UT1: IMPLANTACIÓN DE ARQUITECTURAS WEB

1. Introducción.
2. Principales arquitecturas:
    - 2.1. Arquitectura en 2 niveles: Cliente servidor.
    - 2.2. Arquitectura en 3 capas.
3. Protocolos más utilizados.
4. Servidores web. Características.
5. Servidores de aplicaciones. Características.
6. Principales tecnologías para implementar servidores de aplicaciones.
    - 6.1. Tecnología Java J2EE
    - 6.2. Tecnología Microsoft
    - 6.3. Servidores WAMP/LAMP
7. Aplicaciones web. Modelo MVC. Estructura y despliegue de una aplicación web.

1. Introducción.

La **World Wide Web** (*Web*) nació en 1989. La Web trabaja siguiendo un modelo cliente/servidor y utiliza el protocolo **HTTP** (**HyperText Transfer Protocol**) para la comunicación entre los equipos. El lenguaje estándar en el que se codifican los contenidos hipertexto (páginas) es **HTML** (**Hypertext Markup Language**).

Los **servidores Web** son la base en la que se apoya el funcionamiento de la World Wide Web

Un poco de historia: cuando se crearon los primeros servidores de páginas Web (*Web servers*), su única misión era recuperar una página Web estática de su disco duro y enviársela al cliente (navegador). El uso generalizado de servidores Web hizo necesario incrementar los servicios ofrecidos, se buscó desarrollar tecnologías que permitieran dar dinamismo a las páginas. Fueron desarrollados lenguajes que pueden ser incluidos dentro de archivos HTML y que se ejecutan en el lado del servidor, para generar páginas dinámicas con acceso a bases de datos, como por ejemplo las páginas **ASP** , **JSP** o **PHP**.

Finalmente la evolución ha llevado a crear un nuevo término: **servidor de aplicaciones** (**Application server)**. A un nivel básico, casi todos los servidores Web actuales son también servidores de aplicaciones, ya que incluyen alguna tecnología (CGI, PHP, JSP, etc.) que permite crear aplicaciones que generan contenido dinámico.

2. Principales arquitecturas:

2.1. Arquitectura en 2 niveles: Cliente servidor
2.2. Arquitectura en 3 capas o niveles

2.1. Arquitectura en 2 niveles: Cliente servidor:

La arquitectura cliente-servidor es un modelo de aplicación distribuida en la que las tareas se reparten entre los llamados **servidores** (proveedores de recursos o servicios) y los **clientes** (demandan o solicitan los recursos a los servidores).

La separación entre cliente y servidor es de tipo lógico, donde el servidor no es necesariamente una única máquina, ni un solo programa. Aunque la lógica (cliente-servidor) se puede aplicar a programas que se ejecutan en una única máquina (localhost), tiene más sentido y mayores ventajas en sistemas distribuidos en red. La mayor parte de los servicios de Internet utilizan esta arquitectura cliente-servidor, entre otros: servicio web, servicio FTP, servicio de correo, noticias, mensajería y otros servicios de red, como por ejemplo: DHCP y DNS.

Las principales **características del cliente** son:

- Inicia solicitudes o peticiones al servidor.
- Espera y recibe las respuestas del servidor.

- Normalmente, son aplicaciones que interactúa con los usuarios finales, mediante una interfaz gráfica, como por ejemplo, un navegador, un cliente ftp, un programa cliente de correo etc. Los navegadores web quedan relegados a la visualización de las páginas que son servidas por el servidor

Las principales **características del servidor** son:

- Esperan a que les lleguen solicitudes desde los clientes. Son programas (software) que "escuchan" en un determinado **nº de puerto** (es un número que se asocia con el servicio, por ejemplo, el 80 identifica al servicio web, el 21 al servicio FTP, …). En las solicitudes al servidor, se envía el nº de puerto al que va dirigida la petición, el servicio asociado a dicho nº será quien responda. Un servidor puede ofrecer más de un servicio.
- En el servidor reside la lógica de la aplicación., tras la recepción de una solicitud la procesan y envían la respuesta al cliente.
- Los servidores aceptan conexiones desde múltiples clientes (el nº máximo puede estar limitado)

2.2. Arquitectura en 3 niveles:

En la arquitectura a 3 niveles, se añade a la arquitectura **cliente-servidor** l una nueva capa: la **capa de datos**:

- ✓ **Capa cliente**, el interface de usuario, representada por navegadores Web o clientes ricos (como aplicaciones Java).
- ✓ **Capa servidor** está representada por un **servidor web** o un **servidor de aplicaciones** que ejecuta el código de negocio.
- ✓ **Capa de datos**, representada por bases de datos relacionales u otras fuentes de datos finales.

