**TEMA 1: IMPLANTACION DE ARQUITECTURAS WEB**

1. **ARQUITECTURA WEB**

La **www (world wide web)** es un **servicio de distribución de información** que:

* Permite el **acceso via internet a recursos y aplicaciones** distribuidos en servidores
  + Los recursos están **localizados en diferentes URL** e identificados de forma única por una ip (URI)
  + Los recursos **pueden conectarse entre si a través de enlaces**

**El esquema de funcionamiento de los servidores web** es:

1. **Proveedor** del servicio web

Es quien lo **diseña**, **desarrolla** e **implementa**

1. **Consumidor** del servicio

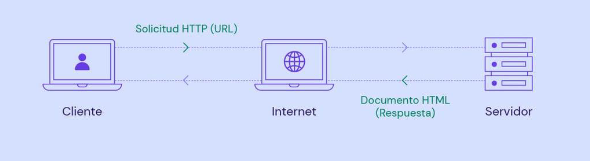
Quien accede para **utilizar** los servicios

1. **Agente** del servicio

**Sirve como enlace** entre proveedor y consumidor para efectos de publicación, búsqueda y localización

La arquitectura www **proviente del modelo cliente-servidor**

**MODELO CLIENTE-SERVIDOR (3 CAPAS)**

****

En un sistema cliente-servidor, **las aplicaciones y el contenido** (como sitios web o archivos) **se presentan en un formato estándar**. El **navegador** es la aplicación que **usamos para pedir este contenido.**

El navegador **envía una solicitud al servidor**, y el servidor **responde con la información solicitada** en un **formato que el navegador puede mostrar**

Para utilizar los **formatos estándar** hay que tener en cuenta los **estándares que define la W3C** para la world wide web (WWW)

* **Modelo estándar de nombres (URL)**

Todos los servidores y el contenido de la WWW **se denominan según un localizador uniforme de recursos (URL)**

**Diferencia entre URI y URL:**

**URL** especifica la **ubicación de un recurso** en **internet**

**URI** se utiliza para identificar cualquier tipo de recurso **no solo los que se encuentran en internet**

* **Formato de contenidos estándar**

Todos **los navegadores soportan** un conjunto de **formatos estándar** como **HTML, XML, JavaScript**

* **Protocolo estándar**

Permiten que **cualquier navegador pueda comunicarse con cualquier servidor web**

El mas usado es **HTTP**

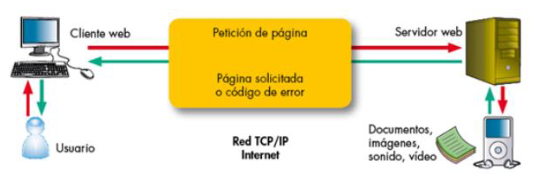
1. **HTTP**

Permite realizar la **tranferencia de información entre un cliente web y un servidor web** **en internet**, dando lugar a lo que se conoce como World Wide Web.

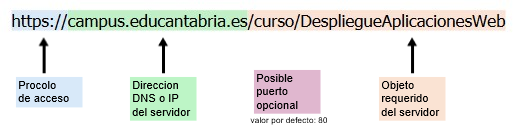
**Caracteristicas principales**

* Es un protocolo del **nivel de aplicación**, esta destinado a que **aplicaciones** en **sistemas diferentes** **se comuniquen** entre si
* Es un **protocolo sin estado**, **no hay consciencia de las peticiones anteriores**, simplemente **se hace una petición de un recurso y se obtiene el recurso**
* Las **peticiones** se hacen **en formato de texto legible** por un ser humano
* **No es necesario autenticarse** para realizar la petición de un recurso

**2.1 Funcionamiento del protocolo HTTP**



1. **Un** **usuario accede a una URL**, seleccionan un enlace de un documento HTML o introduciéndola en el campo correspondiente
2. **El cliente web descodifica la URL** separando sus diferentes partes



