DESPLIEGUE DE APLICACIONES WEB

TEMA 1. Implantación de arquitecturas web.

1.- Aspectos generales de arquitecturas web.
- 1.1.- Evolución de los servicios web.
- 1.2.- Tecnologías asociadas a las aplicaciones web.
- 1.3.- Tipos de aplicaciones web.
- 1.4.- Arquitecturas web. Modelos.
- 1.5.- Plataformas web libres y propietarias.
- 1.6.- Escalabilidad.
  - 1.6.1.- Escalabilidad vertical.
  - 1.6.2.- Escalabilidad horizontal.
  - 1.6.3.- Cluster.

2.- Servidor web Apache.
- 2.1.- Instalación y configuración.
- 2.2.- Iniciar Apache.

3.- Aplicaciones web y servidores de aplicaciones.
- 3.1.- El servidor de aplicaciones Tomcat.
- 3.1.1.- Instalación y configuración básica.
- 3.1.2.- Iniciar Tomcat.

4.- Estructura y despliegue de una aplicación web.
- 4.1.- Archivos WAR.
- 4.2.- Despliegue de aplicaciones con Tomcat.
- 4.3.- Descriptor de despliegue.

DESPLIEGUE DE APLICACIONES WEB

TEMA 1. Implantación de arquitecturas web.

1.- Aspectos generales de arquitecturas web.

1.- Aspectos generales de arquitecturas web.

La arquitectura World Wide Web (WWW) de Internet provee un modelo de programación poderoso y flexible. Las aplicaciones y los contenidos son presentados en formatos de datos estándar y son localizados por aplicaciones conocidas como "web browsers", que envían requerimientos de objetos a un servidor y éste responde con el dato codificado según un formato estándar.

Los estándares WWW especifican muchos de los mecanismos necesarios para construir un ambiente de aplicación de propósito general, por ejemplo:

- ✓ **Modelo estándar de nombres:** todos los servidores, así como el contenido de la WWW se denominan según un Localizador Uniforme de Recursos (Uniform Resource Locator: URL).

- ✓ **Formatos de contenidos estándar:** todos los navegadores soportan un conjunto de formatos estándar, por ejemplo HTML, ECMA, JavaScript, etc.

- ✓ **Contenido:** a todos los contenidos en la WWW se les especifica un determinado tipo permitiendo de esta forma que los browsers (navegadores) los interpreten correctamente.

- ✓ **Protocolos estándar:** éstos permiten que cualquier navegador pueda comunicarse con cualquier servidor web. El más comúnmente usado en WWW es HTML (Protocolo de Transporte de Hiper-Texto), que opera sobre el conjunto de protocolos TCP/IP.

Los aspectos generales a destacar en una arquitectura web son los siguientes:

- ✓ Escalabilidad.
- ✓ Separación de responsabilidades.
- ✓ Portabilidad.
- ✓ Utilización de componentes en los servicios de infraestructura.
- ✓ Gestión de las sesiones del usuario.

1.- Aspectos generales de arquitecturas web.

El esquema de funcionamiento de los servicios web requiere de tres elementos fundamentales:

1. **Proveedor del servicio web,** que es quien lo diseña, desarrolla e implementa y lo pone disponible para su uso, ya sea dentro de la misma organización o en público.

2. **Consumidor del servicio,** que es quien accede al componente para utilizar los servicios que éste presta.

3. **Agente del servicio,** que sirve como enlace entre proveedor y consumidor para efectos de publicación, búsqueda y localización del servicio.

De forma genérica podríamos decir que la arquitectura web es un modelo compuesto de tres capas:

1. **Capa de Base de Datos,** donde estaría toda la documentación de la información que se pretende administrar mediante el servicio web y emplearía una plataforma del tipo MySQL, PostgreSQL, etc.

2. En una segunda capa estarían los **servidores de aplicaciones web,** ejecutando aplicaciones de tipo Apache, Tomcat, Resin, etc.

3. En una tercera capa estarían los **clientes del servicio web** al que accederían mediante un navegador web como Firefox, Internet Explorer, Opera, etc.

1.1.- Evolución de los servicios web.

La evolución del uso de Servicios web en las organizaciones está fuertemente ligada al desarrollo de Internet como red prestadora de servicios. Entre los factores que han impulsado el uso de los servicios web se encuentran:

- El **contenido se está volviendo más dinámico:** Los sitios web actuales proporcionan contenidos "instantáneos". Un Servicio web debe ser capaz de combinar contenido proveniente de fuentes muy diferentes.

- El **ancho de banda es menos costoso:** Actualmente un Servicio web puede entregar tipos variables de contenidos como vídeo o audio. A medida que crezca el ancho de banda, los servicios web deben adaptarse a nuevos tipos de contenidos.

- El **almacenamiento es más barato:** Un Servicio web debe ser capaz de manejar cantidades masivas de datos, y debe poder hacerlo de forma inteligente.

- El **éxito de la computación extendida** se está volviendo más importante: Con cientos de millones de dispositivos como teléfonos móviles, agendas electrónicas, etc... existentes actualmente, estamos llegando a un momento en el cual las computadoras están dejando de ser el dispositivo más común en Internet.

En los orígenes del mundo web nos situábamos ante un entorno estático, con páginas en formato HTML que raramente sufrían modificaciones o actualizaciones y en las que apenas había interacción con el usuario.

La **Web 2.0 es la transición que se ha dado desde las aplicaciones tradicionales hacia aplicaciones que** funcionan a través de la web y que están fuertemente enfocadas al usuario final. En este nuevo entorno existen una serie de nuevas tecnologías que, en general, tienen como objetivo:

- ✓ Transformar software de escritorio hacia la web.
- ✓ Separar hojas de estilo.
- ✓ Potenciar el trabajo colaborativo y la utilización de redes sociales.
- ✓ Dar control total a los usuarios en el manejo de su información.

Ejercicio: Tarea 1.Explicar la evolución de la web, desde sus comienzos hasta la actualidad. (Web 1.0, Web 2.0, Web 3.0, etc.)

1.2.- Tecnologías asociadas a las aplicaciones web.

Las aplicaciones web emplean páginas dinámicas, éstas se ejecutan en un servidor web y se muestran en el navegador de un equipo cliente que es el que ha realizado previamente la solicitud. Cuando una página web llega al navegador, es posible que también incluya algún programa o fragmento de código que se deba ejecutar. Ese código, normalmente en lenguaje JavaScript, lo ejecutará el propio navegador.

Es por ello que en este apartado nos centraremos en las tecnologías asociadas a las aplicaciones web que se ejecutarán tanto del lado del servidor como del cliente:

- **ASP (Active Server Pages):** Las "Páginas Activas" se ejecutan del lado del servidor, de este modo se forman los resultados que luego se mostrarán en el navegador de cada equipo cliente que ha realizado la solicitud. Un buen ejemplo de ello son los buscadores, donde un usuario realiza una petición de información y el servidor nos entrega un resultado a medida de nuestra petición.

- **CGI (Common Gateway Interface):** La "Interface Común de Entrada" es uno de los estándares más antiguos en Internet para trasladar información desde una página a un servidor web**.** Este estándar es utilizado para bases de datos, motores de búsqueda, formularios, generadores de email automático. Las rutinas de CGI son habitualmente escritas en lenguajes interpretados como Perl o por lenguajes compilados como C.

- **Java:** Este es un lenguaje que trabaja en el cliente, es decir: se ejecuta en el navegador del equipo cliente y no en el servidor.

- **JavaScript:** Lenguaje que se interpreta y se ejecuta en el cliente. Útil para realizar tareas como mover imágenes por la pantalla, crear menús de navegación interactivos, etc.

