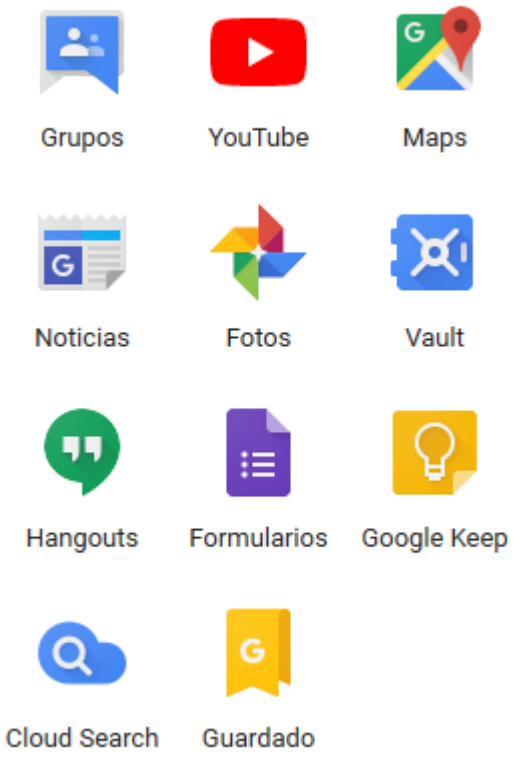
UT 1: Arquitecturas y tecnologías de desarrollo Web

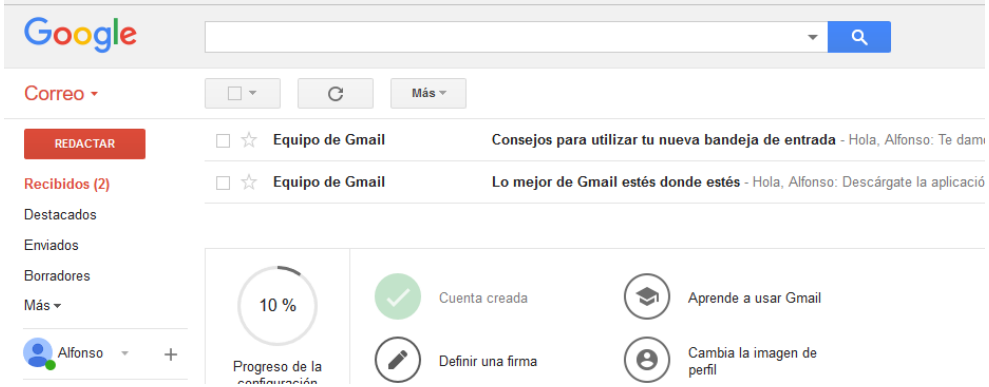
2ºDAW – Desarrollo Entorno Servidor

Alfonso Rebolleda Sánchez

Aplicaciones Web

Aplicación distribuida cuya interfaz de usuario es accesible desde un navegador web.





Arquitectura Web

WWW (World Wide Web)

- Servicio de distribución de información.
- Acceso a millones de recursos electrónicos y aplicaciones distribuidos en servidores por Internet.
- Identificados y localizados por direcciones
  URL (Uniform Resource Locator): http://host[:puerto]/ruta-y-nombre
- Conectados entre sí a través de hiperenlaces (o hipervínculos)
- Desarrollada por el CERN en 1989.
- W3C (World Wide Web Consortium)
  http://www.w3.org/
  http://www.w3c.es/
- Desarrolla estándares web: XHTML, CSS y XML.
- **Página Web vs Sitio Web vs Aplicación Web**