1. **Se abre una conexión TCP o UDP** (dependiendo de la versión HTTP) con el servidor llamado al **puerto correspondiente**

En ese momento **se realiza la petición HTTP**, para ello se envia:

* el **comando necesario** (GET, POST, HEAD … )
* la **dirección del objeto requerido** (el contenido de la URL que sigue a la dirección del servidor)
* la **versión del protocolo HTTP** empleado
* un **conjunto variable información** que incluye datos sorbre las capacidades del navegador, datos opcionales para el servidor etc…

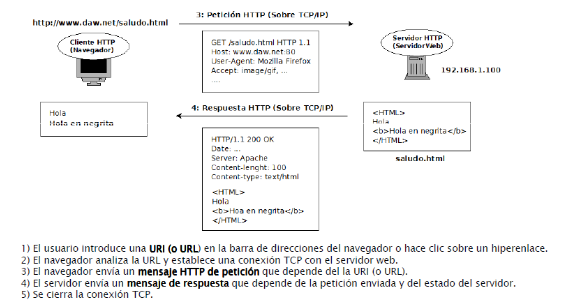
1. **El servidor devuelve la respuesta al cliente.**

* **código de estado** (200 OK, 404 Not Found..)
* **tipo de dato MIME** de la información de retorno (text/html, image/jpeg…)
* **la propia información** (contenido de la pagina web, imagen etc….)

1. **Se cierra la conexión TCP o UDP**

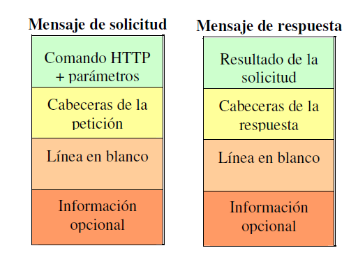
Esto **se repite en cada acceso al servidor HTTP**

Por ejemplo, si se recoge un documento HTML en cuyo interior están insertadas 2 imágenes y 1 vídeo, el proceso anterior se repite cuatro veces, una para el documento HTML y tres más para los recursos (la dos imágenes y el vídeo).



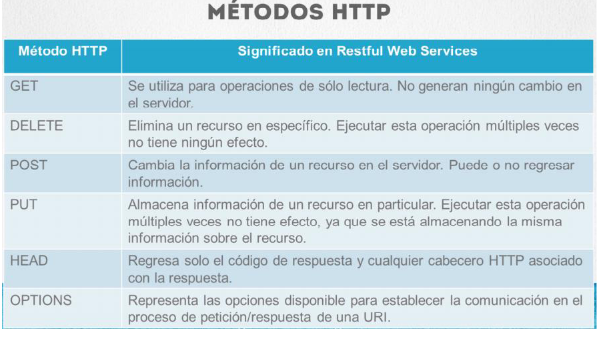
El **diálogo con los servidores HTTP** se establece a través de **mensajes formados por líneas de texto con instrucciones específicas**

Solo existen **dos tipos de mensajes**, unos para **realizar las peticiones** y otro **para las respuestas.**

****

**2.2 Métodos HTTP**

Se utilizan para **indicar la acción** que se desear realizar **para un recurso**

****

**Después de realizar la petición** el servidor genera una **respuesta HTTP** que contendrá un **código de estado de respuesta**

* **1XX**: Respuestas **informativas**
* **2XX**: **Éxito** en la transferencia
* **3XX**: **Redirección**, el recurso a cambiado de ubicación
* **4XX**: **Error** en la petición del cliente
* **5XX**: El servidor **no ha podido completar la respuesta**

**2.3 HTTPS**

Es el protocolo **seguro** de transferencia de hipertexto, es la **versión segura de HTTP**

**Para asegurar la información**, el protocolo HTTPS **requiere de certificados** y **si son validados la información será transferida cifrada**

Cifrar la información **requiere un tiempo de computación**, por lo que será perjudicado el rendimiento del servidor