En la arquitectura de tres-capas , hay una separación física entre el cliente que solicita la información, los programas que la procesan y los datos sobre los que operan. El servidor web gestiona la solicitud y la respuesta al cliente, el servidor de aplicaciones ejecuta parte de la lógica del programa y accede a los datos de una base de datos, (puede haber un servidor de datos independiente a través del cual se acceda a los datos).

El proceso sería el siguiente:

- Los usuarios finales interactúan con el software cliente, normalmente un navegador Web, para usar la aplicación.
- Cuando un navegador Web origina una solicitud, se envía al servidor Web. Asumiendo que la solicitud requiere una aplicación para procesar o acceder a los datos, el servidor web reenvía la petición al servidor de aplicaciones.
- El servidor de aplicaciones ejecuta el código de la aplicación apropiada, accede a las fuentes de datos si lo necesita y devuelve el resultado al servidor web.
- El servidor web, a su vez, reenvía la respuesta de vuelta al cliente.

Algunas ventajas de la arquitectura a 3 niveles:

- Permite una mayor flexibilidad
- Mayor seguridad, ya que la seguridad se puede definir de forma independiente para cada servicio y en cada nivel.
- Mejor rendimiento, ya que las tareas se comparten entre servidores.

3. Protocolos de aplicación más utilizados.

Un **protocolo** es un conjunto de reglas y procedimientos estándar que permite la comunicación entre procesos. Existen diversos protocolos, en Internet los protocolos utilizados pertenecen al conjunto de protocolos **TCP/IP**, y siguen el modelo cliente-servidor, entre los más utilizados se encuentran:

- Protocolo **HTTP**

Desde 1990, el protocolo **HTTP** (Hiper Text Transfer Protocol o Protocolo de transferencia de hipertexto) es el protocolo más utilizado en Internet. El propósito del protocolo HTTP es permitir la transferencia (principalmente en formato HTML), entre un navegador (cliente) y un servidor web localizado mediante una dirección llamada **URL** (Uniform Resource Locator o Localizador uniforme de recursos).

La comunicación entre el navegador y el servidor se realiza en 2 etapas:

- El navegador realiza una solicitud HTTP.
- El servidor procesa la solicitud y después envía una respuesta HTTP.

- Protocolo **HTTPS**

El protocolo **HTTPS** o Protocolo seguro de transferencia de hipertexto (Hiper Text Transfer Protocol Secure) es la versión segura del protocolo HTTP. Permite realizar transacciones de forma segura , encriptada, mediante la utilización del protocolo **SSL** (Secure Sockets Layer) y la utilización de certificados de servidor que certifiquen la identidad del servidor. La URL de acceso comienza con **https://** y en los navegadores comunes cuando se utiliza el protocolo HTTPS para acceder a una página, se visualiza el icono del candado en la barra del navegador.

- Protocolo **FTP**

El protocolo **FTP** (File Transfer Protocol o Protocolo de transferencia de archivos), es un protocolo que permite la transferencia de archivos entre un cliente y un servidor, este protocolo es más antiguo (1971) y define la manera en que los datos deben ser transferidos a través de la red TCP/IP.

- Protocolo **SMTP/POP3/IMAP4**

El protocolo **SMTP** (Simple Mail Transfer Protocol o Protocolo simple de transferencia de correo) es el protocolo estándar que permite el envío y recepción de correo entre servidores. Los protocolos **POP3** (Protocol Office Post v.3) e **IMAP4** (Internet Messages Access Protoco,l versión 4) permiten el acceso o lectura de los correos de cada usuario, que están almacenados en un servidor;POP3 descarga los correos desde el servidor al equipo del usuario, IMAP permite la lectura de los correos sin descargarlos al equipo local..