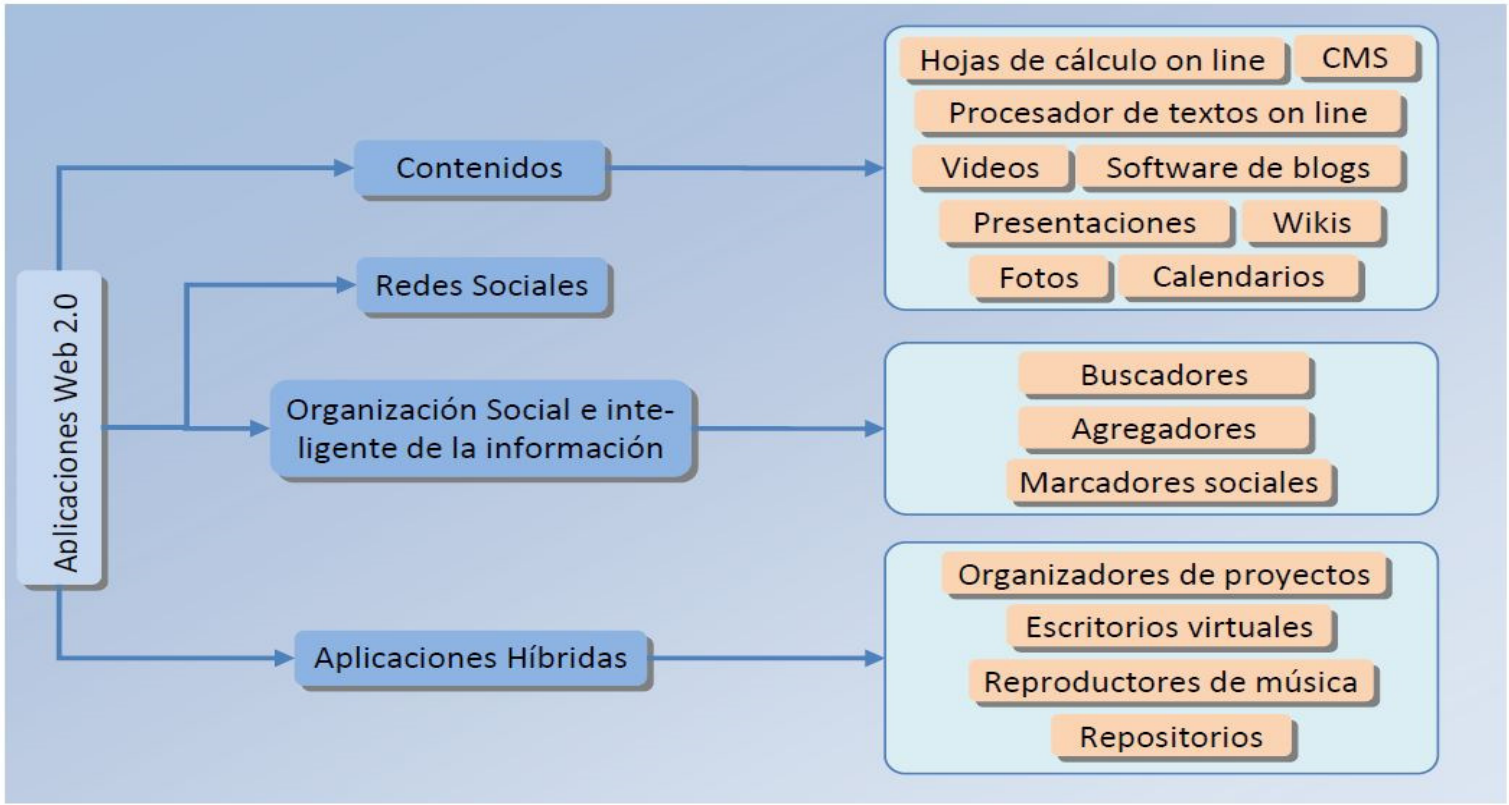
- **PHP (Hypertext Preprocessor):** Este lenguaje es, como ASP, ejecutado en el lado del servidor. PHP es similar a ASP y puede ser usado en circunstancias similares. Es muy eficiente, permitiendo el acceso a bases de datos empleando servidores como MySQL y, por lo tanto, suele utilizarse para crear páginas dinámicas complejas.

- **VBScript (Visual Basic Scripting):** La respuesta de Microsoft a JavaScript. Es una buena herramienta para cualquier sitio destinado a ser mostrado exclusivamente en el navegador Microsoft Internet Explorer. El código en VBScript puede, además, estar diseñado para su ejecución en el lado del cliente o en el del servidor, la diferencia es que un código que se ejecuta en el lado del servidor no es visible en el lado del cliente. Éste recibe los resultados, pero no el código.

1.3.- Tipos de aplicaciones web.

Cualquier proyecto que se quiera desarrollar en Internet, bien sea comercio electrónico, reservas de billetes de vuelo on-line, información meteorológica, registro de usuarios, simuladores de hipotecas, etc, conlleva el desarrollo de una aplicación web.

En definitiva, una aplicación web es una plataforma orientada a automatizar los procesos de servicios que se quieran ofrecer a usuarios.



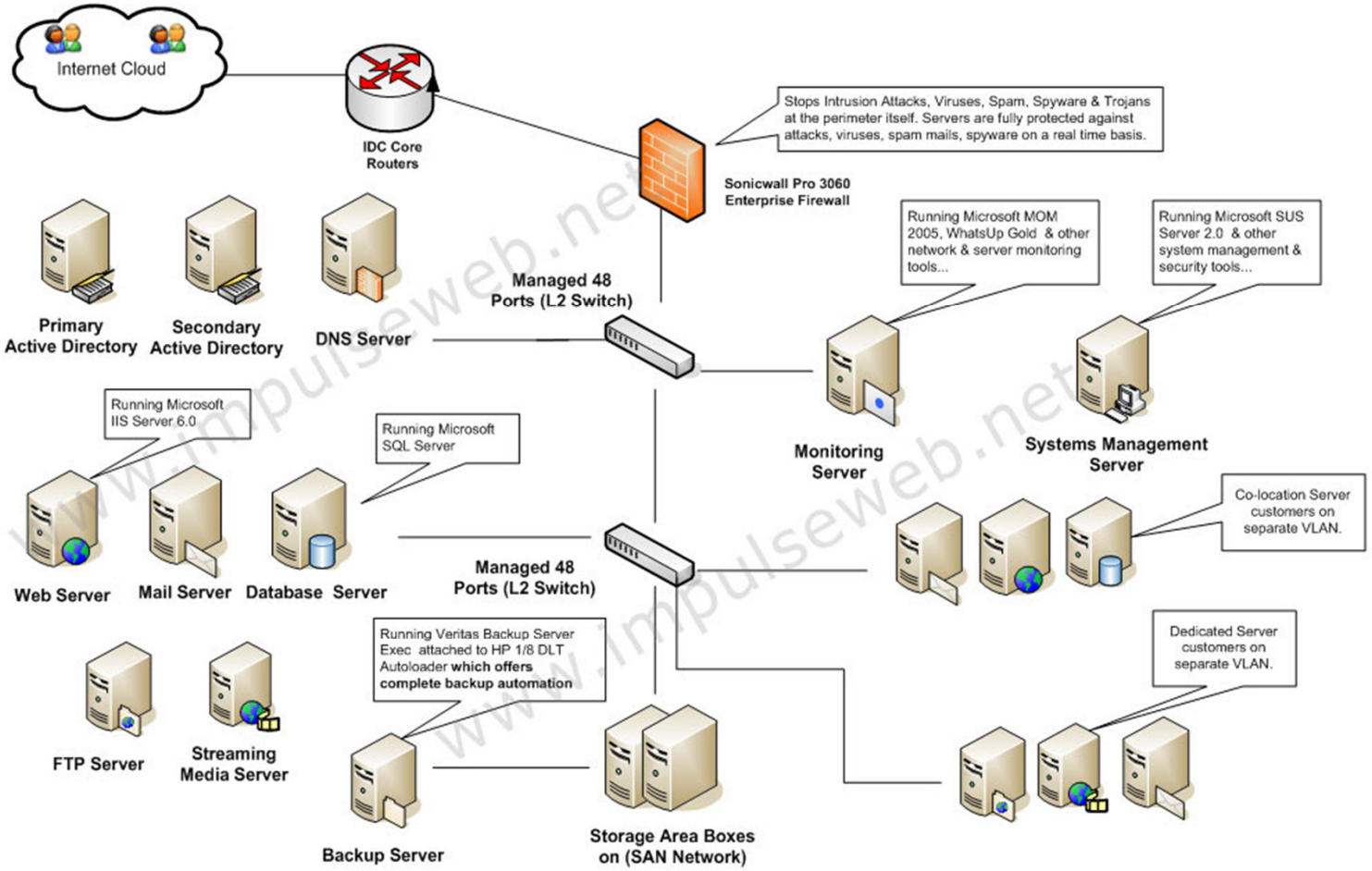
1.4.- Arquitecturas web.

¿Qué es la arquitectura web?