1. **TECNOLOGIAS ASOCIADAS A LAS APLICACIONES WEB**

Las tecnologías asociadas a las aplicaciones Web que se ejecutarán tanto del lado

del servidor como del cliente:

* **ASP (Active Server Pages)**
  + Las paginas activas **se ejecutan del lado del servidor**, lo que siginifica que **es procesado en el servidor antes de enviar el resultado al navegador del usuario**
  + Un **ejemplo** de uso de ASP son los **buscadores**, donde **un usuario realiza una petición** de información y **el servidor nos entrega un resultado** a medida de nuestra petición
* **PHP (Hypertext Preprocessor)**
  + Este lenguaje es **ejecutado en el lado del servidor**
  + Es **similar a ASP** y puede ser usado en circunstancias similares
  + **Permite el acceso a bases de datos** empleando servidores como MySQL y, por lo tanto, suele utilizarse para crear páginas dinámicas complejas.
* **CGI** **(Common Gateway Interface)**
  + Las rutinas de CGI son habitualmente escritas en lenguajes interpretados como Perl o por lenguajes compilados como C.
  + Es utilizado para bases de datos, motores de búsqueda, formularios, generadores de mail automático, foros, comercio electrónico, rotadores y mapas de imágenes, juegos en línea, etc.
* **CSS (Cascading Style Sheets)**
  + Se usan **para formatear las páginas Web**
* **Java**
  + **Se ejecuta en el navegador del equipo cliente y no en el servidor**
  + Una misma aplicación **puede funcionar en diversos tipos de ordenadores y sistemas operativos**
  + **Tipos de programas en java:**
    - **Aplicaciones independientes**: Programas completos que se ejecutan en una ventana propia, igual que cualquier otro software
    - **Applets**: Pequeños programas que se incrustan dentro de una página web y se ejecutan directamente en el navegador.
  + **Orientado a objetos**: Significa **que los programas se construyen** **usando módulos** llamados **objetos**.

Cada uno de estos módulos es **independiente** y tiene sus propias funcionalidades.

**Facilita el desarrollo**, la actualización y la reutilización del código

* **JavaScript**
  + **Se interpreta y se ejecuta en el cliente**
  + Util para realizar tareas como mover imágenes por la pantalla, menus de navegación, juegos etc…
  + En las **paginas web** suele **preferirse JavaScript**
* **VBScript (Visual Basic Scripting)**
  + Es una buena herramienta para cualquier sitio destinado a ser **mostrado exclusivamente en el navegador Microsoft Internet Explorer**
  + **El código** puede, estar diseñado para su ejecución **en el lado del cliente o en el del servido**

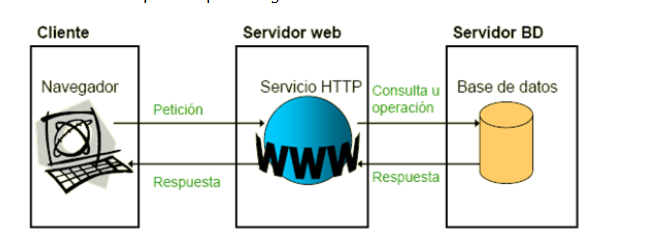
De forma genérica podríamos decir que **la arquitectura Web es un modelo compuesto de tres capas:**

* **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.
* **Capa de servidores de aplicaciones Web**: ejecutando **aplicaciones de tipo Apache, Tomcat, etc.**
* **Capa de clientes del servicio Web:** al que accederían **mediante un navegador Web** como Firefox, Chrome etc.