4. Servidores web. Características.

Un servidor Web es un programa que escucha en un puerto (normalmente el **80**), a la espera de peticiones desde la aplicación cliente (navegador). Desde el navegador se solicita un recurso que hace referencia a un archivo que está en el disco duro del servidor (podría estar ubicado en otro servidor), es capaz de recuperarlo y enviarlo al cliente. Tanto la petición como la respuesta se encapsulan siguiendo el protocolo HTTP. La arquitectura (figura) se basa en el modelo cliente-servidor y cada parte está típicamente en una máquina distinta de la red, aunque no hay ningún problema si residen en la misma máquina, (localhost).

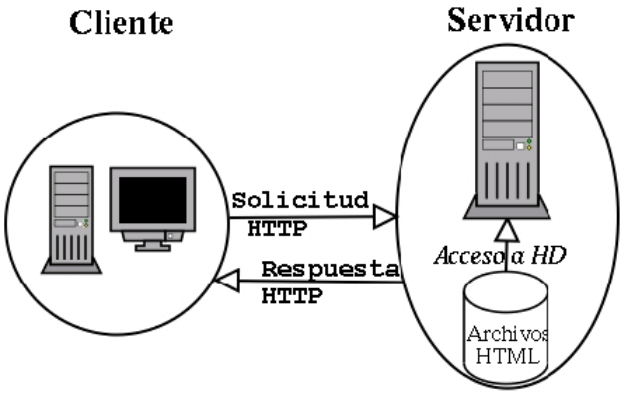

Figura: Arquitectura de funcionamiento de un servidor Web.

Hoy en día existen multitud de productos que funcionan como servidores Web, entre los más utilizados destacan, entre otros, como software propietario: el servicio web de **Internet Information Services (IIS),** de Microsoft, como software libre , el servidor web **Apache**, (Apache Software Foundation).

Internet Information Services (IIS)

Internet Information Services, es una serie de servicios para los ordenadores que funcionan con sistemas operativos de Microsoft Windows. Los servicios que ofrece son: **FTP** (File Transfer Protocol, Protocolo de transferencia de archivos),**SMTP** (correo saliente),NNTP (noticias) y **HTTP/HTTPS** (servidor web/servidor web seguro). Es considerado por Microsoft, como su servidor de aplicaciones.

Los servicios de IIS permiten configurar y administrar de forma sencilla un sitio Web y FTPy publicar páginas web tanto local como remotamente, configurando el servidor como un **servidor de aplicaciones** y convirtiendo a un ordenador en un servidor de Internet o Intranets.

El servidor web tiene varios módulos que le dan capacidad para procesar distintos tipos de páginas, por ejemplo **Active Server Pages** (**ASP**) y **ASP.NET**. También pueden ser incluidos los de otros fabricantes, como **PHP** o **Perl**.

El servidor HTTP Apache

Es un servidor web HTTP de código abierto, de **Apache Software Foundation**., para plataformas **Unix** (BSD,GNU/Linux, etc.), **Microsoft** Windows**, Macintosh** y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual.

. Apache es el componente de servidor web en la popular plataforma de aplicaciones **XAMP**P (Linux,Windows, Mac/Apache/MySQL/PHP. Algunos de los más grandes sitios web del mundo están ejecutándose sobre Apache. La capa frontal (front end) del motor de búsqueda Google está basada en una versión modificada de Apache, denominada Google Web Server (GWS).

Microsoft Internet Information Services (**IIS**) es el principal competidor de Apache.

5. Servidores de aplicaciones. Características.

Un **servidor de aplicaciones** no es más que una extensión software que gestiona la mayor parte o la totalidad de las funciones de lógica de negocio y de acceso a los datos de una aplicación.

Un servidor de aplicaciones clásico (figura) se apoya en un modelo cliente-servidor de tres capas:

1. **Presentación**: una interfaz generalmente gráfica que reside en los clientes. El ejemplo típico es un navegador.
2. **Lógica de negocio**: el conjunto de programas que dan soporte a la aplicación.
3. **Almacenamiento**: generalmente una base de datos.

A modo de ejemplo básico, cuando un cliente introduce un pedido desde un navegador, llega a un servidor Web que envía la solicitud al servidor de aplicaciones, éste ejecuta la lógica y también recupera y actualiza los datos del cliente, desde las fuentes finales.

