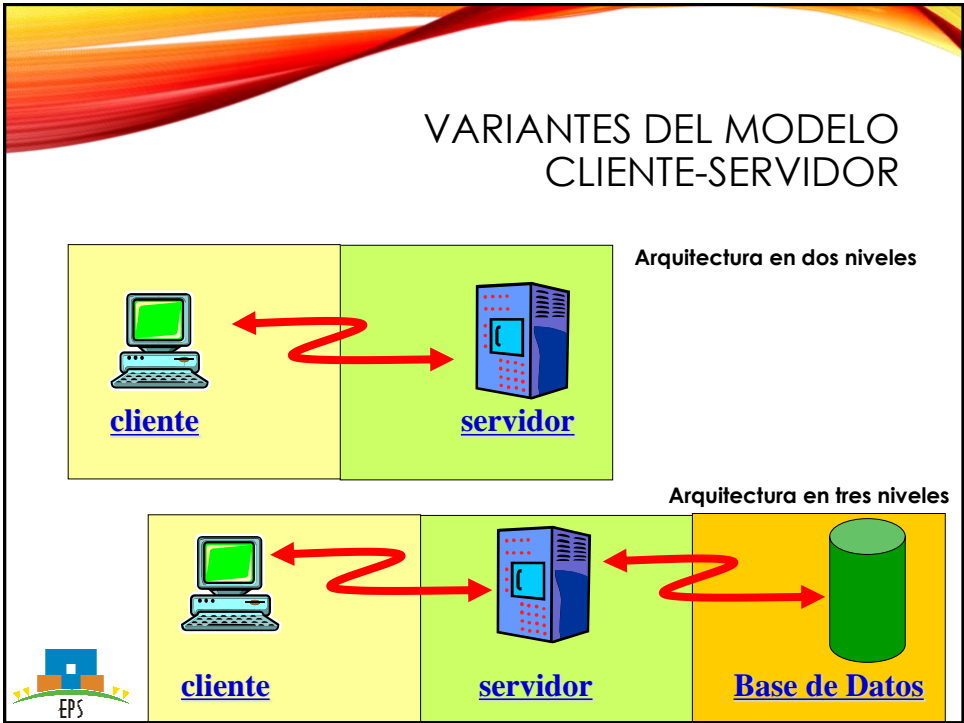
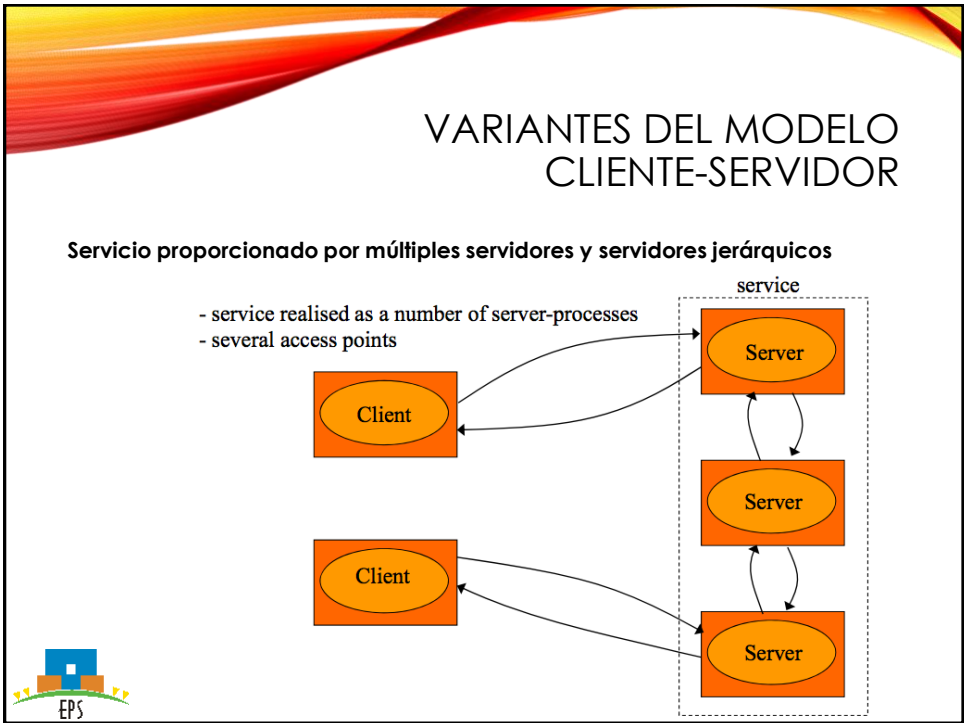
Un SI típicamente consta de tres elementos principales:

- **Interfaz de usuario** (normalmente, gráfica)
- **Lógica de aplicación**
- **Datos**



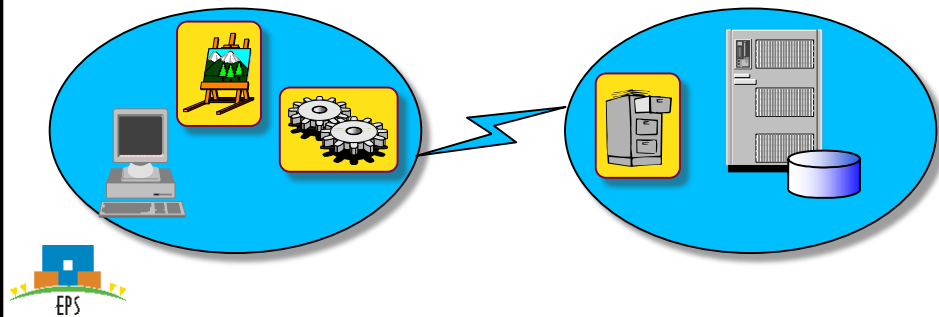
The diagram illustrates the distribution of system components (GUI, Aplicación, Datos) between the Client and Server, categorized as Client-heavy or Server-heavy. A horizontal bar represents the distribution, with a central pivot point. On the left, under 'Cliente', is a blue pyramid labeled 'GUI'. On the right, under 'Servidor', is a blue pyramid labeled 'Datos'. In the center, under 'Aplicación', is a blue pyramid. Above the bar, arrows indicate the distribution: 'Cliente Pesado' (left) and 'Servidor Pesado' (right). To the right of the bar, a vertical stack of three blue boxes represents the system components: 'INTERFAZ' (top), 'LÓGICA' (middle), and 'DATOS' (bottom). A double-headed vertical arrow is next to this stack. The 'LÓGICA' box is connected to the 'Aplicación' pyramid on the bar. The 'INTERFAZ' box is connected to the 'GUI' pyramid. The 'DATOS' box is connected to the 'Datos' pyramid.