Antes de poder entrar a definir lo que es la arquitectura web primero es necesario enmarcarla correctamente. Dentro del sector de las tecnologías de la información hay diversos roles relacionados con la arquitectura, pero básicamente podríamos hacer la siguiente división:

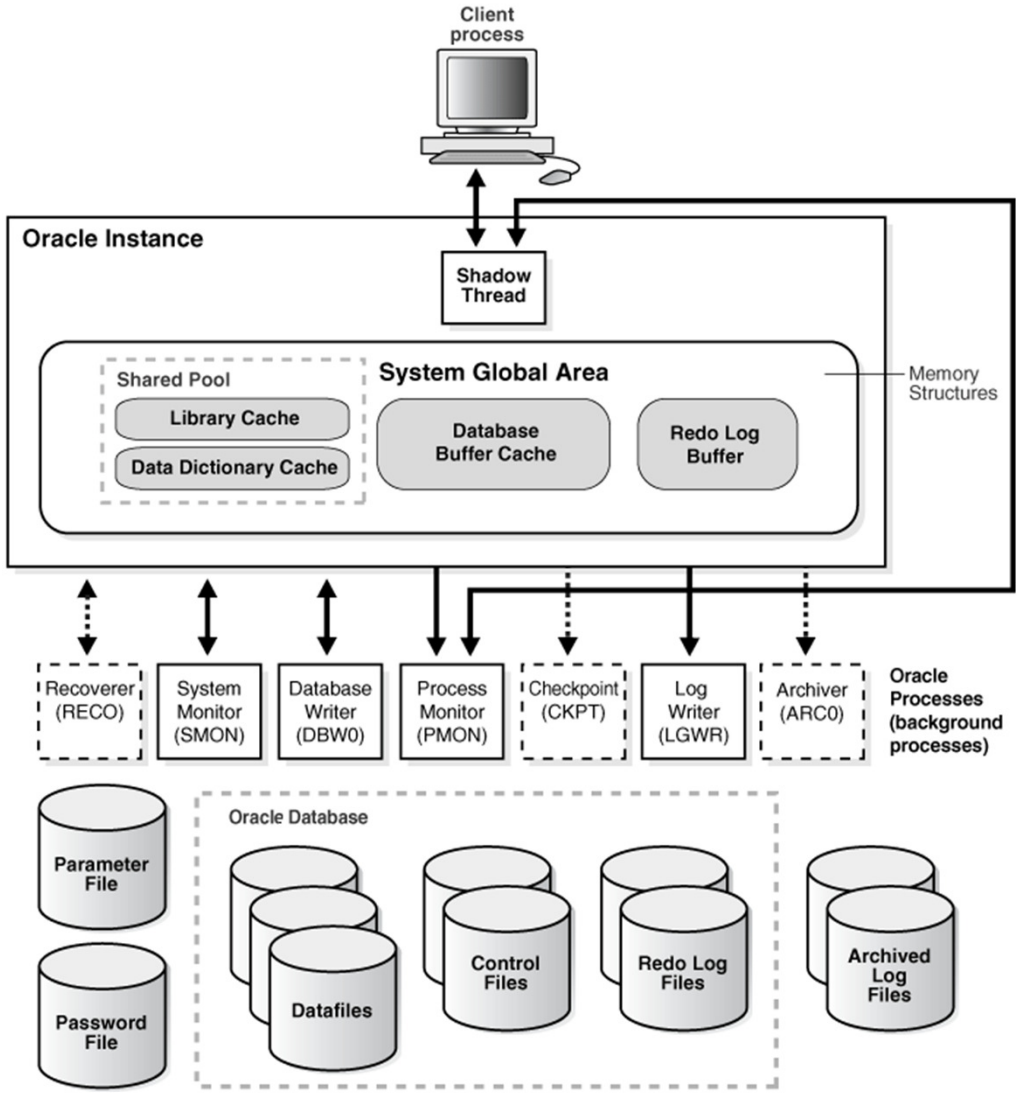
1.4.- Arquitecturas web.

Arquitectos de sistemas: conjugan distintos elementos hardware (máquinas y otros dispositivos) con elementos software (sistemas operativos) para construir sistemas capaces de ofrecer los recursos que necesitan las distintas aplicaciones o servicios destinados a correr sobre ellos.



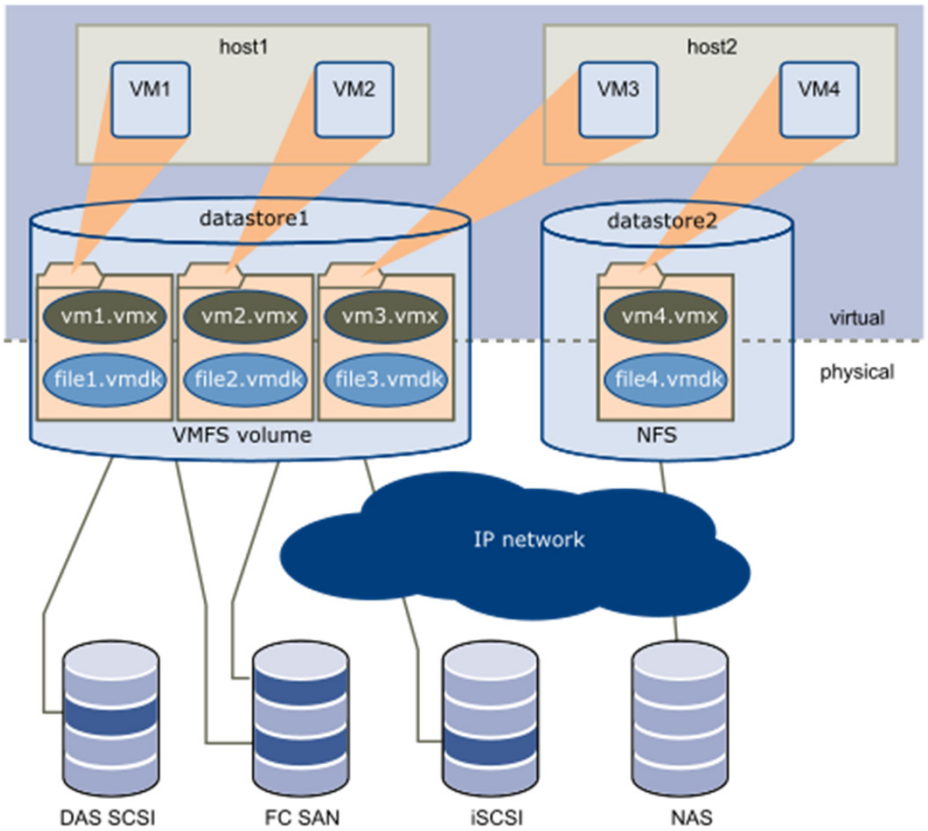
1.4.- Arquitecturas web.

Arquitectos de datos: diseñan cómo se va a estructurar la información manejada por las aplicaciones mediante el uso de bases de datos.



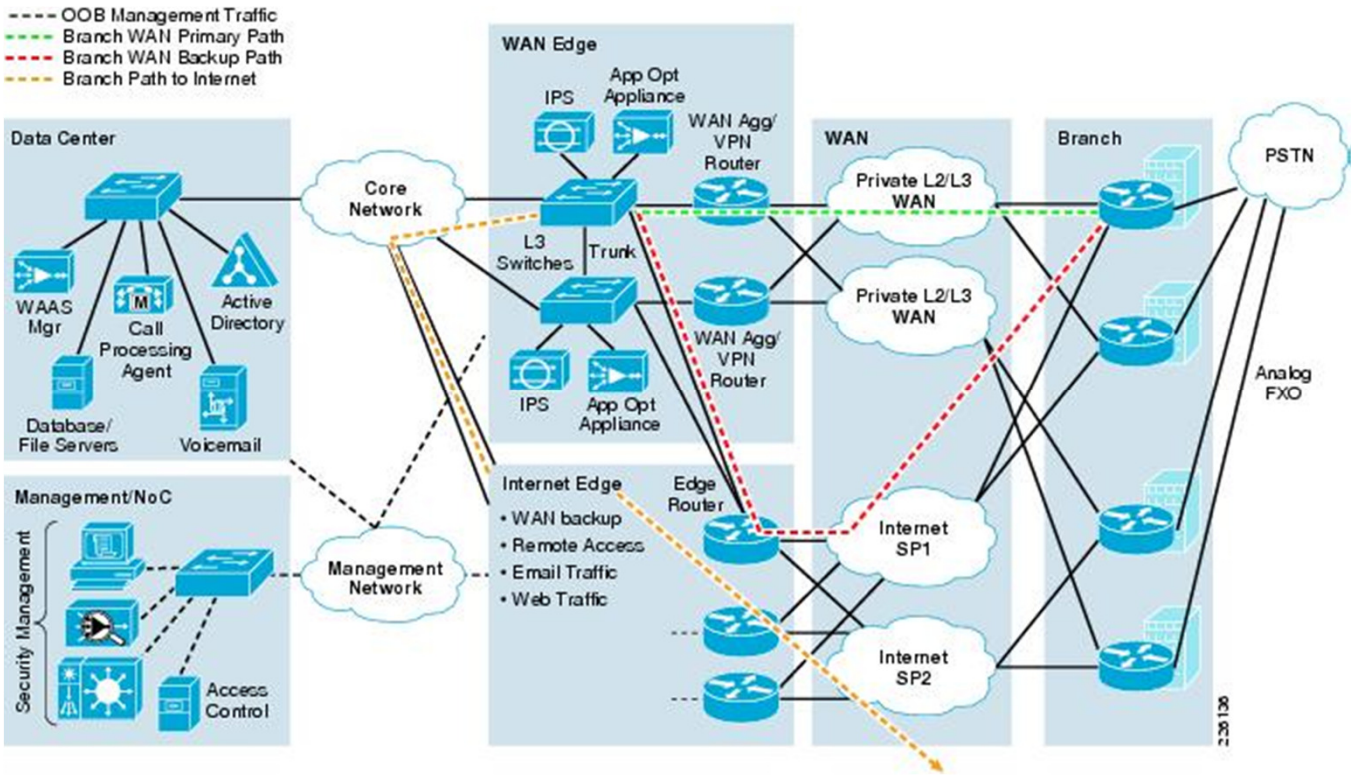
1.4.- Arquitecturas web.