Arquitectura Cliente/Servidor

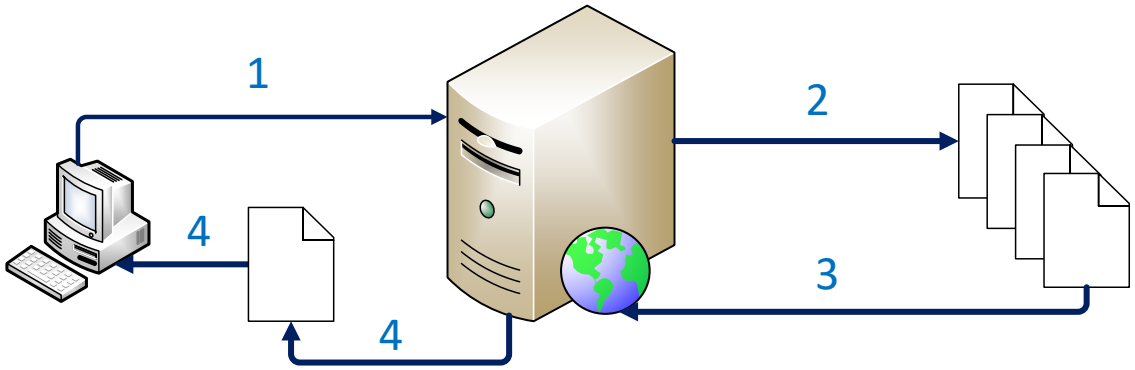
El funcionamiento de la Web es posible gracias a la coexistencia de una serie de componentes hardware y software:

- Hubs, repetidores, puentes, pasarelas, enrutadores, etc.
- Protocolos de comunicaciones (TCP, IP, HTTP, FTP, SMTP …)
- Sistema de nombres de dominio (DNS) para la búsqueda y utilización de recursos => URL del recurso solicitado.
- Software específico para consumir los recursos

Arquitectura Cliente/Servidor

Peticiones HTTP

1. Desde el cliente se solicita a un servidor web un recurso (página web)
2. El servidor busca esa recurso en una ubicación
3. Si el servidor encuentra esa página, la recupera.
4. Y por último se la envía al cliente para que éste pueda mostrar su contenido.

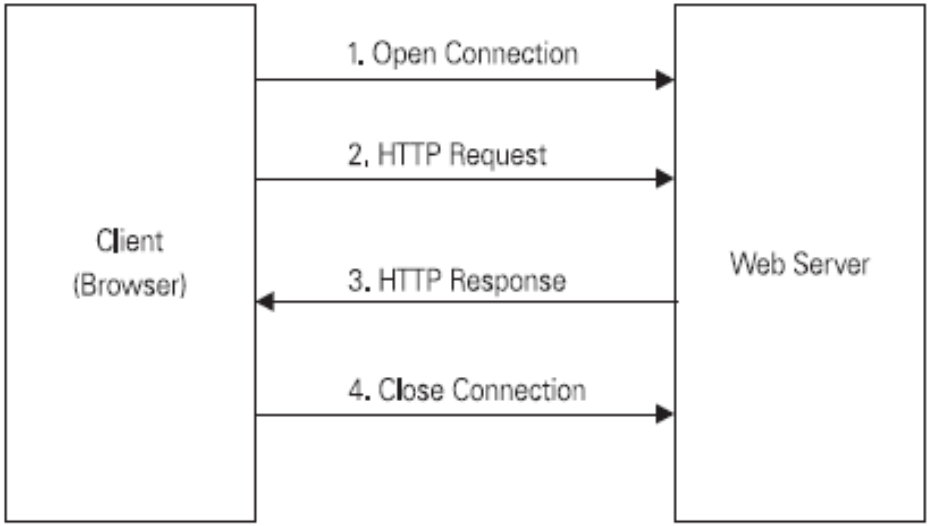


Arquitectura Cliente/Servidor

Protocolo HTTP

- Navegador necesita interactuar con un servidor web, realiza una **request** (una solicitud o petición) hacia el servidor web, éste procesa la request y emite una **response** hacia el navegador.