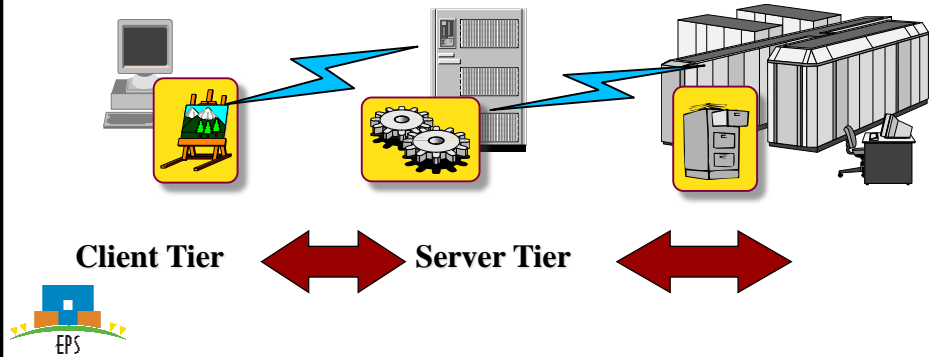
APLICACIONES CLIENTE-SERVIDOR DE 2 NIVELES (2-TIER)

- La lógica de la aplicación se ejecuta junto con la lógica de presentación (modelo cliente pesado)
- El servidor realiza acceso a datos
- Implementación sencilla para aplicaciones pequeñas
- Ampliación de la aplicación y migración a otros entornos compleja

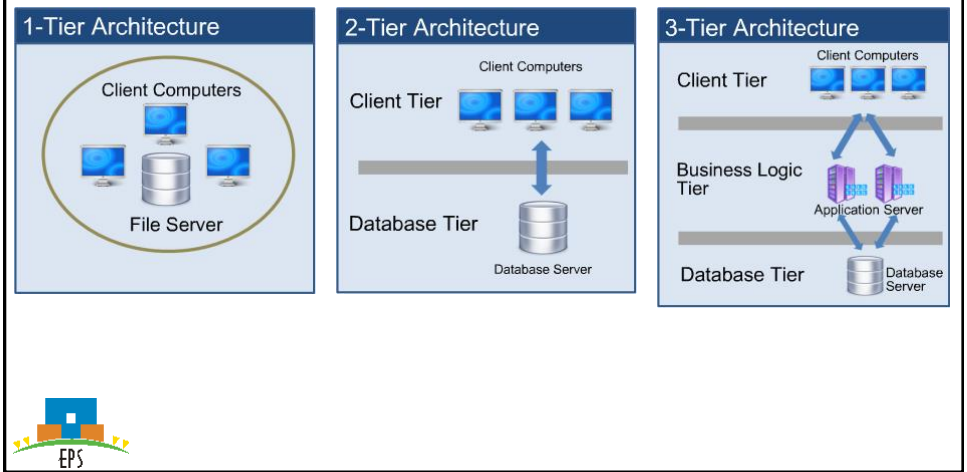


APLICACIONES CLIENTE-SERVIDOR DE 3 NIVELES (3-TIER)

- Lógica de aplicación y de presentación separadas (**modelo cliente ligero**)
- La lógica de la aplicación reside en cualquiera de los nodos de la red
- Aplicaciones más robustas y fácilmente escalables