1. **MODELOS DE CAPAS**

**4.1 Modelo de 3 capas:**

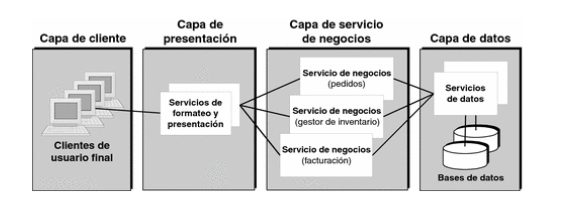
* **Capa de presentación (o cliente)**
  + **La que se presenta al usuario**
  + **Navegabilidad**, validación de los datos de entrada, **formateo** de los datos de salida, **presentación** de la web etc…
* **Capa de negocio (o servidor web)**
  + La que **recibe peticiones del usuario** y **desde donde se le envían las respuestas**
  + Se **verifica que las reglas establecidas se cumplen**
* **Capa de acceso a datos (o servidor BD)**
  + Es la **formada por determinados gestores de datos**
  + Se encargan de **almacenar**, **estructurar** y **recuperar** los daos solicitados por la capa de negocio

****

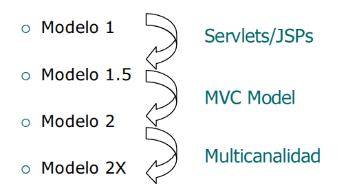
**4.2 Modelo de 4 capas**

Añade una capa al modelo de 3 capas

* **Capa de servicios**
  + Permite **desacoplar la interfaz de usuario del resto de capas** permitiendo que las **funcionalidades de nuestra aplicación sean accesibles por otras aplicaciones o servicios**
  + Con la arquitectura de 4 capas no se plantea el desarrollo de una aplicación al uso, si no **el desarrollo de un sistema compuesto por servicios que interactúan**



**4.3 Evolución de los modelos de arquitectura web**



* **Modelo 1**: Se **diseñan** en un **modelo web cgi**, cuya **salida** por pantalla era **HTML**
* **Modelo 1.5**: Aplicado a la tecnología java, las **paginas JSP** se encargan de la **presentación** y los **beans** del **modelo de negocio y acceso a datos**
* **Modelo 2**: Los **javabeans** se encargan de el **modelo de negocio** y para la **navegación** se usa el **patrón** **MVC**
* **Modelo 2.X**: Pueden desarrollar **aplicaciones multicanal**, que puedan ser **atacadas desde distintos tipos de clientes remotos**

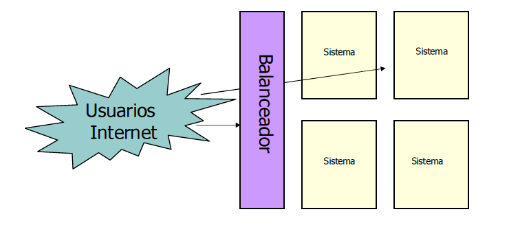
Se emplean **plantillas** **XSL** para **transformar los datos XML**

1. **ASPECTOS GENERALES DE LA ARQUITECTURA WEB**

* **Escalabilidad**
* **Separación de responsabilidades**
* **Portabilidad**
* **Utilización de componentes en los servicios de infraestructura**
* **Gestión de las sesiones del usuario**
* **Aplicación de patrones de diseño**
  1. **Escalabilidad**
* **El correcto dimensionamiento de la aplicación**
* La **adaptabilidad del sistema al incremento de la demanda**

**Varias opciones:**

* **Escalabilidad Horizontal**: añadir más recursos para dividir la carga



**El balanceador** es un **algoritmo** que **distribuye las peticiones HTTP entre los diferentes clones del sistema**

**La selección del clon es aleatoria**, de forma que no podemos garantizar que las distintas peticiones del mismo usuario sean atendidas por el mismo clon, por lo que **no se mantiene la sesión del usuario en el servidor**

**La sesión la debe mantener mediante el uso de Cookies o en base de datos**

* **Escalabilidad Vertical: Aumentar los recursos de un solo servidor** en lugar de distribuir la carga entre clones, para ello **se necesita un Middleware entre las capas** , que **permiten la comunicación remota**
  1. **Separación de responsabilidades**
* **Cada clase o componente** en el sistema **debe tener una única responsabilidad**