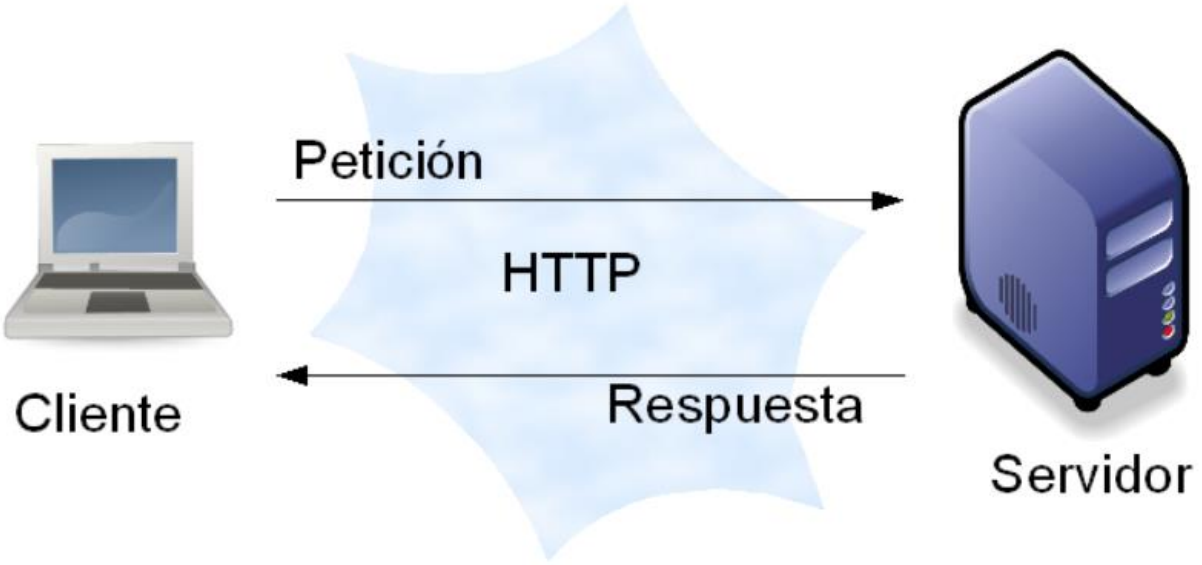
- HTTP es un **protocolo sin estado**, lo que significa que una vez realizada la request, y recibida la correspondiente response en el navegador, la conexión entre el navegador y el servidor web deja de existir (el navegador y el servidor web no permanecen conectados)



Arquitectura Cliente/Servidor

La **arquitectura cliente/servidor** está basada en la idea de **servicio**.

- El cliente es un componente consumidor de servicios.
- El servidor es un proceso proveedor de dichos servicios.
- La comunicación entre ambos componentes se lleva a cabo a través del intercambio de mensajes.
- Normalmente el cliente, a través de un navegador, inicia el intercambio de información, solicitando datos al servidor .
- El servidor responde enviando uno o más flujos de datos al cliente.



Arquitectura Cliente/Servidor

Servidores Web

Su cometido básico es proveer de contenido estático a un cliente que ha realizado una petición a través de un navegador.

Las peticiones al servidor contienen una dirección de tipo URL formada por:

- Referencia a un cierto protocolo (HTTP, FTP, etc.)
- Dirección IP o nombre de dominio del servidor
- Descripción del recurso en forma de ruta al objeto que queremos acceder.
- Opcionalmente se puede incluir el puerto por el que el servidor escucha las peticiones del cliente.

Arquitectura Cliente/Servidor

Servidores Web

Programa que contesta y genera la respuesta HTTP a las peticiones de recursos web por parte del cliente.

Funcionalidad básica:

- Se conecta con el cliente.
- Recibe el mensaje HTTP de la petición (GET, HEAD, POST)
- Procesa el mensaje HTTP .
- Localiza y envía el resultado (en forma de mensaje HTTP)

Los servidores de altas prestaciones, además:

- Tratan múltiples peticiones: hilos para manejar cada conexión.
- Generan dinámicamente contenido: ASP, PHP, JSP

Servidores Web: Apache HTTP server, IIS, Nginx, Lighttpd, …

Arquitectura Cliente/Servidor

Clientes Web

- Originan el tráfico web: Envían las peticiones y reciben las respuestas.
- Dos clases de clientes web: navegadores y robots

  Navegadores (Iexplorer, Chrome, Opera, FireFox, Safari …).Utilizan caches de memoria y disco.

