



# Desarrollo web entorno servidor

---



**Ciclo:** DAW  
**Módulo:** DWES  
**Curso:** 2020-2021  
**Autor:** César Guijarro Rosaleny




<b>Introducción.....</b>	<b>3</b>
<b>Páginas web estáticas y dinámicas.....</b>	<b>5</b>
Páginas web dinámicas.....	6
Componentes servidor web.....	7
<b>Servidores HTTP.....</b>	<b>8</b>
<b>Lenguajes de programación.....</b>	<b>10</b>
<b>Sistemas Gestores de Base de Datos.....</b>	<b>14</b>
<b>Bibliografía.....</b>	<b>17</b>

## Introducción

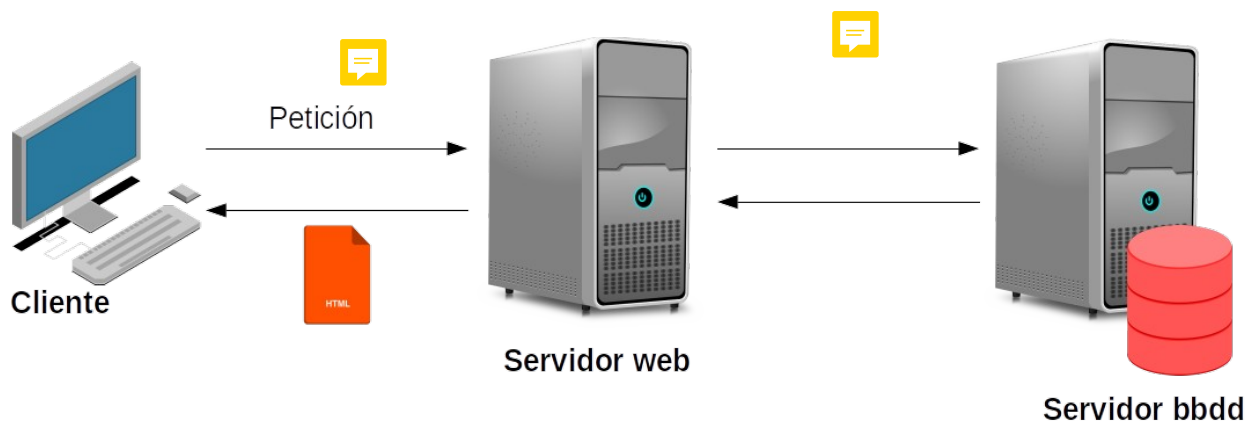


En toda conexión web existen dos partes bien separadas: cliente y servidor.

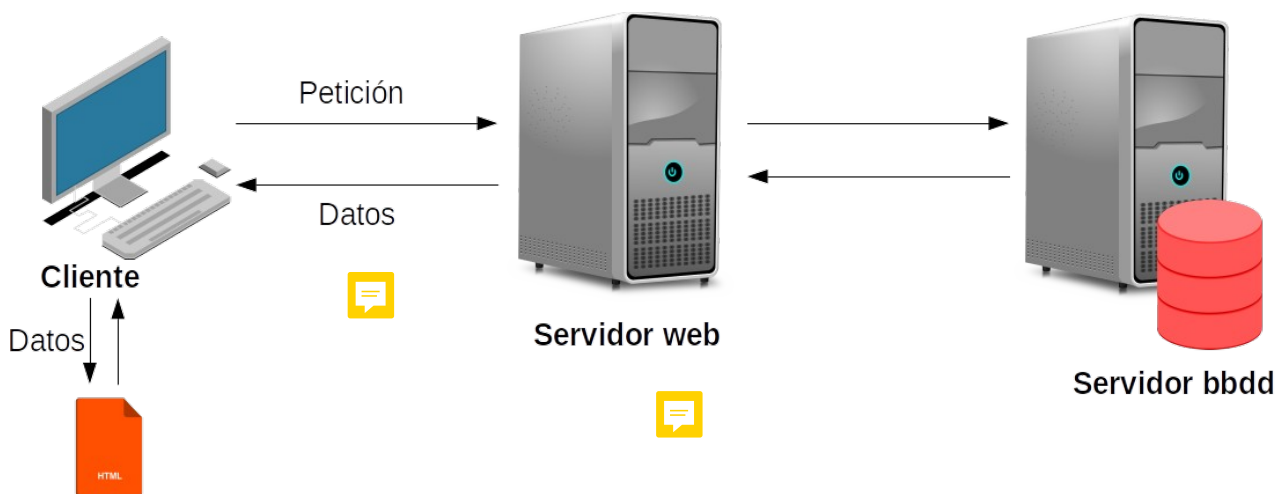
Cuando el cliente (el navegador web, por ejemplo) accede a una URL el servidor le devuelve el archivo que tiene configurado como principal (index.html, index.php, server.html...). A partir de ahí, el cliente va navegando por el sitio web accediendo a los diferentes archivos (usuarios.html, clientes.html...). 