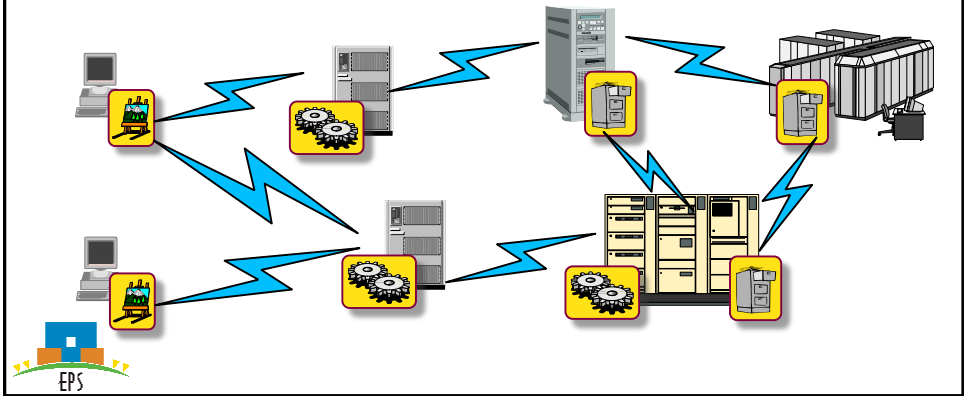


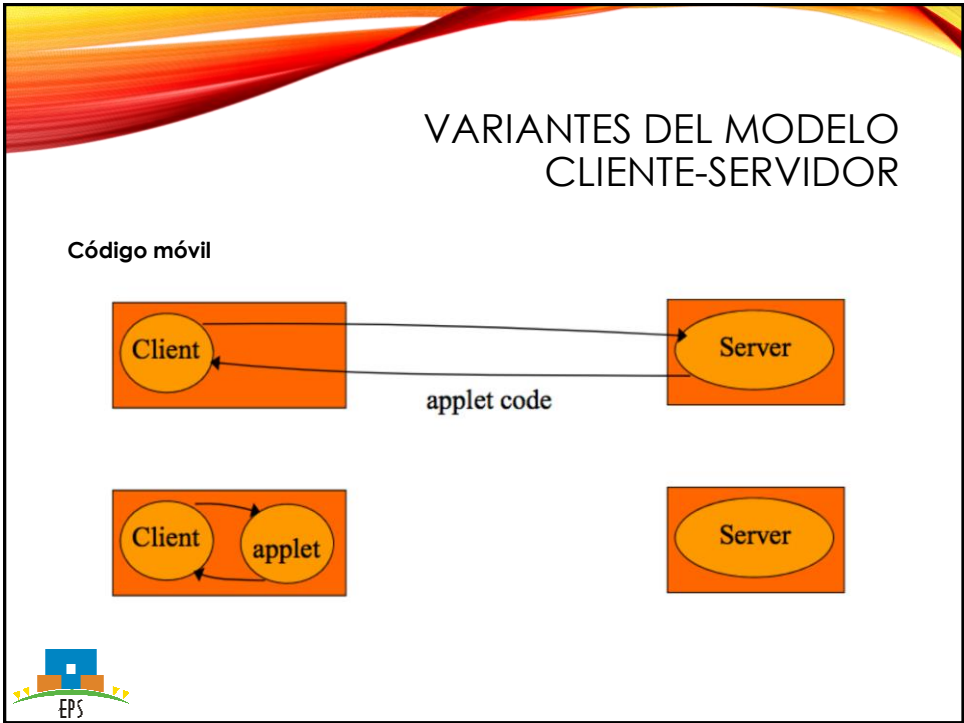
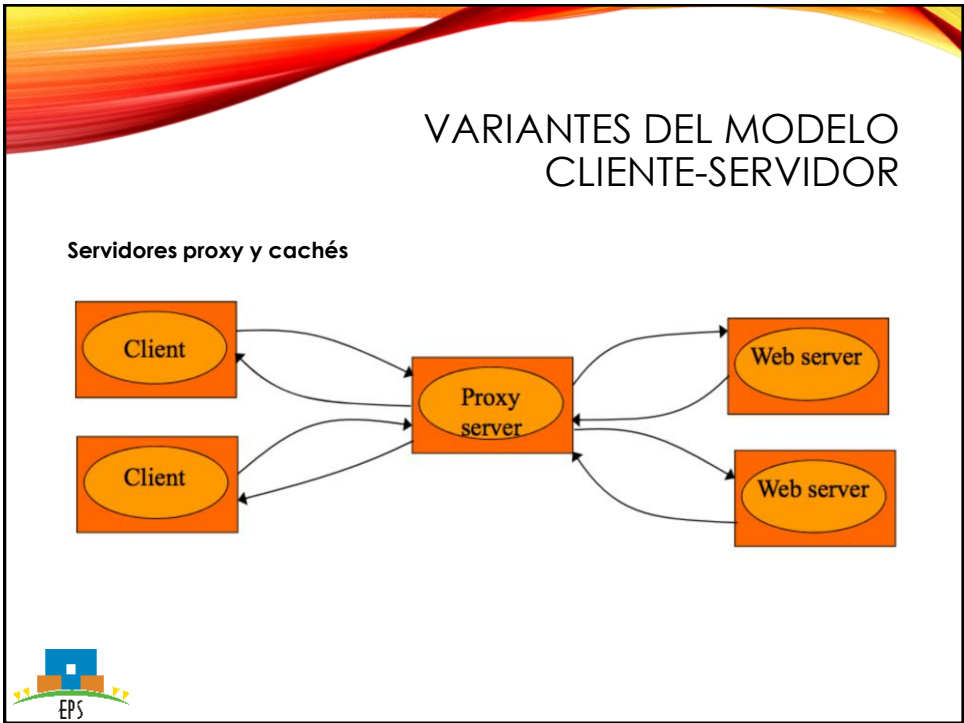
ARQUITECTURAS SOFTWARE EN CAPAS