  Robots (Motores de búsqueda): las peticiones son automatizadas.
- Construyen y envían la petición HTTP .
- Reciben, interpretan y presentan la respuesta.
- El protocolo por defecto es http
- Caché local: sirve recursos guardados en la caché sin conectarse al
- servidor
- Gestión de Cookies.

Web estáticas vs Web dinámicas

Páginas web estáticas: almacenadas en su forma definitiva, tal y como se crearon, y su contenido no varía. Son útiles para mostrar una información concreta, y mostrarán esa misma información cada vez que se carguen.

Páginas web dinámicas: contenido cambia en función de diversas variables, como puede ser el navegador que estás usando, el usuario con el que te has identificado, o las acciones que has efectuado con anterioridad.

Dos tipos web dinámicas:

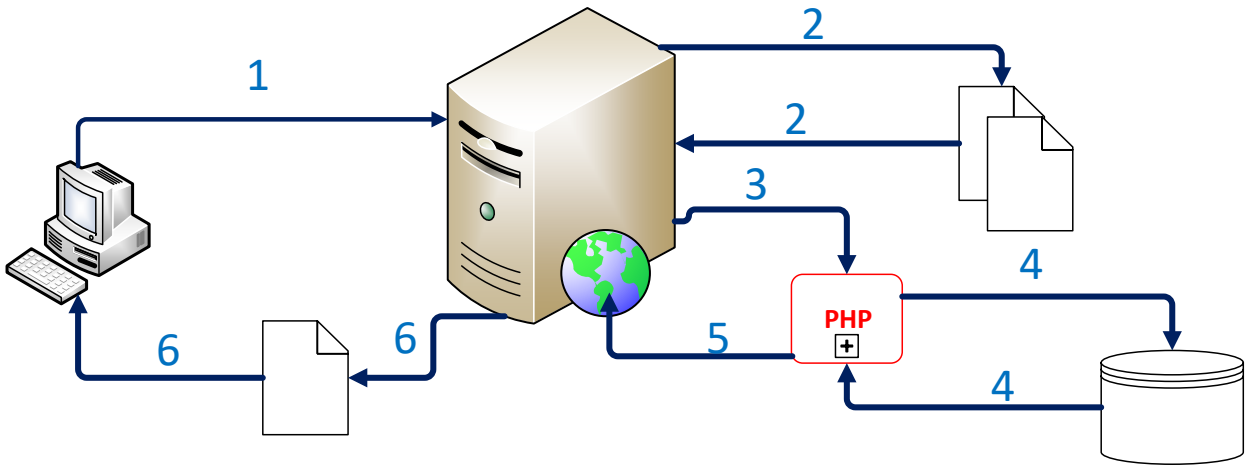
- Aquellas que incluyen código que ejecuta el navegador. El código ejecutable (normalmente JavaScript) se incluye dentro del HTML y se descarga junto con la página. Cuando el navegador muestra la página en pantalla, ejecuta el código embebido.

- Aquellas en que HTML se forma como resultado de la ejecución de un programa (ejecución tiene lugar en el servidor web). Páginas con extensiones .php, .asp, .jsp, .cgi o .aspx. Contenido que se descarga al navegador es similar al de una página web estática.

