DESARROLLO WEB- SERVIDOR

La unidad cubre los siguientes contenidos principales: Arquitecturas cliente/servidor, Aplicaciones Web, Lenguajes de programación en entorno servidor, Herramientas de programación (IDE), y Tecnologías de Servidor.

Modelos y Arquitectura Cliente/Servidor

Los modelos de programación en entornos cliente/servidor utilizan el **URL (Uniform Resource Locator)**, que sigue el formato `Protocolo://nombrededominio/recurso` o `Protocolo://maquina[:puerto]/recurso`.

La arquitectura Cliente-Servidor se basa en la idea de servicios:

1. El cliente solicita servicios (REQUEST).

2. El servidor es el proceso proveedor de dichos servicios (RESPONSE).

La comunicación se realiza mediante el **intercambio de mensajes**. El cliente, típicamente a través de un navegador, inicia la solicitud de datos, y el servidor responde enviando flujos de datos.

Capas de la Arquitectura Cliente-Servidor

La arquitectura contempla varias capas:

1. **Capa Presentación (o IGU):** Es la capa que interactúa con el usuario, conocida como **interfaz gráfica de usuario (IGU)**. Debe ser "amigable" y se comunica únicamente con la capa de negocio.

2. **Capa Lógica o de Negocio:** Es la capa intermedia. Aquí residen los programas que se ejecutan y se establecen **todas las reglas que deben cumplirse**. Recibe peticiones del usuario y las procesa, comunicándose con la capa de presentación y la capa de persistencia.

3. **Capa Persistencia o Datos:** Es la capa donde residen y se accede a los datos. Está formada por uno o más **Sistemas Gestores de Bases de Datos (SGBD)** que reciben solicitudes de almacenamiento o recuperación desde la capa de negocio.

Tipos de Arquitecturas Cliente-Servidor

Las arquitecturas se pueden clasificar según el número de capas que interactúan:

• **Cliente-Servidor de 2 capas:** El servidor responde utilizando sus propios recursos, sin requerir otra aplicación/servidor adicional. Un ejemplo es la Gestión de Inventario, donde la lógica de validación puede residir en el cliente y la Base de Datos en el servidor.

• **Cliente-Servidor de 3 capas:** El servidor requiere **otra aplicación/servidor** para proporcionar parte del servicio. Un ejemplo es un Sistema de Reservas de Vuelos, que maneja la interfaz, el almacenamiento de datos y el procesamiento de la solicitud (búsqueda y validación de disponibilidad).

• **Cliente-Servidor de N capas:** Los procesos están distribuidos en diversas capas (lógicas y/o físicas), ejecutándose en diferentes equipos. Un ejemplo de esto es una **Tienda online**.

Aplicaciones Web

Las aplicaciones Web son proporcionadas por un servidor Web y utilizadas por clientes (navegadores). Consisten en una colección de páginas, en gran parte **dinámicas** (como ASP, PHP, etc.), agrupadas para dar un servicio.

Las aplicaciones Web se clasifican en:

1. **Aplicaciones Web Estáticas:** El cliente recibe una página que no conlleva ningún tipo de acción ni genera respuesta del servidor. Utilizan HTML para la organización visual.

2. **Aplicaciones Web Dinámicas:** La interacción del cliente con la página web provoca algún cambio en la visualización (ej., cambio de formato, comienzo de animaciones). Incluyen DHTML, Flash, CSS, JavaScript, etc..

3. **Aplicaciones Web Interactivas:** La interacción del cliente genera un **diálogo entre el cliente y el servidor**. Son las más utilizadas en Internet actualmente. Utilizan tecnologías del lado cliente (HTML, AJAX, Flash, applets) y del lado servidor (lenguajes embebidos como PHP, ASP, JSP, servlets).

Para la **ejecución de código en un servidor Web** se requieren:

1. Un **Servidor Web** (recibe peticiones y elige el módulo de ejecución).

2. Un **Módulo encargado de ejecutar el código** (dependiente del lenguaje/tecnología).

3. Una **Aplicación de BD** (normalmente un SGBD).

4. El **Lenguaje de programación**.

Los **Servidores de Aplicaciones** ejecutan aplicaciones web o de escritorio, e incluyen lenguajes de programación, conectores de BD y **middleware** (software de conectividad) para la intercomunicación, confiabilidad y seguridad.