Arquitectos de almacenamiento: diseñan redes de almacenamiento (SAN) que permitan almacenar toda la información que generan las distintas aplicaciones que corren sobre los sistemas de información.



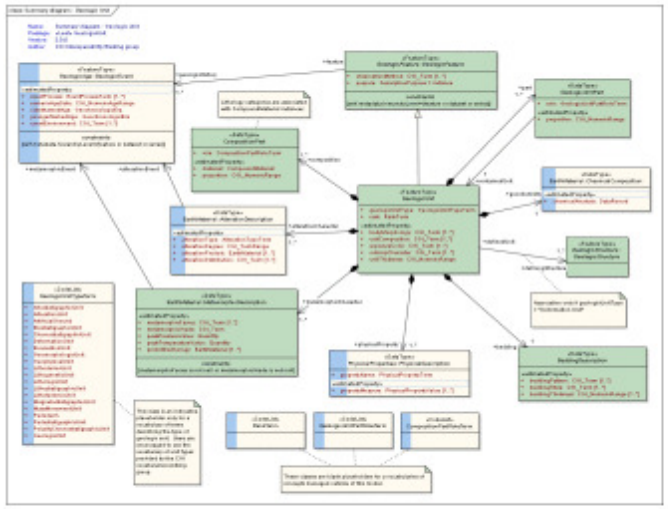
1.4.- Arquitecturas web.

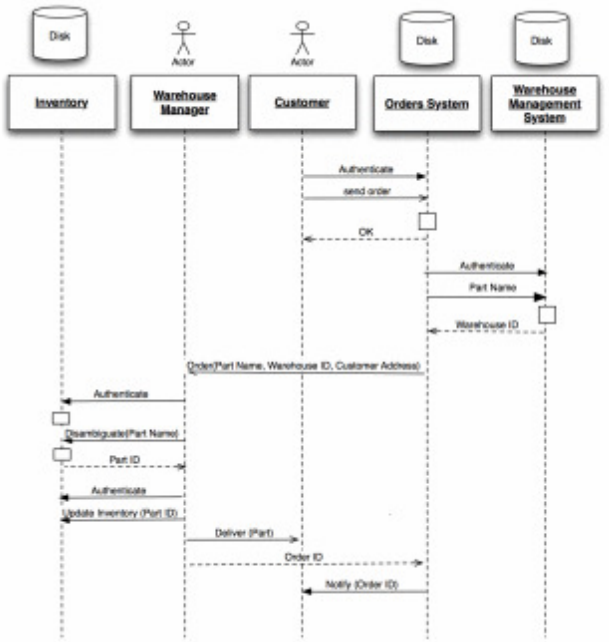
Arquitectos de redes: planean y diseñan las redes de comunicación que permitan el intercambio de datos entre distintos sistemas de información.



1.4.- Arquitecturas web.

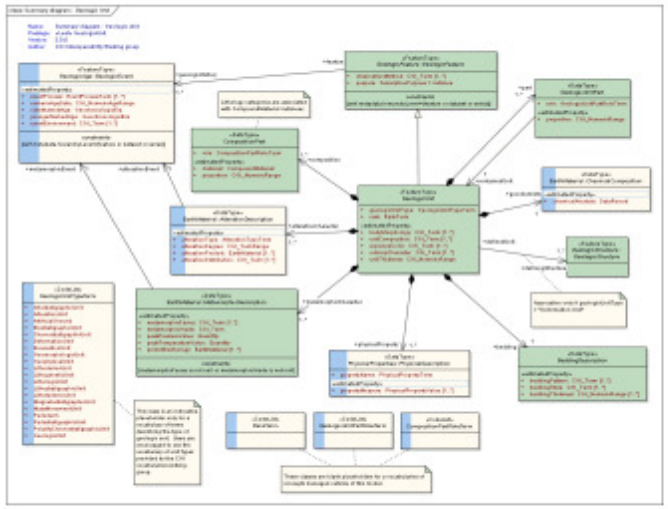
Arquitectos de software: más típicamente conocidos como ingenieros de software, diseñan y construyen las aplicaciones que van a permitir ofrecer el servicio que necesitan los usuarios de los sistemas de información u otras aplicaciones.

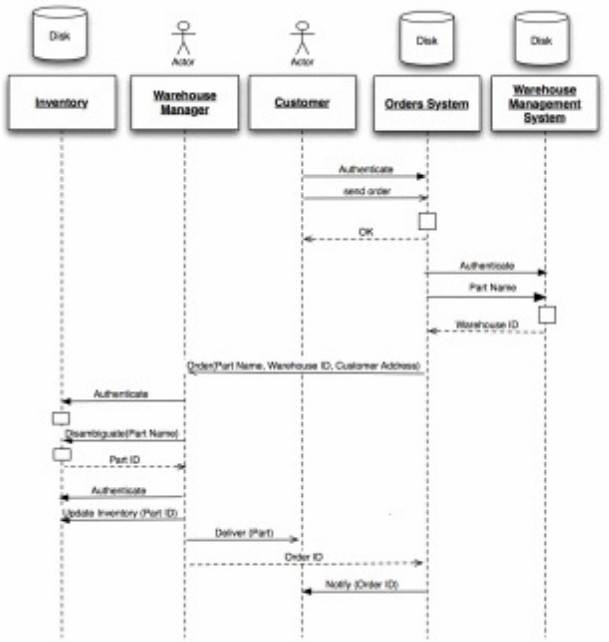




1.4.- Arquitecturas web.

Arquitectos de software: más típicamente conocidos como ingenieros de software, diseñan y construyen las aplicaciones que van a permitir ofrecer el servicio que necesitan los usuarios de los sistemas de información u otras aplicaciones.





1.4.- Arquitecturas web.

Arquitectura web

Sería en este último grupo donde se encuadrarían los **arquitectos web**, como un subgrupo de los arquitectos de software especializado en diseñar y contruir aplicaciones que se van a utilizar a través de lo que conocemos como la Web, es decir, haciendo uso del protocolo HTTP para comunicarse con el usuario o con otras aplicaciones web.

Las competencias que genuinamente son propias de un arquitecto web son las siguientes:

Diseño de la interfaz de usuario de la aplicación web. En el caso de un sitio web se referiría al diseño de la propia web, tanto su aspecto visual (colores, imágenes, tipografía empleada, posicionamiento de los distintos bloques de contenido dentro de las distintas páginas, etc.), como de la estructuración de los contenidos en diversas secciones y apartados enlazables a través de un menú con las distintas opciones disponibles. Aquí entrarían en juego distintas disciplinas como las del diseño gráfico, la usabilidad, la experiencia de usuario (UX), la interacción usuario-máquina, los mapas del sitio o mapas web, etc., así como distintos términos como HTML5, CSS, DOM, Javascript, AJAX, estándares web, etc.