Web estáticas vs Web dinámicas

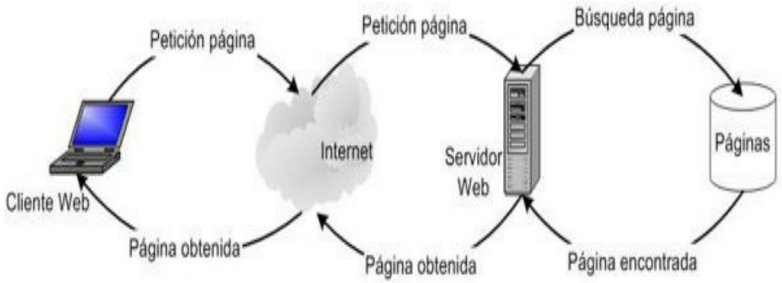
Funcionamiento:

1. Cliente web solicita a un servidor web una página web.
2. El servidor busca esa página y la recupera.
3. En el caso de que se trate de una página web dinámica (contenido debe ejecutarse para generar HTML se enviará al cliente), el servidor web contacta con el módulo responsable de ejecutar el código y se lo envía.
4. Como parte del proceso de ejecución, puede ser necesario obtener información de algún repositorio (normalmente una base de datos).
5. El resultado de la ejecución será una página en formato HTML, similar a cualquier otra página web no dinámica.
6. El servidor web envía el resultado obtenido al cliente web, que la procesa y muestra en pantalla.

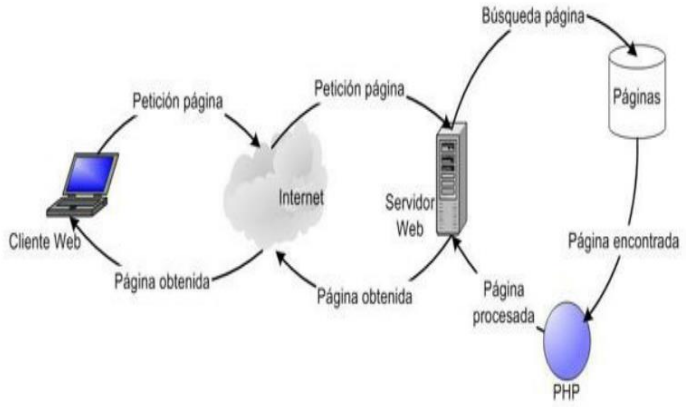


Web estáticas vs Web dinámicas

Web estática



Web dinámica



Web estáticas vs Web dinámicas

Ventajas web estáticas:

- No es necesario saber programar para crear un sitio que utilice únicamente páginas web estáticas. Simplemente habría que conocer HTML/XHTML y CSS, e incluso esto no sería indispensable se podría utilizar algún programa de diseño web para generarlas.

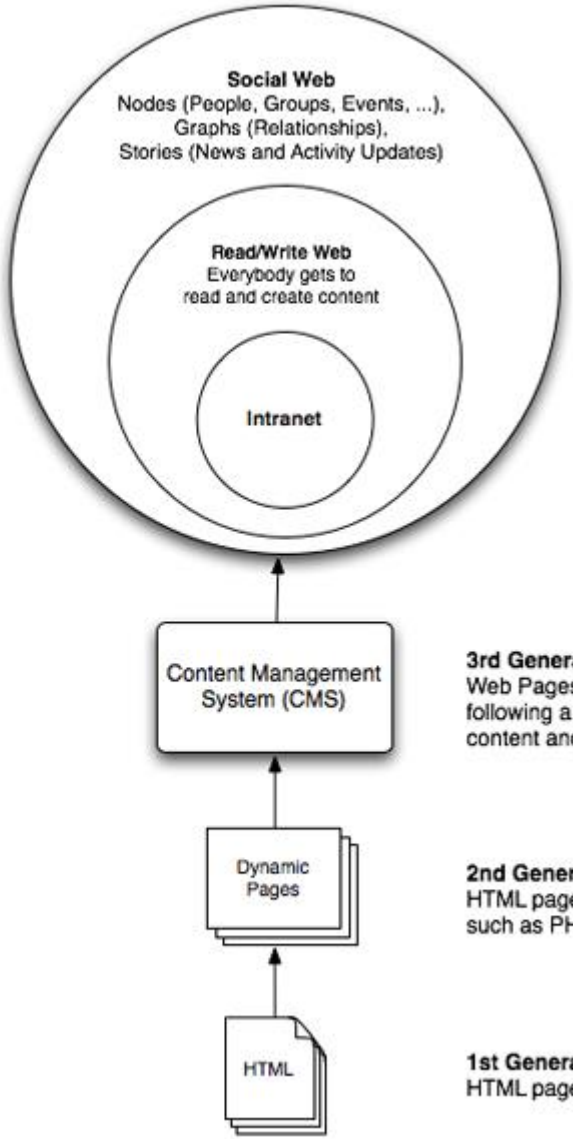
- Sólo necesitan un servidor web que se comunique con navegador, no necesita módulos para procesar una web dinámica.