Dependiendo de si el contenido del archivo es fijo o se genera en cada petición, hablamos de páginas web estáticas o dinámicas.

Con este enfoque, el servidor hace todo el trabajo, mientras que el cliente se dedica a hacer peticiones y mostrar la respuesta. El servidor se dedica a recibir las peticiones de los clientes, las procesa, accede a la base de datos, monta las vistas (los diferentes archivos html) y las envía al cliente (si el contenido es estático, el proceso se simplifica, ya que no tiene que acceder a ninguna bbdd ni montar las vistas).



Hoy en día existen multitud de dispositivos que pueden realizar peticiones a servidores web, y muchos de ellos no pueden interpretar los archivos html, de ahí la importancia de los servicios web (como las API Rest), donde el servidor se dedica únicamente a recibir las peticiones de los clientes, acceder a la bbdd y devolver los datos al cliente (en formato XML, JSON...). En este caso, es el cliente el encargado de montar las diferentes vistas con los datos recibidos del servidor.



De esta forma, el servidor no hace todo el trabajo, dejando al cliente la parte de montar las vistas (o gestionando los datos como quiera) aligerando la carga de trabajo.

## Páginas web estáticas y dinámicas

---

Las páginas webs estáticas fueron las primeras en usarse. El servidor muestra siempre la misma información en todo momento, dependiendo sólo del archivo *html* solicitado.

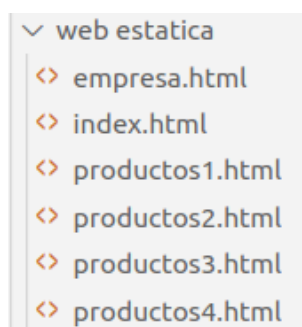
Supongamos que creamos una web de una tienda online. Podríamos crear varios archivos *html*: *index.html*, donde estaría nuestra página de inicio, *productos.html*, con la información de los productos que vendemos y *empresa.html*, con los datos de nuestra empresa.



```
▼ web estatica
  <> empresa.html
  <> index.html
  <> productos.html
```

Estos archivos mostrarían siempre la misma información, independientemente de qué usuario accediera a nuestra web, que navegador usara, el número de artículos que tuviésemos a la venta...

Ahora supongamos que queremos mostrar un listado de productos en *productos.html*, y además queremos paginar la información de forma que se muestren 20 productos por página. Si nuestra tienda tuviera 70 artículos, deberíamos crear varias páginas de productos para mostrarlos de 20 en 20.



```
▼ web estatica
  <> empresa.html
  <> index.html
  <> productos1.html
  <> productos2.html
  <> productos3.html
  <> productos4.html
```

Si tenemos suerte y nuestra tienda online crece, deberíamos crear cientos, miles o incluso millones (pensad en una web como Amazon con millones de artículos a la venta) de archivos *productos#.html*, cosa del todo inviable.

Además, si quisiésemos cambiar la información que mostramos de cada artículo, tendríamos que editar todos esos archivos. O si queremos mostrar ofertas personalizadas a cada cliente, el número de archivos crecería, para mostrar los productos en general y las ofertas de cada cliente.

Con las páginas webs dinámicas resolvemos todos estos problemas. Con este enfoque, creamos sólo una página *productos.html*, y mediante algún lenguaje de programación (PHP, Java, Python, Node...) accedemos a una base de datos donde tenemos almacenados los productos y montamos las diferentes páginas de forma dinámica.

## Páginas web dinámicas

Las páginas web dinámicas generan los archivos *html* de forma dinámica, es decir, dependiendo de alguna opción (como el número de página en nuestro ejemplo de los productos) muestran una información u otra.

Por ejemplo, si quisiésemos mostrar la información del cliente en nuestra tienda online, según el id de ese cliente mostraría sus datos, sin necesidad de crear varias páginas para cada cliente.