Diseño e implementación de la lógica de la aplicación, es decir, del conjunto de funcionalidades que ofrecerá ésta, como el procesamiento de los datos introducidos por el usuario, el cálculo de resultados a partir de distintos datos de entrada, el diseño y ejecución de algoritmos, la manipulación de la información almacenada en una base de datos, la ejecución de diversas acciones como consecuencia del cumplimiento de diversas condiciones o del disparo de algún evento, etc. Es decir, planear y diseñar lo que luego se llevará a cabo mediante el uso de uno o varios lenguajes de programación.

Diseño de la arquitectura de la información, es decir, determinar la información del mundo real que tendrá que tratar una aplicación, diseñar un modelo conceptual que sea un fiel reflejo de dicho mundo real con sus distintas entidades y relaciones, determinar el modelo de datos que mejor se adapte a dicho modelo conceptual, implementar ese modelo de datos sobre un motor de bases de datos concreto y trasladar a él la información necesaria para el correcto funcionamiento de nuestra aplicación.

1.4.2. - Arquitecturas web. Modelos.

Modelos de arquitecturas web

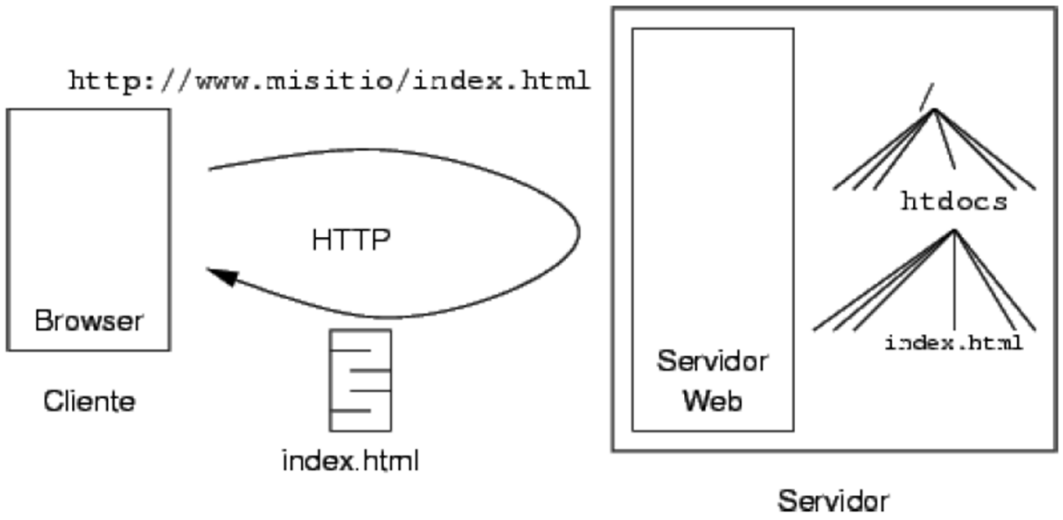
1.4.3. - Arquitectura de las aplicaciones Web.

Una aplicación Web es proporcionada por un servidor Web y utilizada por usuarios que se Conectan desde cualquier punto vía clientes Web (browsers o navegadores).

La arquitectura de un Sitio Web tiene tres componentes principales:

Un servidor Web
Una conexión de red
Uno o más clientes

El servidor Web distribuye páginas de información formateada a los clientes que las solicitan. Los requerimientos son hechos a través de una conexión de red, y para ello se usa el protocolo HTTP. Una vez que se solicita esta petición mediante el protocolo HTTP y la recibe el servidor Web, éste localiza la página Web en su sistema de archivos y la envía de vuelta al navegador que la solicitó.



1.4.3. - Arquitectura de las aplicaciones Web.

Las aplicaciones Web están basadas en el modelo Cliente/Servidor que gestionan servidores web, y que utilizan como interfaz páginas web.

Las páginas Web son el componente principal de una aplicación o sitio Web. Los browsers piden páginas (almacenadas o creadas dinámicamente) con información a los servidores Web. En algunos ambientes de desarrollo de aplicaciones Web, las páginas contienen código HTML y scripts dinámicos, que son ejecutados por el servidor antes de entregar la página.

Una vez que se entrega una página, la conexión entre el browser y el servidor Web se rompe, es decir que la lógica del negocio en el servidor solamente se activa por la ejecución de los scripts de las páginas solicitadas por el browser (en el servidor, no en el cliente). Cuando el browser ejecuta un script en el cliente, éste no tiene acceso directo a los recursos del servidor. Hay otros componentes que no son scripts, como los applets (una aplicación especial que se ejecuta dentro de un navegador) o los componentes ActiveX. Los scripts del cliente son por lo general código JavaScript o VBSscript, mezclados con código HTML.

La colección de páginas son en una buena parte dinámicas (ASP, PHP, etc.), y están agrupadas lógicamente para dar un servicio al usuario. El acceso a las páginas está agrupado también en el tiempo (sesión). Los componentes de una aplicación Web son:

1. **Lógica de negocio.**

   Parte más importante de la aplicación.

   Define los procesos que involucran a la aplicación.

   Conjunto de operaciones requeridas para proveer el servicio.

1.4.3. - Arquitectura de las aplicaciones Web.

2. Administración de los datos.

Manipulación de BD y archivos.

3. Interfaz

Los usuarios acceden a través de navegadores, móviles, PDAs, etc.

Funcionalidad accesible a través del navegador.

Limitada y dirigida por la aplicación.

Las aplicaciones web se modelan mediante lo que se conoce como modelo de capas, Una capa representa un elemento que procesa o trata información. Los tipos son:

Modelo de dos capas: La información atraviesa dos capas entre la interfaz y la administración de los datos.

Modelo de n-capas: La información atraviesa varias capas, el más habitual es el modelo de tres capas.

1.4.3. - Arquitectura de las aplicaciones Web.

Modelo de dos Capas.

Gran parte de la aplicación corre en el lado del cliente (fat client). Las capas son:

1. Cliente (fat client): La lógica de negocio está inmersa dentro de la aplicación que realiza el interfaz de usuario, en el lado del cliente.

2. Servidor: Administra los datos.

Las limitaciones de este modelo son:

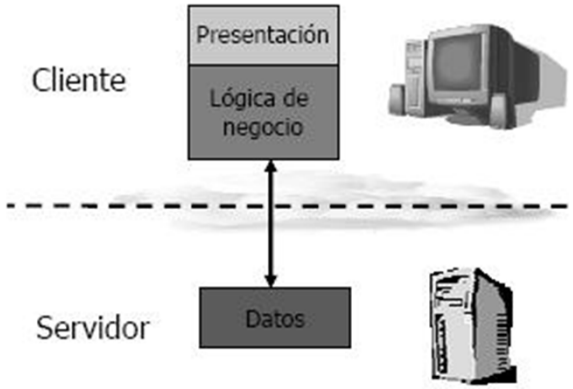
Es difícilmente escalable

Número de conexiones reducida

Alta carga de la red

La flexibilidad es restringida

La funcionalidad es limitada



1.4.3. - Arquitectura de las aplicaciones Web.

Modelo de tres Capas.

Esta diseñada para superar las limitaciones de las arquitecturas ajustadas al modelo de dos capas, introduce una capa intermedia (la capa de proceso) Entre presentación y los datos, los procesos pueden ser manejados de forma separada a la interfaz de usuario y a los datos, esta capa intermedia centraliza la lógica de negocio, haciendo la administración más sencilla, los datos se pueden integrar de múltiples fuentes, las aplicaciones web actuales se ajustan a este modelo.

Las capas de este modelo son:

1. **Capa de presentación** (parte en el cliente y parte en el servidor)

   Recoge la información del usuario y la envía al servidor (cliente)

   Manda información a la capa de proceso para su procesado

   Recibe los resultados de la capa de proceso

   Generan la presentación

   Visualizan la presentación al usuario (cliente)

1.4.3. - Arquitectura de las aplicaciones Web.

Modelo de tres Capas.