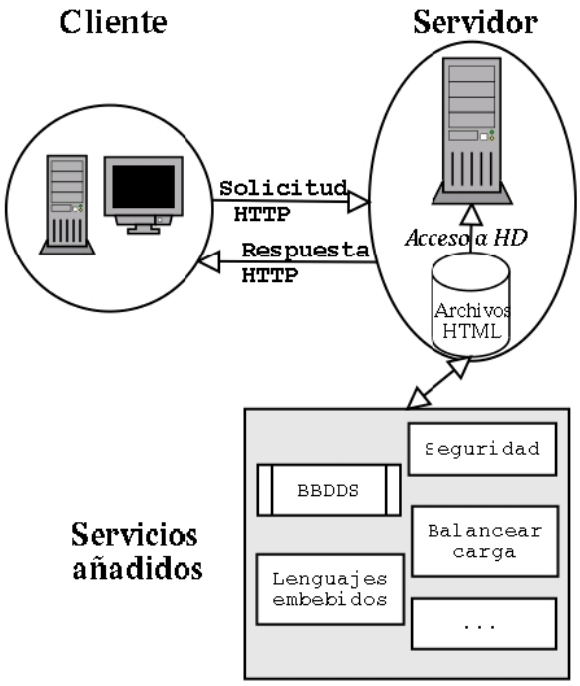

Figura: Arquitectura de funcionamiento de un servidor de aplicaciones.

Principales servicios que añadidos a un servidor web lo convierten en servidor de aplicaciones:

- Generación de HTML, XHTML, XML, … para enviar al cliente
- Acceso a bases de datos
- Funcionamiento multiproceso o multhilo para atender múltiples peticiones
- Sesiones de usuario, provee, mediante objetos de sesión, de persistencia de datos de los diferentes usuarios
- Encapsula la lógica de negocio en componentes, cada uno con sus propios mecanismos de seguridad
- Soporte para aplicaciones seguras, certificados de clientes contra el servidor, control de accesos de usuarios y grupos a recursos o componentes como bases de datos.
- Balanceo de carga, sobre clusters de servidores de forma que pueden enviar peticiones a distintos servidores, dependiendo de la carga y disponibilidad (sistemas tolerantes a fallos).

Los servidores de aplicaciones suelen asociarse con servidores de alto rendimiento pensados para dar servicio a sitios Web con grandes necesidades: afluencia de visitas, transacciones movimiento de datos, etc. Un sistema empresarial podría consistir en varios servidores de aplicaciones, junto a varios servidores de bases de datos y servidores Web. El código de la aplicación puede distribuirse entre los servidores de aplicaciones.

6. Principales tecnologías para implementar servidores de aplicaciones.

6.1. Tecnología Java J2EE

6.2. Tecnología Microsoft

6.3. Servidores WAMP/LAMP

6.1. Tecnología Java J2EE

J2EE (Java 2 Enterprise Edition) es una especificación que propone un estándar para servidores de aplicaciones, define diferentes tecnologías: el lenguaje Java, la máquina virtual Java, los componentes JavaBaens y el conjunto de estándares: **EJB, JSP, Servlets, JDBC, JNDI, RMI** componen la plataforma J2EE.

La plataforma J2EE es un entorno abierto y multiplataforma permite desarrollar y desplegar aplicaciones que se pueden ejecutar en un cualquier servidor de aplicaciones compatible con las especificaciones J2EE, independientemente del sistema o la plataforma en la que corra dicho servidor.

Servidores de aplicaciones J2EE compatibles. Todos los servidores de aplicaciones que quieran ser etiquetados como servidores de aplicaciones J2EE deben pasar un test de compatibilidad, que garantiza la correcta implementación de las tecnologías Java.

- **Un servidor web compatible con las especificaciones J2EE**, debe e incluir un **contenedor web** (**contenedor servlet + contenedor jsp**), ejemplo: **apache tomcat**.
- **Un servidor de aplicaciones compatible con las especificaciones J2EE** debe incluir: un **contenedor web + contenedor EJB (Enterprise Java Beans)** que permite la ejecución de todo tipo de aplicaciones J2EE, en particular de aplicaciones web. Ejemplos: **JBoss** de código abierto y gratuito y **BEA Weblogic Server**, de pago.