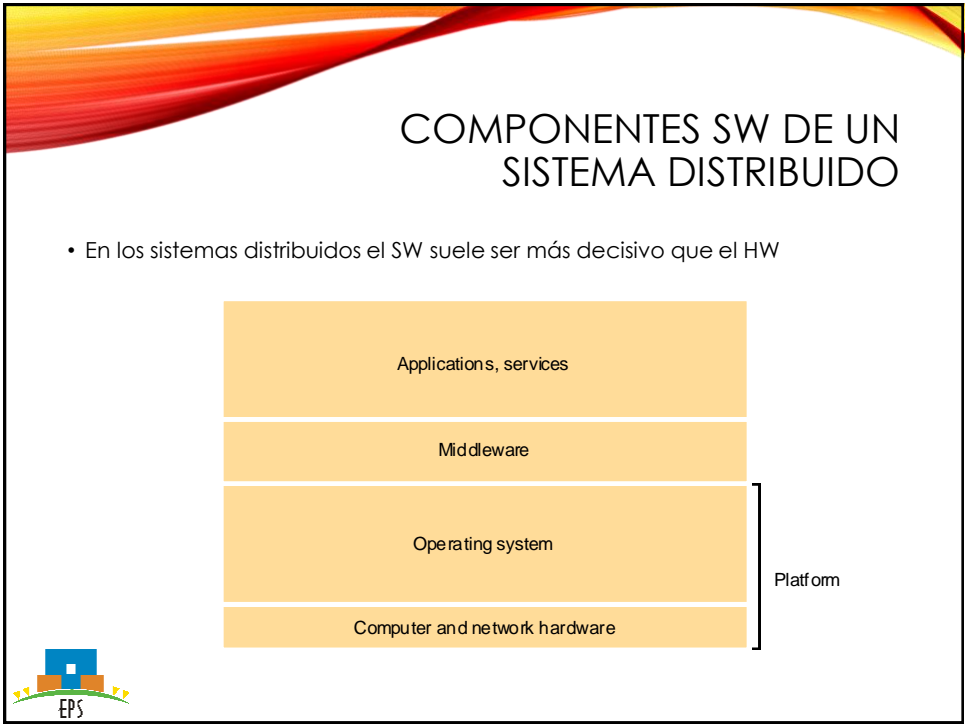
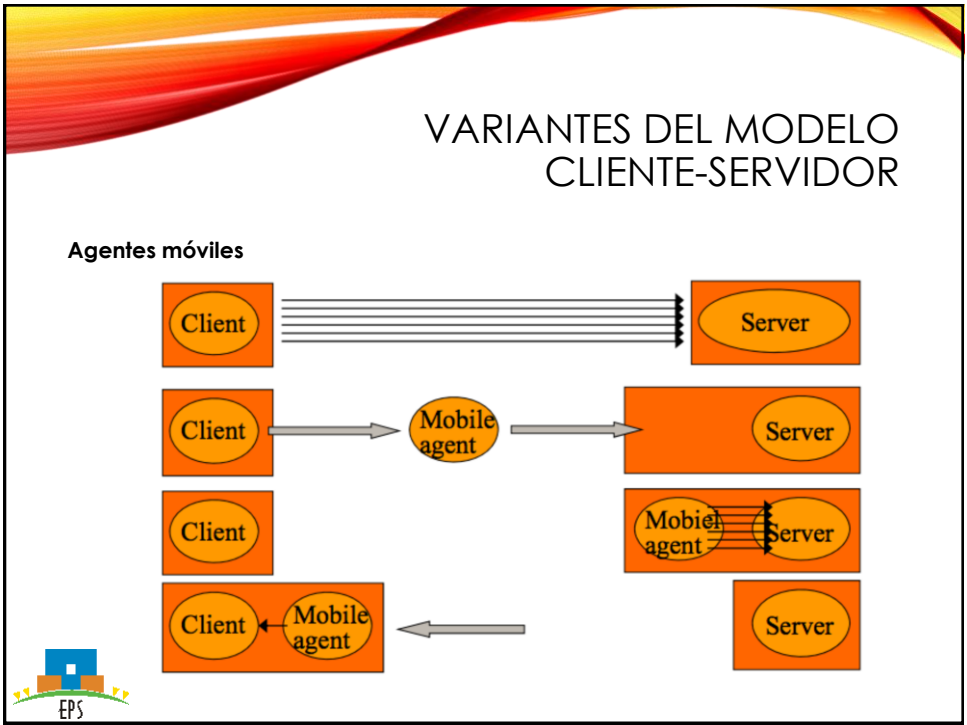


GENERALIZACIÓN: APLICACIONES EN N NIVELES (N-TIER)

- Múltiples niveles de clientes y servidores
- Representan nuevos modelos de arquitecturas de sistemas distribuidos basados en la arquitectura cliente-servidor

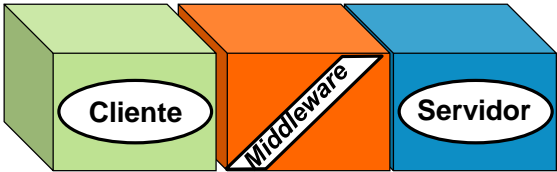






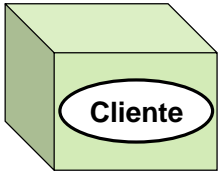
COMPONENTES SW DE UN SISTEMA CLIENTE-SERVIDOR

- Se suelen considerar tres módulos principales:
 - Cliente. Solicita servicios.
 - *Middleware*. Enlace entre cliente y servidor o entre servidores.
 - Servidor. Proporciona servicios.



CLIENTES

- Cliente es todo **proceso** que reclama servicios a otro
- No es equivalente a aplicación que interacciona con el usuario, aunque es el modelo de cliente más habitual
- Un servidor, a su vez, puede ser cliente de otros servidores
- La funcionalidad del proceso cliente marca la operativa de la aplicación (flujo de información o lógica de negocio)



SERVIDORES

- Servidor es todo **proceso** que proporciona un servicio a otros
 - Servidores de disco
 - Servidores de impresión
 - Servidores de comunicaciones
 - Servidores de presentación
 - Servidores de procesos
 - Servidores de bases de datos
 - Servidores de seguridad
 - ...
- Ambigüedad: Normalmente se llama "servidor" tanto a las aplicaciones o procesos que realizan un servicio como a los ordenadores que les sirven de soporte. En esta asignatura normalmente nos referiremos a los procesos de servicio

MIIDDLEWARE

- Se ejecuta tanto en el servidor y en el cliente
- Interfaz Cliente/Servidor o Servidor/Servidor
- Tres niveles:
 - Protocolo específico del servicio
 - Network Operating System, NOS.
 - Protocolo de transporte.