2. Capa de proceso (servidor web)

Recibe la entrada de datos de la capa de presentación

Interactúa con la capa de datos para realizar operaciones

Manda los resultados procesados a la capa de presentación

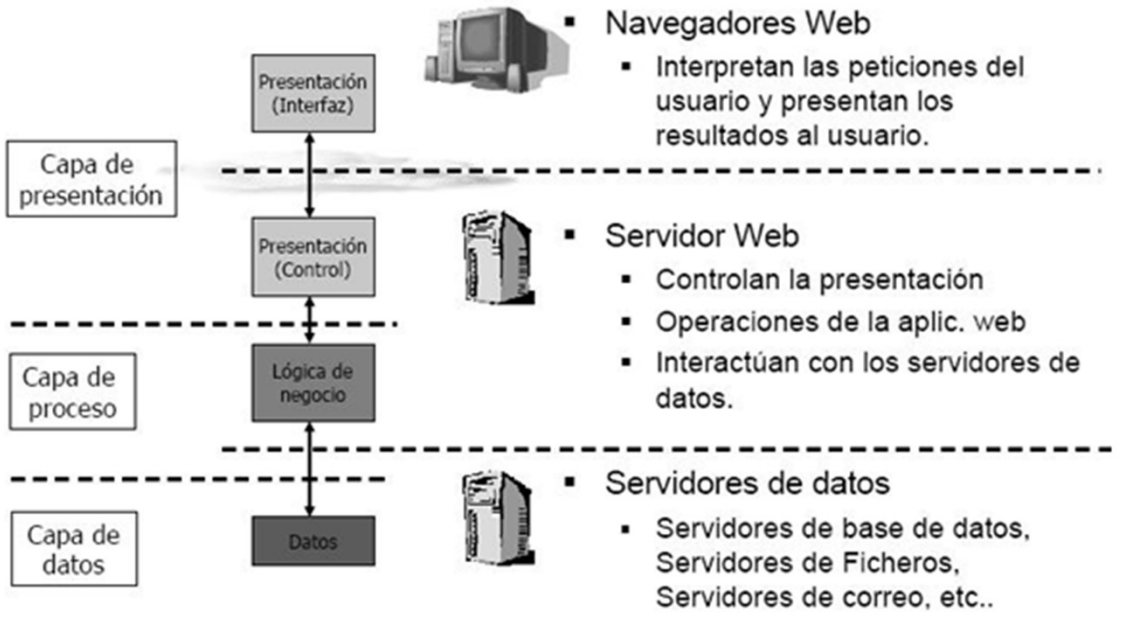
3. Capa de datos (servidor de datos)

Almacena los datos

Recupera datos

Mantiene los datos

segura la integridad de los datos



1.5.- Escalabilidad.

Las aplicaciones web se ejecutan en un entorno donde el número de clientes que solicitan el servicio puede variar en gran medida en función del momento. Es por ello que hay una característica de esencial importancia como es la escalabilidad.

El éxito o el fracaso de un sitio web orientado al usuario común vendrá determinado, entre otros aspectos, por el dimensionamiento del sistema sobre el que se instala y soporta el software que sustenta dicho sitio. En consecuencia, uno de los requisitos fundamentales de una aplicación web es que sea completamente escalable sin que un aumento de los recursos dedicados a la misma suponga modificación alguna en su comportamiento o capacidades.

La escalabilidad de un sistema web puede ser:

- **Vertical**: de manera ascendente "upgrades" a cada nodo.
- **Horizontal**: consiste en aumentar el número de nodos.
- **Cluster**: consiste en crear agrupaciones de servidores.

1.6.- Plataformas web libres y propietarias.

Una plataforma web es el entorno de desarrollo de software empleado para diseñar y ejecutar un sitio web. En términos generales, una plataforma web consta de cuatro componentes básicos:

1. El **sistema operativo,** bajo el cual opera el equipo donde se hospedan las páginas web y que representa la base misma del funcionamiento del computador. En ocasiones limita la elección de otros componentes.

2. El **servidor web** es el software que maneja las peticiones desde equipos remotos a través de la Internet. En el caso de páginas estáticas, el servidor web simplemente provee el archivo solicitado, el cual se muestra en el navegador. En el caso de sitios dinámicos, el servidor web se encarga de pasar las solicitudes a otros programas que puedan gestionarlas adecuadamente.

3. El **gestor de bases de datos** se encarga de almacenar sistemáticamente un conjunto de registros de datos relacionados para ser usados posteriormente.

4. **Un lenguaje de programación interpretado** que controla las aplicaciones de software que corren en el sitio web.

Diferentes combinaciones de los cuatro componentes señalados, basadas en las distintas opciones de software disponibles en el mercado, dan lugar a numerosas plataformas web, aunque, sin duda, hay dos que sobresalen del resto por su popularidad y difusión: LAMP y WISA.

La plataforma **LAMP** trabaja enteramente con componentes de **software libre** y no está sujeta a restricciones propietarias. El nombre **LAMP** surge de las iniciales de los componentes de software que la integran:
Linux: Sistema operativo.
Apache: Servidor web.
MySQL: Gestor de bases de datos.
PHP: Lenguaje interpretado PHP, aunque a veces se sustituye por Perl o Python.

La plataforma **WISA** está basada en tecnologías desarrolladas por la compañía Microsoft; se trata, por lo tanto, de **software propietario.** La componen los siguientes elementos:
Windows: Sistema operativo.
Internet Information Services: servidor web.
SQL Server: gestor de bases de datos.
ASP o ASP.NET: como lenguaje para scripting del lado del servidor.

DESPLIEGUE DE APLICACIONES WEB

TEMA 1. Implantación de arquitecturas web.

2.- Servidor web Apache.

2.- Servidor web Apache.

Un servidor web es un programa que se ejecuta de forma continua en un ordenador (también se utiliza el término para referirse al ordenador que lo ejecuta), se mantiene a la espera de peticiones por parte de un cliente (un navegador de Internet) y contesta a estas peticiones de forma adecuada, sirviendo una página web que será mostrada en el navegador o mostrando el mensaje correspondiente si se detectó algún error.

Uno de los servidores web más populares del mercado y el más utilizado actualmente es **Apache**, de código abierto y gratuito, disponible para Windows y GNU/Linux, entre otros.

En cuanto a su arquitectura podemos destacar lo siguiente:
- Estructurado en módulos.
- Cada módulo contiene un conjunto de funciones relativas a un aspecto concreto del servidor.
- El archivo binario **httpd** contiene un conjunto de módulos que han sido compilados.
- La funcionalidad de estos módulos puede ser activada o desactivada al arrancar el servidor.
- Los módulos de Apache se pueden clasificar en tres categorías:

  - ✓ Módulos base: Se encargan de las funciones básicas.
  - ✓ Módulos multiproceso: Encargados de la unión de los puertos de la máquina, aceptando las peticiones y atendiéndolas.
  - ✓ Módulos adicionales: se encargan de añadir funcionalidad al servidor.

  El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation. La licencia de software, bajo la cual el software de la fundación Apache es distribuido, es una parte distintiva de la historia de Apache HTTP Server y de la comunidad de código abierto.

  La **Licencia Apache** permite la distribución de derivados de código abierto y cerrado a partir de su código fuente original.

2.1.- Instalación y configuración.

Empezamos por identificarnos en la máquina con el usuario **root** y, a continuación, ejecutamos:

apt-get install apache2

Debido a que pretendemos montar una plataforma LAMP, por sus ventajas derivadas de las características del software libre, instalaremos también los siguientes componentes: MySQL y PHP.

apt-get install php5 mysql-client mysql-admin mysql-query-browser phpmyadmin

Una vez instalado, para verificar si funciona, podemos hacerlo desde un navegador, escribiendo en la barra de direcciones :

http://localhost ó http://127.0.0.1

o bien, si accedemos desde otro equipo de la red a la dirección IP de esta máquina, deberíamos obtener una página con el mensaje "**It Works!**", confirmando así su correcto funcionamiento.

Si el puerto especificado en la directiva Listen del fichero de configuración es el que viene por defecto, es decir, el puerto 80 (o cualquier otro puerto por debajo del 1024), entonces es necesario tener privilegios de usuario root (superusuario) para iniciar Apache, de modo que pueda establecerse una conexión a través de esos puertos privilegiados.

Si en cualquier momento deseásemos parar, reiniciar o arrancar el servidor, podríamos emplear los siguientes comandos respectivamente**:**

sudo /etc/init.d/apache2 stop o **sudo service apache2 stop**
sudo /etc/init.d/apache2 restart o **sudo service apache2 restart**
sudo /etc/init.d/apache2 start o **sudo service apache2 start**
sudo /etc/init.d/apache2 reload o **sudo service apache2 reload**

Video de Instalación Apache + MySQL + PHP + PHPMyAdmin: **https://www.youtube.com/watch?v=juPhP1iHWSs**

Ejercicio: Tarea 2. Instalación Apache + MySQL + PHP + PHPMyAdmin

DESPLIEGUE DE APLICACIONES WEB

TEMA 1. Implantación de arquitecturas web.

3.- Aplicaciones web y servidores de aplicaciones.

3.- Aplicaciones web y servidores de aplicaciones.