Desventajas web estáticas:

- Actualización de contenido debe hacerse de forma manual editando la página que almacena el servidor web.

- Limitaciones para diseño aplicaciones

Web estáticas vs Web dinámicas



4th Generation
Content can be public or available to registered members only.

3rd Generation
Web Pages generated on the fly using a Software following a business logic. A Few people create content and most people only read them.

2nd Generation
HTML pages generated by scripting languages such as PHP, Perl, etc.

1st Generation
HTML pages build by web designers one by one

Aplicaciones Web

Las aplicaciones web emplean páginas web dinámicas para crear aplicaciones que se ejecuten en un navegador.

Existen aplicaciones web para multitud de tareas.

Estas aplicaciones tienen ciertas ventajas e inconvenientes si las comparas con las aplicaciones tradicionales que se ejecutan sobre el sistema operativo de la propia máquina.

Características:

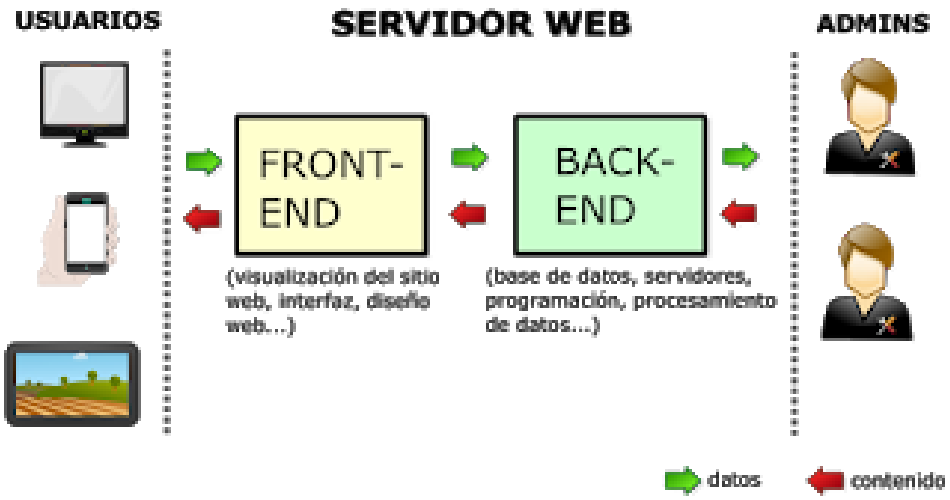
- No es necesario instalarlas en aquellos equipos en que se vayan a utilizar
- Se pueden utilizar desde cualquier dispositivo que dispongan de navegador web.
- Se pueden utilizar desde cualquier lugar en el que dispongamos de conexión con el servidor.

Aplicaciones Web

Muchas aplicaciones Web se basan en la generación de páginas dinámicas.

Esquema general:

- Una parte externa (**front-end**) conjunto de páginas que ven la gran mayoría de usuarios que las usan (usuarios externos).
- Una parte interna (**back-end)** conjunto de páginas dinámicas que utilizan las personas que producen el contenido y las que administran la aplicación web (usuarios internos) para crear contenido, organizarlo, decidir la apariencia externa, etc.