Específico Servicio	ODBC	TxRPC	Correo	ORB	HTTP
NOS	Directorio	Seguridad	Tiempo		
	RPC	MOM	OO	WS	API directa
Transporte	NetBIOS	TCP/IP	IPX/SPX	SNA	

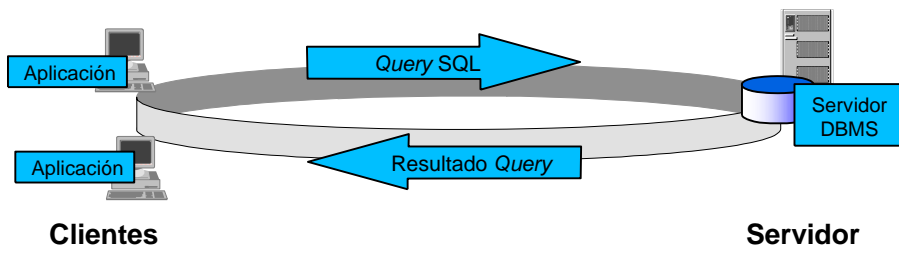
MODELOS DE SISTEMAS DISTRIBUIDOS

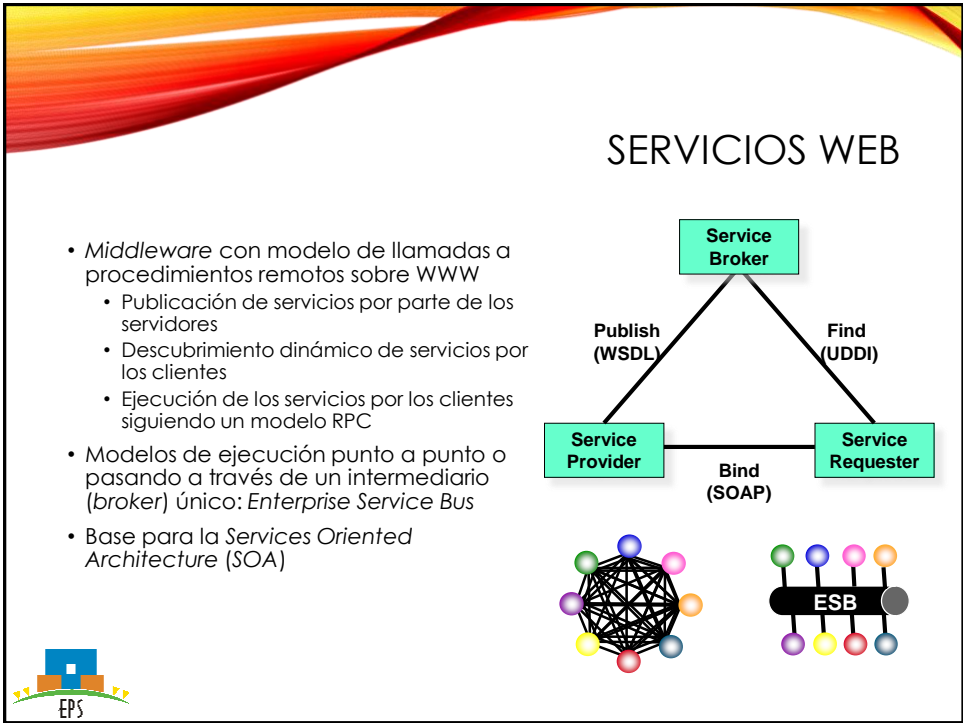
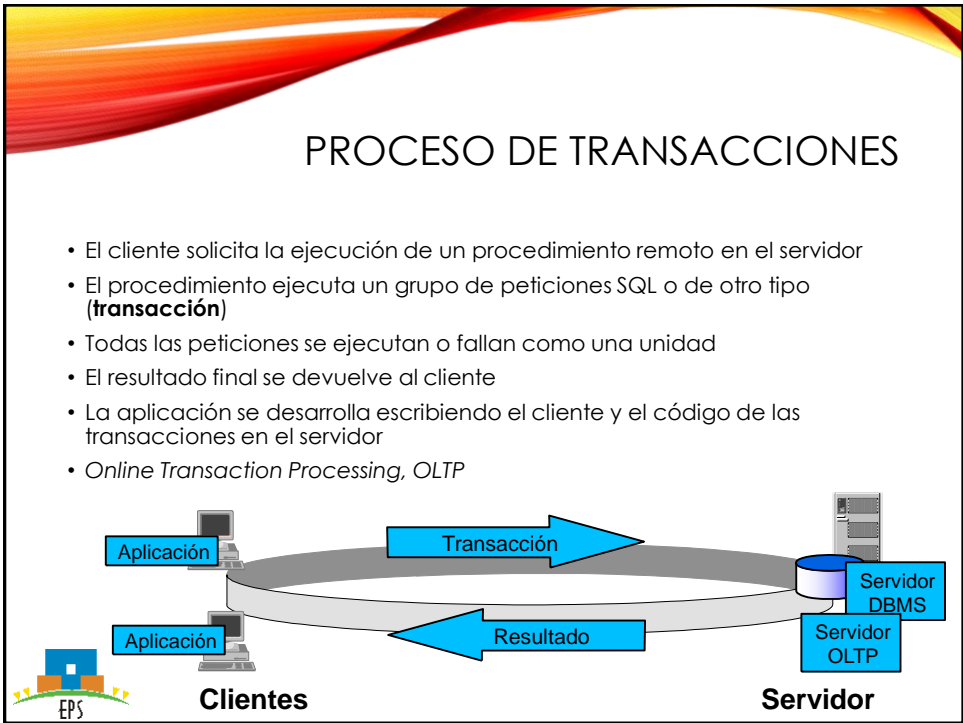
- Red de compartición de recursos
- **Servidores de base de datos**
- **Proceso de transacciones**
- Sistemas de soporte de trabajo en grupo
- Sistemas de objetos distribuidos
- **Servicios Web**
- **Sistemas distribuidos con clientes basados en la WWW**