Se define una aplicación web como una aplicación informática que se ejecuta en un entorno web, de forma que se trata de una aplicación cliente-servidor junto con un protocolo de comunicación previamente establecido:

- ✓ Cliente: navegador.
- ✓ Servidor: servidor web
- ✓ Comunicación: protocolo HTTP

Un **servidor de aplicaciones** es un software que proporciona aplicaciones a los equipos o dispositivos cliente, por lo general, a través de Internet y utilizando el protocolo http. Los servidores de aplicación se distinguen de los servidores web en el uso extensivo del contenido dinámico y por su frecuente integración con bases de datos.

Las principales ventajas de la tecnología de los servidores de aplicaciones es la **centralización y disminución** de la complejidad en el desarrollo de las aplicaciones, ya que no necesitan ser programadas, sino que son ensambladas desde bloques provistos por el servidor de aplicación.

Otra de las ventajas es la **integridad de datos y código** ya que, al estar centralizada en una o un pequeño número de máquinas servidoras, las actualizaciones están garantizadas para todos los usuarios.

El término servidor de aplicaciones se aplica a todas las plataformas. Dicho término se utiliza para referirse a los servidores de aplicaciones basadas en web, como el control de las plataformas de comercio electrónico integrado, sistemas de gestión de contenido de sitios web y asistentes o constructores de sitios de Internet.

3.1.- El servidor de aplicaciones Tomcat.

Tomcat es el servidor web (incluye el servidor Apache) y de aplicaciones del proyecto Jakarta, con lo cual, gestiona las solicitudes y respuestas http y, además, es servidor de aplicaciones o contenedor de Servlets y JSP.

Incluye el compilador Jasper, que compila JSP covirtiéndolas en servlets.

Tomcat es un contenedor de servlets con un entorno JSP. Un contenedor de servlets es un shell de ejecución que maneja e invoca servlets por cuenta del usuario.

Tomcat: https://es.wikipedia.org/wiki/Tomcat

Proyecto Jakarta
El **Proyecto Jakarta** crea y mantiene software de código abierto para la plataforma Java. Opera como un proyecto paraguas bajo el auspicio de la *Apache Software Foundation*, y todos los productos producidos por Jakarta son liberados bajo la Licencia Apache.

https://es.wikipedia.org/wiki/Jakarta_Project

JSP
JavaServer Pages (JSP) es una tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas basadas en HTML y XML, entre otros tipos de documentos. JSP es similar a PHP, pero usa el lenguaje de programación Java.

https://es.wikipedia.org/wiki/JavaServer_Pages

Servlets
El **servlet** es una clase en el lenguaje de programación Java, utilizada para ampliar las capacidades de un servidor. Aunque los servlets pueden responder a cualquier tipo de solicitudes, éstos son utilizados comúnmente para extender las aplicaciones alojadas por servidores web, de tal manera que pueden ser vistos como applets de Java que se ejecutan en servidores en vez de navegadores web. Este tipo de servlets son la contraparte Java de otras tecnologías de contenido dinámico Web, como PHP y ASP.NET.

https://es.wikipedia.org/wiki/Java_Servlet

Ejercicio: Tarea 3. Elabora un documento de texto con la definicion y las principales caracteristicas de los conceptos anteriores (Tomcat, Proyecto Jakarta, JSP, Servlets).

3.1.1.- Instalación y configuración básica.

En primer lugar, destacar que para instalar cualquier versión de Tomcat es necesario tener instalado JDK (Kit de desarrollo de Java), ya que el objetivo es que las peticiones a Apache se redirijan a Tomcat empleando un conector proporcionado por Java en este caso.

Para realizar esto buscaremos el paquete Java que más nos interese e instalamos para ello aptitude como buscador de versiones tecleando apt-get install aptitude y cuando lo tengamos instalado tecleamos:

aptitude search "?provides(java-runtime)"

Instalamos la primera versión que aparece, por lo que tecleamos:

apt-get install default-jre

Podemos ver que versión de JDK nos ha instalado mediante la instrucción:

java -version

Una vez instalado el paquete anterior, es necesario crear una variable de entorno para indicar en dónde se ha instalado, y añadir a la variable **PATH** el directorio en donde se encuentran los archivos binarios para que puedan ser invocados desde cualquier parte; para ello, añadimos en nuestro caso las siguientes líneas al archivo **/etc/profile**:

JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre/
PATH=$PATH:$JAVA_HOME/bin
export PATH JAVA_HOME

Por defecto, el propietario de este archivo es **root, p**ara poder editar el archivo **/etc/profile** necesitamos cambiar los permisos para nuestro usuario mediante el comando:

sudo chown NombredeUsuario:adm /etc/profile

Actualizamos las variables de entorno mediante el comando:

source /etc/profile

Comprobar que las variables de entorno se han exportado correctamente:

echo $JAVA_HOME

3.1.1.- Instalación y configuración básica.

Llegado este punto descargamos Tomcat, para ello abrimos en un navegador la URL:
http://ldp.rediris.es/mirror/apache/tomcat/tomcat-9/

Una vez allí comprobaremos cuál es la última versión estable de Tomcat y, desde la carpeta "bin", copiamos el link de descarga al **apache-tomcat-X.XX.X.tar.gz.**

En nuestro caso el link sería:
apache-tomcat-9.0.12.tar.gz

con lo cual podemos emplear el siguiente comando para descargarlo:
wget http://ldp.rediris.es/mirror/apache/tomcat/tomcat-9/v9.0.12/bin/apache-tomcat-9.0.12.tar.gz

Descomprimimos el archivo descargado:
tar xvzf apache-tomcat-9.0.12.tar.gz

Movemos a la carpeta de destino :
sudo mv apache-tomcat-9.0.12 /usr/local/

Podemos hacer un link para hacer más cómodas las actualizaciones:
sudo ln -s /usr/local/apache-tomcat-9.0.12/ /usr/local/tomcat

En Tomcat, la gestión del servicio se realiza a través del script incluido llamado **catalina**, al que le podemos proporcionar los parámetros "start" y "stop", con lo que arrancaríamos o pararíamos el servicio manualmente:
sh /usr/local/tomcat/bin/catalina.sh start

Para comprobar que nuestro servidor está ya escuchando, introducimos en un navegador la URL http://127.0.0.1:8080 o localhost:8080, y éste debería mostrar la página de inicio de Tomcat.

Ejercicio: Tarea 4. Instalación de Tomcat.

3.1.2.- Iniciar Tomcat.

Tomcat va a estar escuchando en el puerto 8080 y va a tener su propio directorio de trabajo. La misión de apache2 va a ser interceptar todas las peticiones en el puerto 80 y derivar las que considere necesarias a Tomcat.

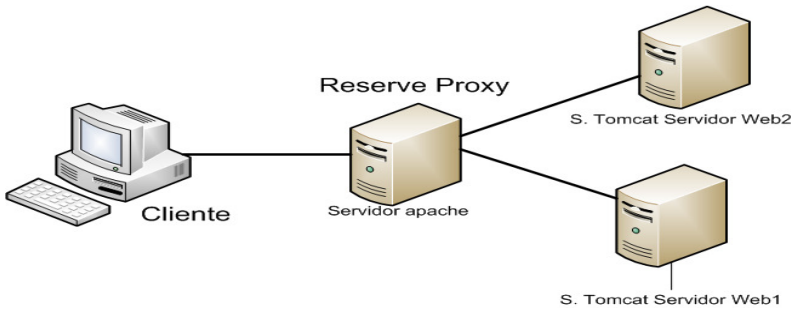
De este modo observamos la ventaja de la escalabilidad, ya que apache, al funcionar como proxy, puede tener una batería de tomcats a los que balancear las conexiones, haciendo que, si nuestras necesidades crecen, nuestras máquinas puedan ampliarse en número siendo completamente transparente para los usuarios.

Ahora tenemos en nuestro sistemas dos servicios que responden a solicitudes web, pero son diferentes. Por lo que debemos otorgar a cada uno su funcionalidad .

Podemos hacer que convivan juntos de la siguiente manera, la idea es la siguiente:

- Todas las peticiones la recibe el servidor de apache
  - Si es una solicitud web estática o php la resuelve apache
  - Si es una solicitud de una aplicación web (jsp, servlet) se la entrega a tomcat para que resuelva el
- Al final es apache el que devuelve la solicitud al cliente

Este funcionamiento es totalmente trasparente para el usuario y es útil para que apache balancee la carga entre diferentes servidores de aplicaciones web y aligerar el trabajo.