Ejecución Código Servidor vs Cliente

Navegador solicita a un servidor web una página, es posible que antes de enviarla haya tenido que ejecutar algún programa para obtenerla. **El código se ejecuta en el entorno del servidor web.**

Cuando una página web llega al navegador, es también posible que incluya algún programa o fragmentos de código que se deban ejecutar. **El código se ejecuta en el entorno del cliente web.**

Técnica de desarrollo web conocida como AJAX, que nos posibilita realizar programas en los que el código JavaScript que se ejecuta en el navegador pueda comunicarse con un servidor de Internet para obtener información con la que, por ejemplo, modificar la página web actual.

Tecnologías Web de Servidor

Componentes para diseño aplicaciones Web:

- Un **servidor web** para recibir las peticiones de los clientes web y enviarles la página que solicitan (una vez generada puesto que hablamos de páginas web dinámicas). El servidor web debe conocer el procedimiento a seguir para generar la página web: qué módulo se encargará de la ejecución del código y cómo se debe comunicar con él.

- El **módulo encargado de ejecutar el código** o programa y generar la página web resultante. Este módulo debe integrarse de alguna forma con el servidor web, y dependerá del lenguaje y tecnología que utilicemos para programar la aplicación web.

- Un **gestor de base de datos**, que almacenará la información a mostrar.

- El **lenguaje de programación** que utilizarás para desarrollar las aplicaciones.

Tecnologías Web de Servidor

Además de los componentes es importante decidir cómo se a organizar el código de la aplicación.

Muchas de las arquitecturas organizan código de las aplicaciones en capas o niveles. El motivo de dividir en capas el diseño de una aplicación es que se puedan separar las funciones lógicas de la misma.

- **Capa de presentación**: se encarga de dar formato a los datos para presentárselo al usuario final
- **Capa Lógica**: utiliza los datos para ejecutar un proceso y obtener un resultado.
- **Capa Acceso a Datos**: acceso a datos almacenados en SGBD

Arquitecturas

Java EE (Java Enterprise Edition) antes conocida como J2EE. Es una plataforma orientada a la programación de aplicaciones en lenguaje Java. Puede funcionar con distintos gestores de bases de datos, e incluye varias librerías y especificaciones para el desarrollo de aplicaciones de forma modular.

Dentro de esta arquitectura existen distintas tecnologías como las páginas JSP y los servlets, ambos orientados a la generación dinámica de páginas web, o los EJB, componentes que normalmente aportan la lógica de la aplicación web.

Arquitecturas

AMP siglas de Apache, MySQL y PHP/Perl/Python. Las dos primeras siglas hacen referencia al servidor web (Apache) y al servidor de base de datos (MySQL). La última se corresponde con el lenguaje de programación utilizado, que puede ser PHP, Perl o Python.

Dependiendo del sistema operativo que se utilice para el servidor, se utilizan las siglas LAMP (para Linux), WAMP (para Windows) o MAMP (para Mac). También es posible usar otros componentes, como el gestor de bases de datos PostgreSQL en lugar de MySQL.

Todos los componentes de esta arquitectura son de código libre (open source).

Arquitecturas

CGI/Perl combinación de dos componentes: Perl (potente lenguaje de código libre creado originalmente para la administración de servidores) y CGI, un estándar para permitir al servidor web ejecutar programas genéricos, escritos en cualquier lenguaje (también se pueden utilizar por ejemplo C o Python), que devuelven páginas web (HTML) como resultado de su ejecución.

El principal inconveniente de esta combinación es que CGI es lento

Perl

Arquitecturas

ASP.Net es la arquitectura Microsoft para el desarrollo de aplicaciones Web. Es la parte de la plataforma .Net destinada a la generación de páginas web dinámicas.

El lenguaje de programación puede ser Visual Basic.Net o C#. La arquitectura utiliza el servidor web de Microsoft, IIS, y puede obtener información de varios gestores de bases de datos.