Para poder generar esas páginas de forma dinámica necesitamos instalar en nuestro servidor HTTP un interprete del lenguaje de programación utilizado y, generalmente, un sistema gestor de base de datos.




¿Significa éso que siempre es mejor usar páginas webs dinámicas? No, depende del tipo de web que estemos creando. Las webs dinámicas tienen sus ventajas y desventajas, y según la información que queremos mostrar usaremos éstas o webs estáticas.

Si nuestro sitio web se limita a mostrar información sin necesidad de mostrar contenido dinámico es mejor usar archivos *html* estáticos, ya que el servidor no tendrá que procesar los archivos del lenguaje utilizado para convertirlos en *html* y servirá las páginas más rápido.

En cualquier caso, hoy en día casi todas las webs tienen algún tipo de contenido dinámico (información almacenada en una bbdd, páginas con información de usuarios...), con lo que si queremos trabajar como programadores web, tendremos que aprender como crear este tipo de contenido.

## Componentes servidor web

Para crear un sitio web dinámico y que sea accesible, necesitamos una serie de componentes (tanto en nuestro hosting como en local para desarrollarlo):

- **Servidor HTTP**  que será el encargado de gestionar las peticiones y enviar la respuesta (archivos *html*, datos JSON...)
- **Intérprete del lenguaje utilizado**, para poder interpretar el lenguaje de programación y crear los archivos *html* o datos. 
- **Sistema Gestor de Base de Datos**, para poder almacenar nuestros datos. 

## Servidores HTTP

---

Un **servidor HTTP** o **servidor web** es un programa informático que procesa las peticiones de los clientes y genera una respuesta en cualquier lenguaje o aplicación del lado del cliente. El código recibido por el cliente es renderizado por un navegador web. Generalmente se utiliza el protocolo HTTP para estas comunicaciones.

Actualmente existen muchos servidores HTTP. A continuación veremos algunos de los más usados.



El servidor web **Apache** es un servidor HTTP de código abierto para plataformas UNIX (BSD, GNU/Linux...), Microsoft Windows, Macintosh y otras.

El servidor apache es desarrollado y mantenido por una comunidad de usuarios bajo la supervisión de la [Apache Software Foundation](https://www.apache.org/), dentro del proyecto HTTP Server (httpd).

Desde 1996, Apache es el servidor HTTP más utilizado. Alcanzó su máxima cuota de mercado en 2005, siendo el servidor empleado en el 70% de los sitios web en el mundo. Sin embargo, ha sufrido un descenso en su cuota de mercado en los últimos años. En 2009 se convirtió en el primer servidor web que alojó más de 100 millones de sitios web.





**Nginx** es un servidor web/proxy inverso ligero de alto rendimiento, y un proxy para protocolos de correo electrónico (IMAP/POP3).

Es software libre y de código abierto, licenciado bajo la licencia BSD simplificada. También existe una versión comercial distribuida bajo el nombre de Nginx plus. Es multiplataforma, por lo que corre en sistema UNIX, Windows y Macintosh.

El sistema es usado por una larga lista de sitios web conocidos, como WordPress, Netflix, GitHub y partes de Facebook.



**Internet Information Services (IIS)** es un servidor web y un conjunto de servicios para el sistema operativo Microsoft Windows.

Este servicio convierte a un PC en un servidor web para Internet o una intranet, es decir que en los ordenadores que tienes este servicio instalado se pueden publicar página web tanto local como remotamente.

## Lenguajes de programación

---

Para poder servir webs dinámicas, necesitamos algún lenguaje de programación para poder generar la información (archivos *html*, datos...). Para la parte del cliente (**frontend**), los más usados son HTML, CSS y Javascript. En la parte del servidor (**backend**), tenemos varias opciones, como PHP, ASP.NET, JSP, Python y Node.js, Go, Ruby...



**PHP** (acrónimo de *Hypertext Preprocessor*) es un lenguaje de código abierto especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

El código php es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente que era.

Aunque php es muy fácil de aprender para el principiante, ofrece muchas características avanzadas para los programadores profesionales.

Actualmente, es el lenguaje más usado del lado servidor, entre otras cosas por ser el utilizado por Wordpress para crear sus plugins.



**ASP.NET** es un modelo de desarrollo web unificado, desarrollado y comercializado por Microsoft, que incluye los servicios necesarios para crear aplicaciones web empresariales con el código mínimo. Es el sucesor de la tecnología ASP.