3.1.2.- Iniciar Tomcat.

Apache por defecto busca los ficheros en **/var/www/html**
Tomcat trabaja sobre la carpeta **/usr/local/tomcat/webapps/ROOT**

La petición de una url se puede gestionar, parte por apache y parte por Tomcat, por lo que vamos a cambiar la carpeta por defecto de trabajo para unificarlo. Para ello editamos el fichero **/usr/local/tomcat/conf/server.xml**

nano /usr/local/tomcat/conf/server.xml

en donde encontraremos una línea con "**Host name=**" y lo establecemos a:

<Host name="localhost" appBase="/var/www/html"

Configuraremos Apache como un proxy inverso en modo básico usando la extensión mod_proxy para redirigir las conexiones entrantes a uno o varios servidores backend. Para cargar módulos usamos el comando ***a2enmod*** *apache to enabled module,* este comando sirve para cargar módulos nuevos. Cargaremos los módulos siguientes:

sudo a2enmod proxy el módulo proxy núcleo Módulo Apache para redirigir conexiones.
sudo a2enmod proxy_http
sudo a2enmod proxy_ajp módulo que maneja el protocolo AJP para Tomcat como servidores back-end.
sudo a2enmod rewrite
sudo a2enmod deflate
sudo a2enmod headers
sudo a2enmod proxy_balancer agrega funciones de equilibrio de carga.
sudo a2enmod proxy_connect
sudo a2enmod proxy_html

A continuación rearrancamos el sistema:
sudo /etc/init.d/apache2 restart

3.1.2.- Iniciar Tomcat.

ajp es un protocolo de comunicación interno y muy rápido que usa conexiones TCP persistentes. Es este protocolo se utiliza para comunicar apache2 con Tomcat, aunque podría ser utilizado http, indicando que pregunte en el 8080.

El puerto de trabajo por defecto para Tomcat es el 8009, aunque este puede ser variado desde **/usr/local/tomcat/conf/server.xml**

3.1.2.- Iniciar Tomcat.

Directivas necesarias en *apache*
- ProxyPass
- ProxyPassReverse
- ProxyRequests
- ProxyPreserveHost.

ProxyPass
Esta directiva tiene dos parámetros: El primero Indicamos las URL que delegamos (/) y el segundo el servidor de aplicaciones que atenderá la solicitud. En nuestro caso especificamos tomcat_cluster que está definido previamente.

```
<Proxy balancer://tomcat_cluster>
Order allow,deny
Allow from all
BalancerMember ajp://localhost:8009
</Proxy>
```

Concepto del proxy y el protocolo ajp
En lugar de usar http para comunicar apache-tomcat usamos ajp : Es un protocolo interno, veloz y que usa conexiones ***TCP persistentes*** . Funciona en el puerto 8009

Proxy balancer://tomcat_cluster definimos el cluster con nombre ***tomcat_cluster***

BalancerMember ajp://localhost:8009
Se define un miembro a ***tomcat_cluster***
protocolo ***ajp***
IP ***localhost'*** o ***127.0.0.1***
Puerto ***8009*** que es el que por defecto usa *tomcat*

ProxyPass / balancer://tomcat_cluster/
"/" y todo lo que cuelgue de ella, sea pasado al cluster del tomcat para que lo procese él.

3.1.2.- Iniciar Tomcat.

Order allow,deny
Especifica el orden de asignación de permisos:
Primero las especificaciones de permisos ***allow all'***
Posteriormente restricciones especificadas en la opción ***deny***

Si en lugar de ***ajp*** queremos usar http deberíamos especificar:
http://localhost:8080/ en lugar de ajp://localhost:8009

ProxyPreserveHost On
Mantiene la cabecera http host original, en vez de reescribirla

Parte para apache, otra para tomcat
Queremos que apache devuelva los contenidos estáticos . Estas peticiones deben comenzar por /static/
Para que lo haga apache el segundo parámetro de esta directiva debe ser una negación ***!***.

Configuración de *virtualhost:* Modificamos el fichero de configuración del virtualhost que se pretenda utilizar, empleando el establecido por defecto. **nano /etc/apache2/sites-enabled/000-default** en donde añadimos lo siguiente:

<Proxy balancer://tomcat_cluster>
Order allow,deny
Allow from all
BalancerMember ajp://localhost:8009
</Proxy>
ProxyPreserveHost On
ProxyPass /phpmyadmin/ !
ProxyPass / balancer://tomcat_cluster/
ProxyPassReverse / balancer://tomcat_cluster/

Lo último es cambiar de **/etc/apache2/sites-enabled/000-default** el "**DocumentRoot**" y "**<Directory /var/www/>**" para que apunten a **/var/www/ROOT**, de esta manera podemos decidir qué parte gestiona cada aplicación desde un solo directorio.

3.1.2.- Iniciar Tomcat. CORRECTO

Descargar el archivo 000-deafult.conf de la moodle y susitituirlo en /etc/apache2/sites-available/000-default.conf

ProxyPassMatch ^(/.*\.jsp)$ http://localhost:8080/$1

Cualquier peticion con extension .jsp se la enviamos al Servidor Tomcat.

ProxyPass /tomcat ajp://127.0.0.1:8009/

Enviamos todo el contenido del directorio /tomcat al Servidor Tomcat

DESPLIEGUE DE APLICACIONES WEB

TEMA 1. Implantación de arquitecturas web.

4.- Estructura y despliegue de una aplicación web.

4.- Estructura y despliegue de una aplicación web.

Una **aplicación web** está compuesta de una serie de servlets, páginas jsp, ficheros html, ficheros de imágenes, ficheros de sonidos, texto, clases, etc.; de forma que todos estos recursos se pueden empaquetar y ejecutar en varios contenedores distintos.

Durante la etapa de desarrollo de una aplicación web se emplea la estructura de directorios, y posteriormente, en la etapa de producción, toda la estructura de la aplicación se empaqueta en un archivo **.war**

El código necesario para ejecutar correctamente una aplicación web se encuentra distribuido en una estructura de directorios, agrupándose en ficheros según su funcionalidad. Un ejemplo de la estructura de carpetas de una aplicación web puede ser el siguiente:

/index.jsp
/WebContent/jsp/welcome.jsp
/WebContent/css/estilo.css
/WebContent/js/utils.js
/WebContent/img/welcome.jpg
/WEB-INF/web.xml
/WEB-INF/struts-config.xml
/WEB-INF/lib/struts.jar
/WEB-INF/src/com/empresa/proyecto/action/welcomeAction.java
/WEB-INF/classess/com/empresa/proyecto/action/welcomeAction.class

4.1.- Archivos WAR.

Su nombre procede de Web Application Archive (Archivo de Aplicación Web); permiten empaquetar en una sola unidad aplicaciones web de Java completas, es decir su contenido:

- Servlets y JSP.
- Contenido estático: HTML, imágenes, etc.
- Otros recursos web.

Aportan como ventaja, la simplificación del despliegue de aplicaciones web, debido a que su instalación es sencilla y solamente es necesario un fichero para cada servidor en un cluster, además de incrementar la seguridad ya que no permite el acceso entre aplicaciones web distintas.

Su estructura es la siguiente:

- **/**: En la carpeta raíz del proyecto se almacenan elementos empleados en los sitios web, tipo documentos html, CSS y los elementos JSP (*.html *.jsp *.css).
- **/WEB-INF/**: Aquí se encuentran los elementos de configuración del archivo **.WAR** como pueden ser: la página de inicio, la ubicación de los servlets, parámetros adicionales para otros componentes. El más importante de éstos es el archivo **web.xml**.
- **/WEB-INF/classes/**: Contiene las clases Java empleadas en el archivo .WAR y, normalmente, en esta carpeta se encuentran los servlets.
- **/WEB-INF/lib/**: Contiene los archivos JAR utilizados por la aplicación y que normalmente son las clases empleadas para conectarse con la base de datos o las empleadas por librerías de JSP.

Para generar archivos **.WAR** se pueden emplear diversas herramientas desde entorno IDE "Integrated Development Environment". Por ejemplo, encontramos: **NetBeans** y **Eclipse**, ambos Open-Source.

4.2.- Despliegue de aplicaciones con Tomcat

Una aplicación web puede ser desplegada empleando uno de los siguientes métodos:

- Por medio de archivos WAR (Web Archive).
- Editando los archivos web.xml y server.xml. Este método es el que se pasa a tratar a continuación.

Los directorios que forman una aplicación compilada suelen ser : **www**, **bin**, **src**, **tomcat** y **gwt-cache**.