A modo de ejemplo se muestra un listado con los productos compatibles J2EE, de algunas empresas conocidas:

- BEA Weblogic Server
- IBM WebSphere
- Sun One application server
- Oracle application server
- ColdFusion de Adobe

Ejemplos open source de otros servidores de aplicaciones compatibles J2EE, son:

- Glassfish
- Tomcat
- JBoss
- Jonas

Las **aplicaciones web** java que proporcionan una forma de generar documentos dinámicos son típicamente: los componentes llamados **servlets** y las páginas **JSP** (**Java Server Pages***)*:

- **Java Server Pages (JSP)** - es una especificación abierta (y gratuita) desarrollada por **Sun Microsystems** como un alternativa a la páginas Active Server Pages (**ASP**) de Microsoft. La tecnología JSP utiliza **código Java** como lenguaje de script incrustado en documentos HTML, XML,…se integran con bases de datos que soporten **JDBC** y **ODBC**.

- **Servlets** - A diferencia de una página JSP, un servlet es un componente escrito *puramente en Java*, equivale a una *Clase Java* compuesta por atributos y métodos. Los servlets no tienen interface gráfico de usuario, se utilizan ampliamente dentro de servidores HTTP que deben integrar el **API Java servlet** (una extensión estándar de Java que se utiliza para escribirlos).

6.2. Tecnología Microsoft.

Microsoft utiliza **Internet Information Services** (**IIS**) como servidor de aplicaciones para sus diferentes tecnologías web. Las primeras soluciones que ofreció esta empresa se basaban en el **lenguaje de script ASP** (Active Server Pages) y la **tecnología** de objetos distribuidos **COM** (Component Object Model). Su nueva apuesta fue **.NET** que incluye entre otros: **ASP.NET** .

.NET es un framework de Microsoft para los sistemas Windows, competencia a la tecnología Java de Sun, y a los diversos desarrollos web basados en PHP. El entorno de desarrollo (IDE) más utilizado para desarrollar aplicaciones con ASP.NET, es **Visual Studio .NET** que incluye todos los lenguajes de desarrollo de Microsoft.

Las **aplicaciones web** típicas de la tecnología Microsoft son**:**

- Tecnología **Active Sever Pages** (**ASP**) - Microsoft introdujo esta tecnología en diciembre de1996. Es parte de **Intenet Information Services (IIS)** , son páginas generadas dinámicamente que incluyen dentro de HTML el uso de diferentes scripts (normalmente VBasicScripts y Java Scripts Se integran con bases de datos que soportan ODBC.
- **Tecnología ASP.NET** - Es la parte de .NET encargada del desarrollo en web, conocidas oficialmente como "***web forms***" (formularios web), Apareció en enero de 2002 y es la tecnología sucesora de la tecnología **ASP**. Los formularios web están contenidos en archivos con una extensión **aspx.** ASP.NET sólo funciona sobre el servidor de Microsoft **IIS**, lo que supone una desventaja respecto a otros lenguajes del lado de servidor, ejecutables sobre otros servidores más populares como Apache.
- **ASP.NET MVC**- Es un framework o marco de trabajo que ha sido creado para desarrollar aplicaciones que sigan la filosofía **MVC (Modelo Vista Controlador),** sobre ASP.NET.
- **Web services.** Los servicios web permiten que las aplicaciones trabajen colaborativamente, el software está distribuido en diferentes servidores que actúan como un repositorio de servicios de n aplicaciones distribuidas por internet. Microsoft para conseguir este propósito con su tecnología .NET emplea como protocolo de comunicación una aplicación XML, llamada **SOAP (Simple Object Access Protocol)**.

6.3. Servidores WAMP/XAMP

Estas plataformas son una alternativa a las tecnologías anteriores y son utilizadas en millones de sitios web ya que, al ser software libre, se presenta como una alternativa de fácil acceso para todos.

Estas soluciones se basan en el servidor web **Apache**, el lenguaje **PHP** como lenguaje de script de servidor y la base de datos **MySQL** conocidas popularmente como plataforma de aplicaciones **LAMP** si se implementa sobre **Linux:** (Linux/Apache,MySQLy PHP), y **WAMPP** sobre Windows: (Windows/Apache/MySQL y PHP).