Lenguajes de Programación en Entorno Servidor

Los lenguajes de entorno servidor se pueden clasificar en:

1. **Lenguajes de scripting o interpretados:** Se ejecutan directamente a partir de su código fuente mediante un intérprete que genera el código HTML. **No se compilan** . Ejemplos: Perl, Python, PHP, Ruby y ASP .

2. **Lenguajes compilados a código nativo:** El código fuente se traduce a código binario (dependiente del procesador) antes de la ejecución. Ejemplos: CGI (Common Gateway Interface) programados en lenguajes como C, Perl, C++, Java, etc., que producen un ejecutable.

3. **Lenguajes compilados a código intermedio:** El código fuente se traduce a un **código intermedio (bytecode)**, independiente del procesador, para ejecutarse en distintas plataformas. Un ejemplo es **Java Server Pages (JSP)**, que se ejecuta en servidores que permiten Servlets, los cuales son compilados, almacenados en el contenedor y cargados en memoria tras la primera invocación para agilizar las peticiones posteriores.

Patrones y Herramientas de Desarrollo

Patrón de Diseño MVC

En el desarrollo Web, se utiliza el patrón **MVC (Model, View, Controller)** para separar las capas:

• **Modelo:** Contiene la representación de los datos y sus mecanismos de persistencia.

• **Vista:** Compone la información que se envía al cliente (interfaz de usuario).

• **Controlador:** Actúa como intermediario, gestionando el flujo de información entre el Modelo y la Vista, adaptando los datos a la lógica de negocio.

Herramientas de Programación y Frameworks

Los instrumentos involucrados incluyen navegadores (Google Chrome, Firefox), editores, herramientas de administración de bases de datos, herramientas de tratamiento de imágenes, y **Entornos de Programación (IDE)**. Ejemplos de IDEs son Eclipse, NetBeans, y Visual Studio Code.

Los **IDEs** aportan funcionalidades como:

• **Resaltado de texto** y **indentado automático**.

• Completado automático.

• Navegación en el código. Permite buscar elementos dentros del texto

• **Generación automática de código** crea la estructura básica (esqueleto o *stub*) solo hay que rellenarla.

• Ejecución y depuración.

• **Gestión de versiones** (en combinación con un sistema de control de versiones).

Un **Framework** es una estructura o esqueleto software que facilita la aplicación de patrones de diseño (como MVC). Permite la modularidad y la reusabilidad de código permitiendo aplicar a múltiples interfaces para distintas vistas. Entre sus funcionalidades se encuentran el sistema de plantillas Web, mecanismos de seguridad y autenticación, acceso mapeo y configuración de BD, y **mapeo de "URL amigable"**.

Tecnologías de Servidor

Según el modo en que procesan las peticiones del cliente, los servidores web se clasifican en:

1. **Basados en Procesos:** Un proceso principal escucha peticiones. Cuando llega una, se **duplica creando una copia exacta (*fork*)** para atender la solicitud, mientras el principal sigue escuchando.

2. **Basados en Hilos:** Un proceso principal escucha. Cuando llega una petición, **crea un Hilo de ejecución (*thread*)** que comparte el mismo espacio de memoria (lo que puede generar interbloqueo). El hilo atiende la petición mientras el proceso principal sigue escuchando nuevas peticiones.

3. **Dirigidos por Eventos:** Se basan en **sockets no bloqueantes**. Un *socket* es un espacio de memoria para compartir entre procesos en dos máquinas distintas (cliente/servidor), identificado por IP:puerto. La concurrencia de procesamiento es **simulada**, ya que hay un único proceso y un

solo hilo atendiendo las conexiones. Las lecturas y escrituras entre sockets son asíncronas y bidireccionales

4. **Implementados en el Kernel:** Utilizan un espacio de trabajo del Sistema Operativo (SO) en lugar del área de usuario. Tienen muchos problemas e inconvenientes en la práctica, ya que cualquier fallo a nivel de Kernel puede inutilizar el SO.

Finalmente, la unidad señala que un documento PHP, una vez interpretado en el servidor, produce una **página HTML** que es enviada al cliente.