El código de las aplicaciones puede escribirse en cualquier lenguaje compatible con el CLR (*Common Language Runtime*), entre ellos Microsoft Visual Basic, C#, Jscript .NET y N#.



**JSP** (*JavaServer Pages*) es una tecnología para crear páginas web dinámicas basadas en HTML y XML, entre otros tipos de documentos. Es similar a php, pero usa el lenguaje de programación Java.

La principal ventaja de JSP frente a otros lenguajes es que el lenguaje Java es un lenguaje de propósito general que excede el mundo web y que es apto para crear clases que manejan lógica de negocio y acceso a datos de una manera prolija.



**Python** es un lenguaje de programación interpretado, funcional, orientado a objetos e interactivo.

Python cuenta con una sintáxis muy limpia y clara. Cuenta con módulos, clases, excepciones, tipos de datos de muy alto nivel, así como tipado dinámico.



**Node.js** es un entorno de ejecución para JavaScript del lado del servidor construido con el motor *JavaScript V8*, desarrollado por Google para su navegador Chrome. Node.js usa un modelo de operaciones E/S sin bloqueo y orientado a eventos, que lo hace liviano y eficiente.

Al contrario que la mayoría del código JavaScript, **no se ejecuta en un navegador, sino en el servidor.**



**Go** es un lenguaje de programación desarrollado por Google concurrente y compilado inspirado en la sintaxis de C, que intenta ser dinámico como Python y con el rendimiento de C o C++. Es un proyecto opensource.

Go es un lenguaje de programación compilado y admite el paradigma orientado a objetos, pero a diferencia de los lenguajes de programación más populares no dispone de herencia de tipos y tampoco de palabras clave que denoten claramente que soporta este paradigma.

La sencillez es la característica principal de Go, su sintaxis es clara y concisa.



**Ruby** es un lenguaje de programación interpretado, reflexivo y orientado a objetos.

Combina una sintaxis inspirada en Python y Perl con características de programación orientada a objetos similares a Smalltalk. Comparte también funcionalidad con otros lenguajes de programación como Lisp, Lua y Dylan.

Ruby es un lenguaje de programación interpretado en una sola pasada y su implementación oficial es distribuida bajo una licencia de software libre.

## Sistemas Gestores de Base de Datos

---

La mayoría de aplicaciones web, utilizan algún sistema gestor de base de datos. Existen multitud de SGBD en el mercado, tanto libres como de pago.



**MySQL** es un SGBD relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation, y está considerado como la base de datos de código abierto más popular del mundo, sobre todo para entornos de desarrollo web.

MySQL fue inicialmente desarrollado por MYSQL AB, que fue adquirida por Sun Microsystems en 2008, y ésta a su vez fue comprada por Oracle Corporation en 2010.



**MariaDB** es un SGBD derivado de MySQL con licencia GPL. Es desarrollado por Michael Widenius (fundador de MySQL), la fundación MariaDB y la comunidad de desarrolladores de software libre.

Este SGBD surge a raíz de la compra de Sun Microsystems (compañía que había comprado previamente MYSQL) por parte de Oracle.



**PostgreSQL** es un SGBD relacional orientado a objetos y libre, publicado bajo licencia PostgreSQL, similar a la BSD o la MIT.

Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre o apoyados por organizaciones comerciales. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).



**Microsoft SQL Server** es un SGBD relacional desarrollado por la empresa Microsoft.

SQL Server ha estado tradicionalmente disponible sólo para sistemas operativos Windows de Microsoft, pero desde 2017 también está disponible para Linux y Docker containers.



**MongoDB** (del inglés humongous, "enorme") es un sistema de base de datos NoSQL, orientado a documentos y de código abierto.

En lugar de guardar los datos en tablas, tal y como se hace en las bases de datos relacionales, MongoDB guarda estructuras de datos BSON (una especificación similar a JSON) con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

MongoDB es una base de datos adecuada para su uso en producción y con múltiples funcionalidades.

El código fuente está disponible para los sistemas operativos Windows, GNU/Linux, OS X y Solaris.



## Bibliografía

---

<http://www.php.net>

[https://msdn.microsoft.com/es-es/library/4w3ex9c2\(v=vs.100\).aspx](https://msdn.microsoft.com/es-es/library/4w3ex9c2(v=vs.100).aspx)

<http://www.oracle.com/technetwork/java/index-jsp-138231.html>

<https://www.python.org>

<https://wiki.python.org/moin/SpanishLanguage>

<https://es.wikipedia.org>