**Para ello se utiliza la arquitectura n-capas**

* + **Capa de Presentación**
    - **Navegabilidad** del sistema
    - **Validación de datos** de entrada
    - **Formateo** de **datos** de **salida**
    - **Renderizado** de **presentación**
  + **Capa de negocio**
    - **Lógica de negocio** o dominio del sistema
    - **Resultado del análisis funcional**: Conjunto de reglas de negocio que abstraen del mundo real
    - Debe ser independiente de la capa de presentación
  + **Capa de datos**
    - **Responsabilidades de lógica de persistencia**: Inserción, eliminación, actualización, búsquedas, etc
    - No tiene por qué ser siempre una base de datos relacional

**5.3 Portabilidad**

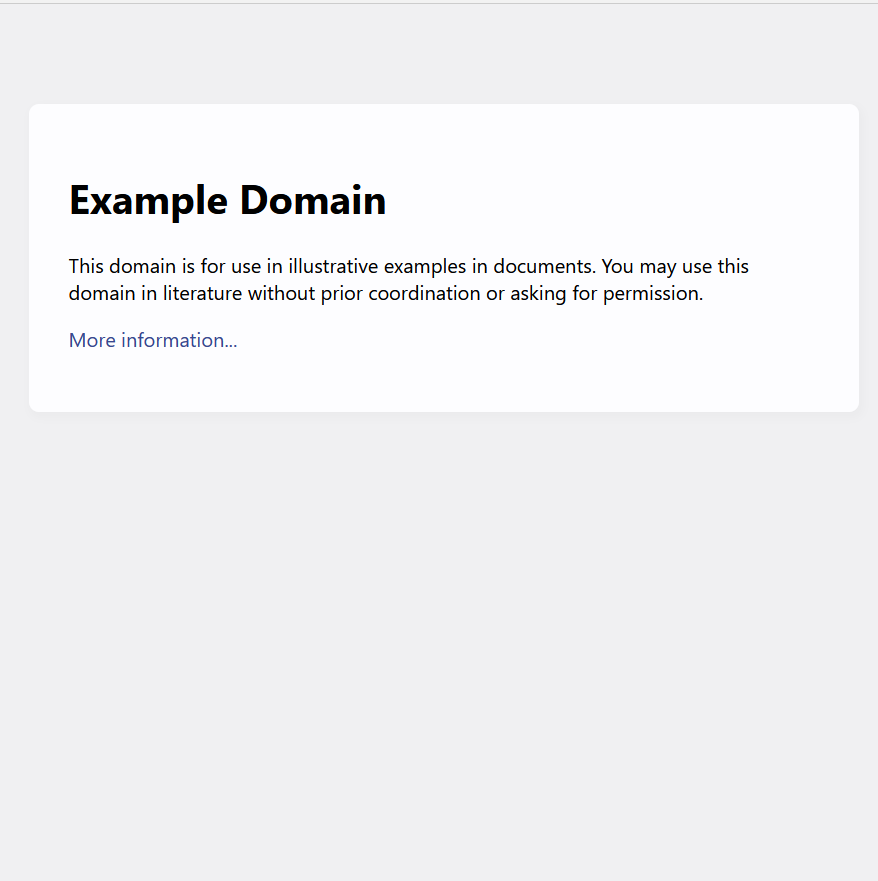
* Debe **adaptarse** a las **distintas arquitecturas físicas** posibles
* Solo deben implicar **cambios en la configuración** pero no en el desarrollo
  1. **Utilización de componentes en los servicios de infraestructura**
* Los componentes de la infraestructura deben ser independientes del dominio
  1. **Gestion de la sesión de usuario**
* Se encarga de las **entidades en sesión de usuario** y **contexto de la aplicación**

**5.5 Aplicación de patrones de diseño**

* Definicion del patrón de diseño

1. **TIPOS DE APLICACIONES WEB**

* **Pagina web estatica:** 
  + Implementada en **HTML**
  + Pueden mostrar en **alguna parte de la pagina** **objetos en movimiento** como videos, fotos, gifs etc..