SERVIDORES DE BASE DE DATOS

- El cliente pasa una petición (query) SQL en un mensajes al servidor (*Data Base Management Server, DBMS*)
- Los resultados de cada consulta SQL se devuelven por la red
- El código que ejecuta la petición SQL se encuentra en el mismo ordenador que los datos





SISTEMAS DISTRIBUIDOS BASADOS EN LA WORLD WIDE WEB

- En origen, sistemas distribuidos basados en la extensión del modelo de cliente ligero bajo protocolo HTTP para el intercambio de información
- Clientes:
 - Universales: navegadores web (Web Browsers)
 - Específicos: programas ejecutados en el cliente
- Servidores Web + Servidor de Aplicaciones:
 - Repositorios de documentos
 - Entorno de ejecución de aplicaciones
- Servidores de *Back-End*:
 - Enlace con programas ya existentes o desarrollos antiguos (*legacy systems*).
 - Programas en el servidor Web les realizan consultas a través de algún tipo de middleware más o menos elaborado
- Los casos más comunes son:
 - Servidores de bases de datos
 - Servidores de proceso de transacciones.



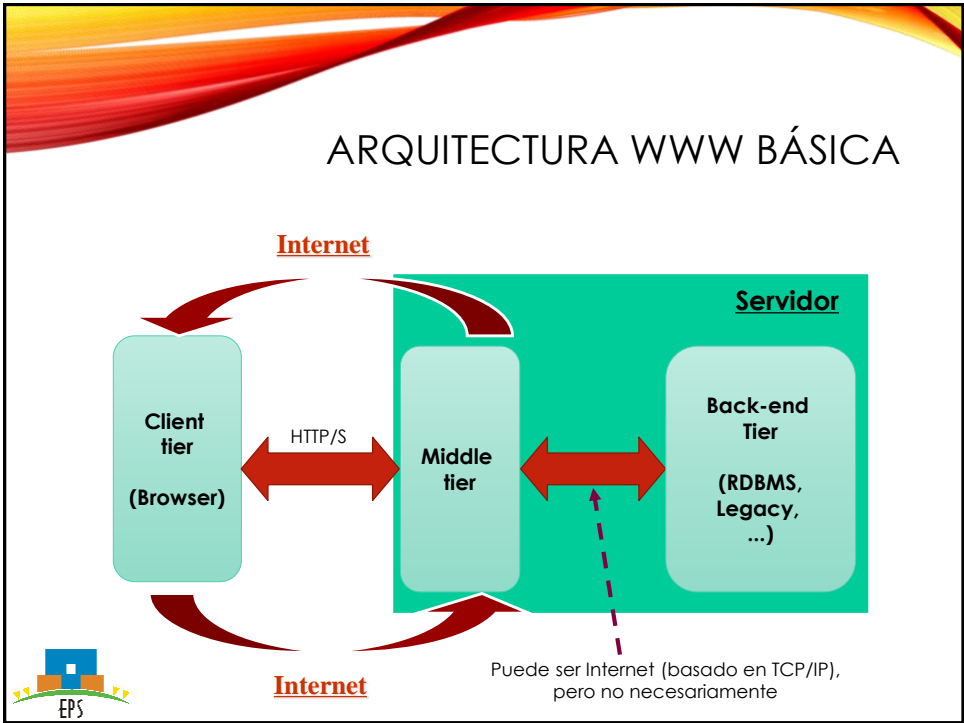
WORLD WIDE WEB VS. INTERNET

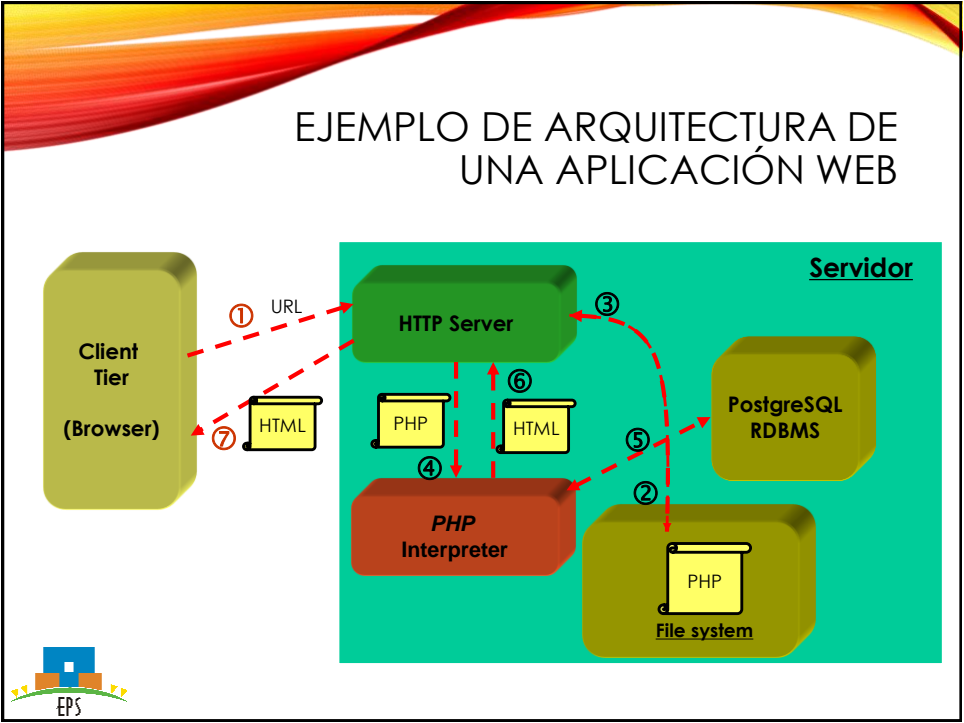
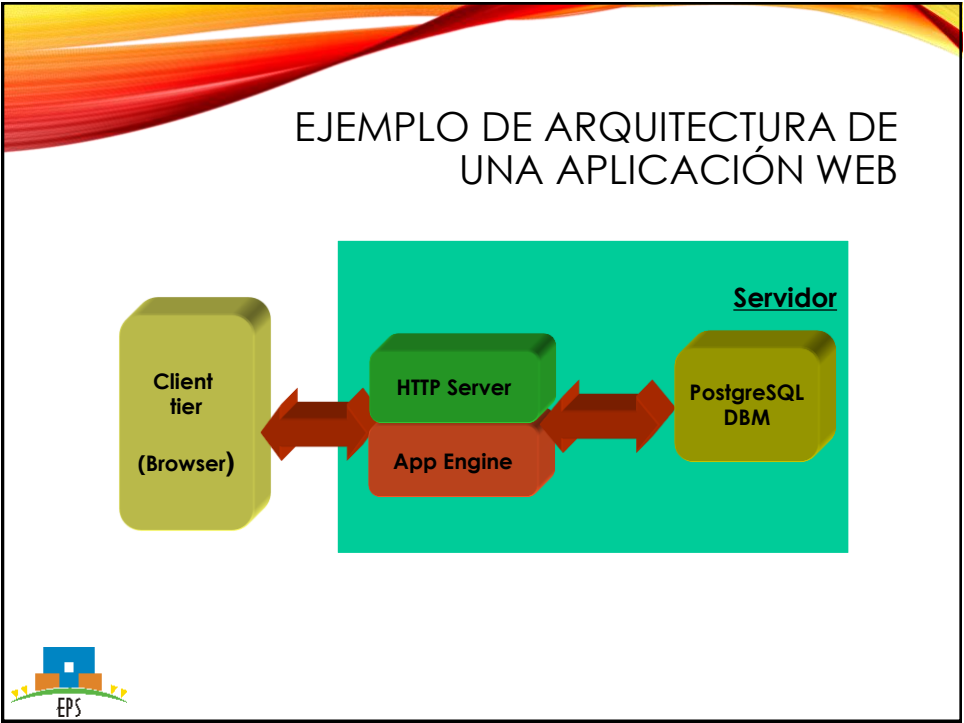


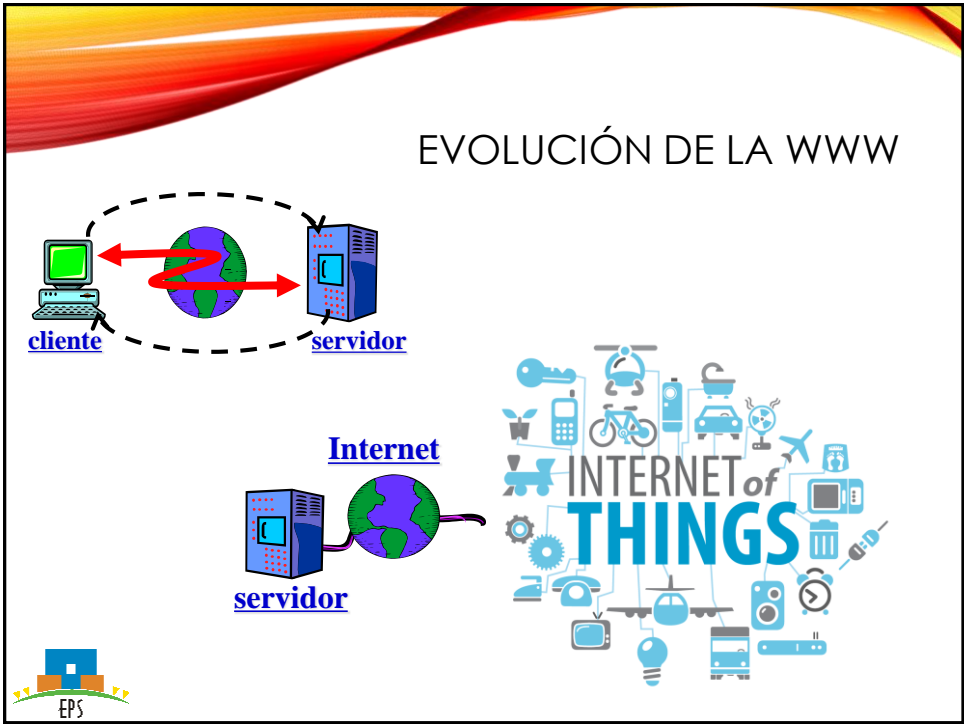
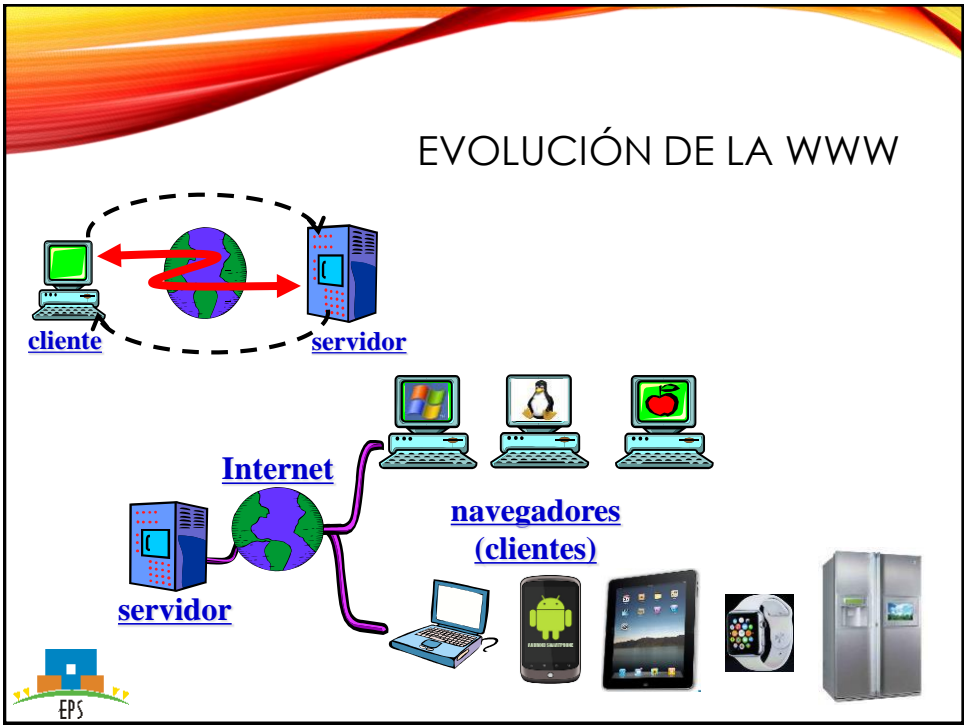
Aunque en contextos no técnicos muchas veces se usan de forma equivalente, hacen referencia a conceptos muy distintos