Para el diseño de páginas web dinámicas se utiliza el lenguaje **PHP** (PHP Hipertext Pre-processor), lenguaje interpretado y de código abierto que se incrusta dentro de código HTML, es ejecutado generalmente en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas. El servidor es el que se encarga de ejecutar el código y enviar el resultado HTML al navegador. Tiene capacidad para conectarse con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con **MySQL** y **PostgresSQL**, también permite aplicar técnicas de programación orientada a objetos

7. Aplicaciones web. Modelo MVC. Estructura y despliegue de aplicaciones web.

El modelo de programación MVC (Modelo-Vista-Controlador).

La estructura **MVC** ("**Model-View-Controller**") es un paradigma de programación utilizado en diversos desarrollos de software que pretende separar la capa visual gráfica de su correspondiente programación y acceso a datos.

MVC logra una división en 3, de las diferentes partes que conforman una aplicación: **la interfaz de usuario, la lógica de negocio y los datos.** Su principal razón de ser es facilitar el mantenimiento del código fuente.

Estructura de una aplicación web.

Una aplicación web consiste en un conjunto de páginas, ficheros html, xml, clases y otro tipo de recursos tales como ficheros de imágenes, de sonidos, de texto que se organizan en una estructura de directorios. Al directorio propio que contiene toda la aplicación web se le denomina **contexto de la aplicación.**

Empaquetado de la aplicación web.

La estructura de directorios y archivos que componen una aplicación web se puede empaquetar en archivos comprimidos (tipo .zip o .jar) para facilitar su despliegue en cualquier servidor compatible; en el caso de las aplicaciones J2EE que se suelen empaquetar en un archivo de extensión **.war.** Un archivo **XML** llamado **descriptor de despliegue** describirá cómo se debe desplegar la aplicación, esto hace fácil desplegar la misma aplicación en varios servidores Web. Para aplicaciones web desplegadas en **IIS**, Visual Studio 2010 incorpora una herramienta que facilita el proceso de despliegue: *IIS Web Deployment Tool* con la que se pueden crear web packages que luego se pueden importar, desde la consola del IIS.

Despliegue de la aplicación web.

El proceso de **despliegue** (deployment) de la aplicación consiste en publicar o implantar toda la estructura que integra la aplicación, en un servidor web o de aplicaciones, por tanto, desplegar una aplicación es hacerla accesible para que pueda ser visualizada desde cualquier navegador. En la fase de desarrollo de la aplicación, mientras se está probando se desplegará en un **servidor de desarrollo** y cuando ya está finalizada, sobre un **servidor de producción**. Para que los usuarios puedan realizar despliegues sobre el servidor, habrá que concederle los permisos adecuados sobre el sitio web, en el que se va a realizar el despliegue.

El proceso inverso (undeploy) desacoplaría la aplicación del servidor, por ejemplo, antes de volver a realizar un despliegue de la misma o cuando queremos eliminarla del servidor.

Contexto del servidor.

La estructura de directorios de la aplicación debe alojarse en un directorio determinado del servidor (**Directorio raíz del servidor**, "/", o **contexto del servidor**), la ruta a este directorio cambia dependiendo del servidor y la tecnología utilizada. Esta ruta le indica al servidor dónde debe localizar todas las páginas que se le soliciten. Para una solicitud vacía desde un navegador: http://servidor/ el servidor intentaría abrir una página inicial del tipo: index.html, index.php,…. que estará localizada en el directorio raíz del servidor.

Contexto de la aplicación web.

El directorio que contienen a la aplicación llamado **contexto de la aplicación** se localizará dentro del directorio raíz del servidor. Por ejemplo, si nuestra aplicación web es desplegada en el directorio ***/miaplicacion*** del servidor, a solicitudes del tipo: ***http://servidor/miaplicacion***, el servidor localizará la página inicial de la aplicación dentro del directorio: */miaplicacion* donde ***"/"*** representa el directorio raíz o contexto del servidor y el directorio *"miaplicacion*" representa el contexto o directorio de la aplicación***.**