**Ejemplo:**

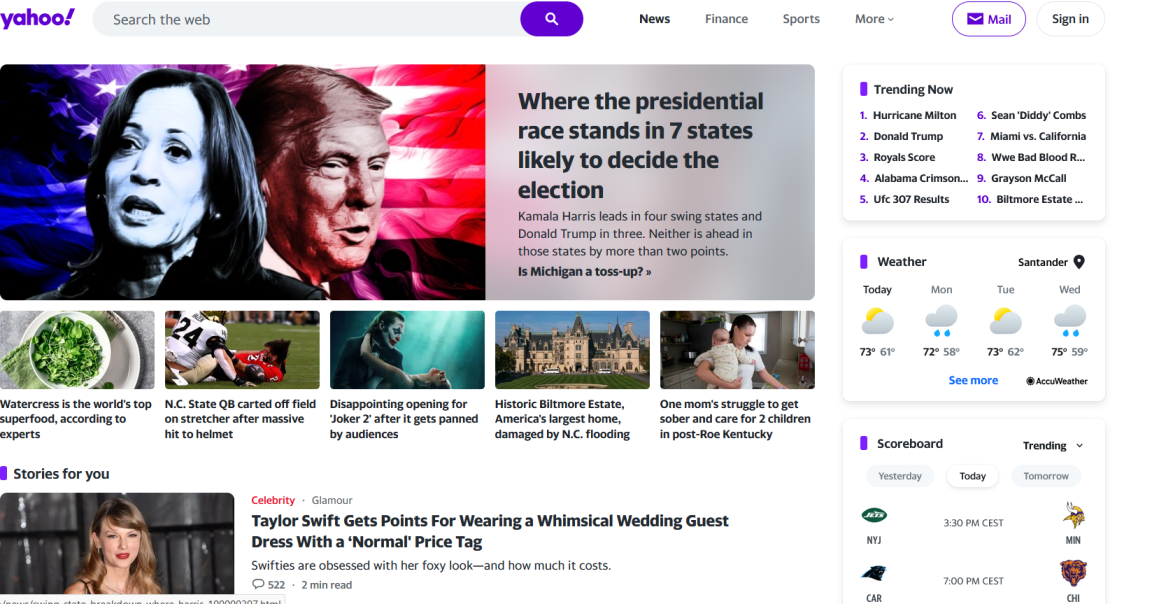
* **Pagina web dinámica**
  + Conjunto de páginas cuyo **contenido cambia según la ubicaciones, acciones pasadas, zonas horarias etc…**

**Ejemplo:** **wikipedia**



* **Portal**:
  + Sitio web que **permite el acceso** a **multiples secciones** como fotros, chats, cuentas de correo, buscador, noticias etc….

**Ejemplo: Yahoo**



* **Tienda virtual o comercio electrónico**
  + Sitio web que **publica productos** y **permite la compra online** a trabes de tarjetas de crédito
  + Ofrece al administrador un panel para subir los productos

**Ejemplo: amazon**

* **Pagina web con gestor de contenidos**
  + Sitio web que **permite a los usuarios** **crear**, **editar**, gestionar y **publicar** **contenido en línea**

**Ejemplo: YouTube**

1. **REQUERIMENTOS DE UN SERVIDOR WEB**

Si pensamos **montar un servidor web en una red**, **debemos tener en cuenta:**

* **Prestaciones de Hardware del servidor**
* **Sistema operativo**
* **Servidor web**
* **Configuración de la red**
  + **IP** para el **servidor web**
  + **IP** para las **pruebas** desde una maquina cliente
  + **Servidor DNS**