Interesante historia: <https://www.youtube.com/watch?v=9hIQjrMHTv4>









INTRODUCCIÓN A LA COMPUTACIÓN EN LA NUBE

- La **computación en la nube** (*cloud computing*) es una tecnología que en los últimos años ha ganado en popularidad y se ha convertido en tendencia para el desarrollo y despliegue de sistemas distribuidos
- Pero, ¿qué es la computación en la nube?
 - Almacenamiento y acceso a través de internet a datos y programas en una ubicación remota (≠ máquina remota)
 - Modelo que permite el acceso ubicuo, simple y bajo demanda a un grupo compartido y configurable de recursos computacionales (servidores, almacenamiento, aplicaciones o servicios) que pueden ser aprovisionados y desplegados con un esfuerzo de administración mínimo o con la mínima intervención del proveedor que da dichos servicios [*National Institute of Standards and Technologies*] → 5-4-3 principios de la computación en la nube
 - Definición e implementación de sistemas distribuidos en los que **todos** los recursos HW y SW son ofrecidos como un servicio (normalmente de pago) por un tercero (*cloud service provider*)



EJEMPLOS DE CLOUD SERVICE PROVIDERS



<https://www.heroku.com/>



<http://aws.amazon.com/>



<https://azure.microsoft.com>



VENTAJAS Y DESVENTAJAS DE LA COMPUTACIÓN EN LA NUBE

- Ventajas
 - Disminución de costes
 - Una infraestructura 100% en la nube no requiere instalar ningún tipo de HW/SW
 - *pay-as-you-go, pay-as-per-use*
 - Permite desarrollos complejos sin necesidad de un conocimiento avanzado
 - En muchas ocasiones el uso de un servicio simplemente requiere de su customización
 - Rapidez de desarrollo y con menos riesgo
 - Acceso a nuevas tecnologías
 - Administración
 - Tiempo y recursos/coste
 - Alta disponibilidad
 - Actualización de sistemas
 - Escalabilidad
- Desventajas
 - Privacidad
 - Falta de control y dependencia del proveedor



COMPUTACIÓN EN LA NUBE. 5 CARACTERÍSTICAS ESENCIALES

1. **Servicios bajo demanda.** Las capacidades (procesamiento y recursos físicos o virtuales) se adquieren bajo demanda en función de las necesidades
2. **Amplio acceso de red.** Los servicios se encuentran disponibles en una red que puede ser privada, compartida o pública, siendo accesibles a través de mecanismos estándar que permiten su uso por clientes ligeros heterogéneos
3. **Pooling de recursos.** Las capacidades se asignan y reasignan dinámicamente entre los distintos "clientes" en función de la demanda
4. **Rápida elasticidad.** El escalado horizontal y vertical de capacidades debe producirse rápidamente (en algunos casos incluso de forma automática)
5. **Medición de servicios.** El uso de recursos debe monitorizarse y reportarse de manera automática de cara a su control y optimización



COMPUTACIÓN EN LA NUBE.
4 MODELOS DE DESPLIEGUE

1. **Nube privada.** Sólo una organización tiene acceso a la infraestructura

2. **Nube pública.** El acceso a la infraestructura es abierto

3. **Nube comunitaria.** La infraestructura es compartida por varias organizaciones

4. **Nube híbrida.** La infraestructura general se compone de dos o más infraestructuras con distintos modelos de despliegue



COMPUTACIÓN EN LA NUBE.
3 MODELOS DE SERVICIO

1. **Infraestructura (IaaS):** Se proporciona capacidad de procesamiento, almacenamiento, red y otros recursos computacionales fundamentales (servidores, sistemas operativos, virtualización...). Esto permite desplegar y ejecutar cualquier software arbitrario. El proveedor del servicio es el dueño del equipamiento y el responsable del housing y el mantenimiento. El "cliente" controla y administra el sistema operativo, las aplicaciones, los datos, y ciertos componentes de la red (p.ej., firewalls)

2. **Plataforma (PaaS):** Este tipo de arquitectura está orientada a desarrolladores. Ofrece un entorno preconfigurado de desarrollo usando los lenguajes de programación, librerías, servicios y herramientas soportados por el proveedor. El "cliente" no gestiona, ni controla la infraestructura

3. **Software (SaaS):** Se da acceso a las aplicaciones que el proveedor ejecuta en su infraestructura, sin tener ningún control sobre ésta