Tiene su propio entorno de desarrollo, Visual Studio.

Lenguajes

Se diferencian en la forma en que se ejecutan en Servidor Web.

- **Lenguajes de guiones (scripting):** programas se ejecutan directamente a partir de su código fuente original.

  Se almacenan normalmente en un fichero de texto plano. Cuando el servidor web necesita ejecutar código programado en un lenguaje de guiones, le pasa la petición a un intérprete, que procesa las líneas del programa y genera como resultado una página web.

  Perl, Python, PHP y ASP (el precursor de ASP.Net).

Lenguajes

Se diferencian en la forma en que se ejecutan en Servidor Web.

- **Lenguajes compilados a código nativo** (Código que puede ser ejecutado directamente por el procesador): código fuente se traduce a código binario antes de ser ejecutado. El servidor web almacena los programas en su modo binario, que ejecuta directamente cuando se les invoca.

  El método principal para ejecutar programas binarios desde un servidor web es CGI. Utilizando CGI podemos hacer que el servidor web ejecute código programado en cualquier lenguaje de propósito general como puede ser C.

Lenguajes

Se diferencian en la forma en que se ejecutn en Servidor Web.

- **Lenguajes compilados a código intermedio**: lenguajes en que código fuente original se traduce a un código intermedio, independiente del procesador, antes de ser ejecutado.

  Es la forma en la que se ejecutan por ejemplo las aplicaciones programadas en Java, y lo que hace que puedan ejecutarse en varias plataformas distintas.

  Java EE (servlets y páginas JSP) y ASP.Net

Lenguajes

Ventajas e inconvenientes:

- Los lenguajes de guiones tienen la ventaja de que no es necesario traducir el código fuente original para ser ejecutados, lo que aumenta su portabilidad. Si se necesita realizar alguna modificación a un programa, se puede hacer en el momento. Por el contrario el proceso de interpretación ofrece un peor rendimiento que las otras alternativas.

- Los lenguajes compilados a código nativo son los de mayor velocidad de ejecución, pero tienen problemas en lo relativo a su integración con el servidor web. Son programas de propósito general que no están pensados para ejecutarse en el entorno de un servidor web.

- Los lenguajes compilados a código intermedio ofrecen un equilibrio entre las dos opciones anteriores. Su rendimiento es muy bueno y pueden portarse entre distintas plataformas en las que exista una implementación de la arquitectura (como un contenedor de servlets o un servidor de aplicaciones Java EE).

Integración con Lenguajes Marcas

Forma de realizar páginas web dinámicas: **integrar las etiquetas HTML en el código de los programas.**

```
<?php
echo "<html>";
echo "<head></head>";
echo "<body>";
echo "<H1>";
echo "Hola, hoy es ";
echo date('l');
echo "</H1>";
echo "</body>";
echo "</html>";
?>
```

Integración con Lenguajes Marcas

Otro enfoque consiste en **integrar el código del programa en medio de las etiquetas HTML** de la página web. De esta forma, el contenido que no varía de la página se puede introducir directamente en HTML, y el lenguaje de programación se utilizará para todo aquello que pueda variar de forma dinámica.

Esta metodología de programación es la que se emplea en los lenguajes ASP, **PHP** y con las páginas JSP de Java EE.

```
 <head></head>
<body>
 <H1>
   Hola, hoy es <?php echo date('l'); ?>.
 <\H1>
</body>
</html>
```

Programación Web con PHP

PHP es un lenguaje de guiones de propósito general, diseñado para el desarrollo de páginas web dinámicas utilizando código embebido dentro del lenguaje de marcas.

Su sintaxis está basada en la de C / C++, y por lo tanto es muy similar a la de Java.

El código se ejecuta en entorno de ejecución con el que se integra el servidor web (normalmente utilizando Apache con el módulo **mod_php**). La configuración tanto del servidor web Apache, como de PHP, se realiza por medio de ficheros de configuración.