Si vamos a **plantearnos un proyecto de implantación** **y configuración de un servidor web**, deberemos tener en cuenta

* **Recursos y conectividad del equipo** **servidor**
* **Servidor web empleado**
* **Posibilidades** del **servidor web empleado**
* **Requisitos de las aplicaciones web**
* Entregables y fechas

**La configuración de un servidor web** dependerá:

* **Del tipo de contenido**, si son **estaticas** o **dinamicas**, si hay transacion de informacion….
* Si el servidor **puede albergar varias paginas o sitios**, si permite **host virtuales** y **como crece en el futuro**
* La **cantidad de paginas web que debe servir** , **escalabilidad** y **estabilidad**
* Realiza **pruebas de funcionamiento**

1. **SERVIDOR DE APLICACIONES WEB**

* Es un **servidor que proporciona o ejecuta aplicaciones en una red**
* Centralizacion y disminución de la complejidad en el desarrollo de aplicaciones web
* **Tipos de servidores de aplicaciones**
  + Servidores de **Java EE**
  + Servidores **.NET** de Microsoft
  + Servidores **Apache+PHP+MySql**

1. **PLATAFORMAS WEB LIBRES**

Una plataforma web consta de **4 componentes básicos:**

* El **sistema** **operativo**
* El **servidor** **web**
* El **gestor** **de** **bases** **de** **datos**
* Un **lenguaje** **de** **programación**

Difrentes combinaciones de los 4, dan lugar a plataformas web como **LAMP y WISA**

* **LAMP:** 
  + **Linux**: sistema operativo
  + **Apache**: servidor web
  + **Mysql**: Gestor de base de datos
  + **PHP**: Lenguaje de programación
* **WISA**
  + **Windows**: sistema operativo
  + **Internet Information Services**: sevidor web
  + **SQL Server**: gestor de base de datos
  + **ASP** o ASP.net: lenguaje para scripting del lado del servidor

1. **SERVIDOR APACHE-HOST VIRTUALES**

* Si queremos tener **distintas IP o distintos nombres en nuestro servidor** Apache, podemos hacerlo recurriendo a **sitios web virtuales**
* Generalmente los sitios web virtuales **pueden estar basados en IP o en nombre**
  + **Basados en IP:** 
    - **Cada sitio web tiene una dirección IP diferente**
    - La **IP** que debemos poner en la **definición de la directiva virtual host cambia**
    - **Cada IP** corresponde a una **interfaz de red** del servidor web
  + **Basados en nombres:**
    - **Con una sola dirección IP están funcionando sitios web con diferentes nombres de dominio**
    - **Asociamos a la IP del servidor los nombres de dominio que necesitamos**

**Archivo de configuración**

* A la hora de configurar los posibles host virtuales, se deberá modificar el **archivo httpd-vhosts.conf de Apache**.
* Dicho fichero contara con diferentes instrucciones como:



* + **IP\_servidor:80**
    - Es la **IP de nuestro servidor** con el **puerto para http**
    - Cuando el servidor tiene interfaz única y lo **usamos solo para nuestras fases de desarrollo** **podemos sustituir el valor por \*80**
    - En el caso de tener **varias direcciones IP**, entonces en la directiva **VirtualHost**, **sustituiríamos el \* por las IP para cada sitio.**
  + **DocumentRoot**
    - La **ruta** con la carpeta **donde tenemos el proyecto**
  + **ServerName**
    - **Nombre de dominio** escogido
  + **ServerAlias**
    - Todos los alias que aceptamos en el navegador para el mismo dominio
  1. **Servidores de dominio basados en nombre**
* Es la **forma más simple** de configurar Hosts Virtuales.
* Necesitamos **configurar** **nuestro servidor DNS** para asignar la **IP** al nombre y **configurar apache**
* Debemos tener claro que **tendremos designada una sola IP para atender todas las peticiones de los sitios, por eso usaremos el valor \* para <VirtualHost>**