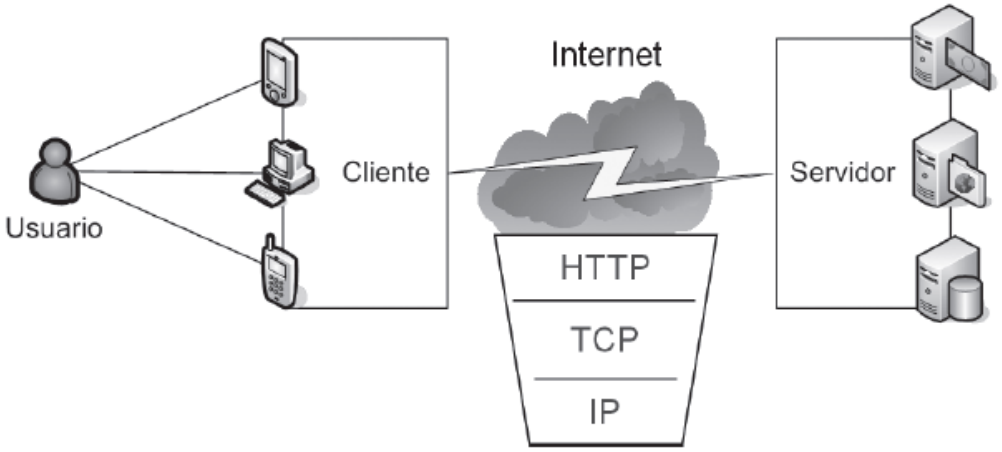
1. MODELOS DE PROGRAMACIÓN EN ENTORNOS CLIENTE/SERVIDOR

La **World Wide Web** (o la **Web**, como se conoce comúnmente) representa un universo de información accesible globalmente a través de Internet. Está formada por un conjunto de recursos interconectados que conforman el conocimiento humano actual. El funcionamiento de la Web es posible debido a la coexistencia de una serie de componentes software y hardware:

- Componentes físicos (repetidores, puentes, pasarelas, encaminadores, etc.).
- Protocolos de comunicaciones (TCP, IP, HTTP, FTP, SMTP, etc.).
- El sistema de nombres de dominio (DNS), que se utiliza para la búsqueda y recuperación de recursos.

El desarrollo en entornos web debe tener en cuenta la distribución de los elementos y la función que tiene cada uno de ellos. La configuración arquitectónica más habitual se basa en el **modelo cliente/servidor**, basado en la idea de servicio, en el que el cliente es un componente consumidor de servicios y el servidor es un proceso proveedor de servicios. Además, el cliente y el servidor se relacionan únicamente mediante el intercambio de mensajes.

El agente que solicita la información se denomina **cliente**, mientras que el componente software que responde a esa solicitud es el que se conoce como **servidor**. Normalmente, el cliente inicia el intercambio de información, solicitando datos al servidor, que responde enviando uno o más flujos de datos al cliente. Además de la transferencia de datos real, este intercambio puede requerir información adicional, como la autenticación del usuario o la identificación del archivo de datos que se vaya a transferir.

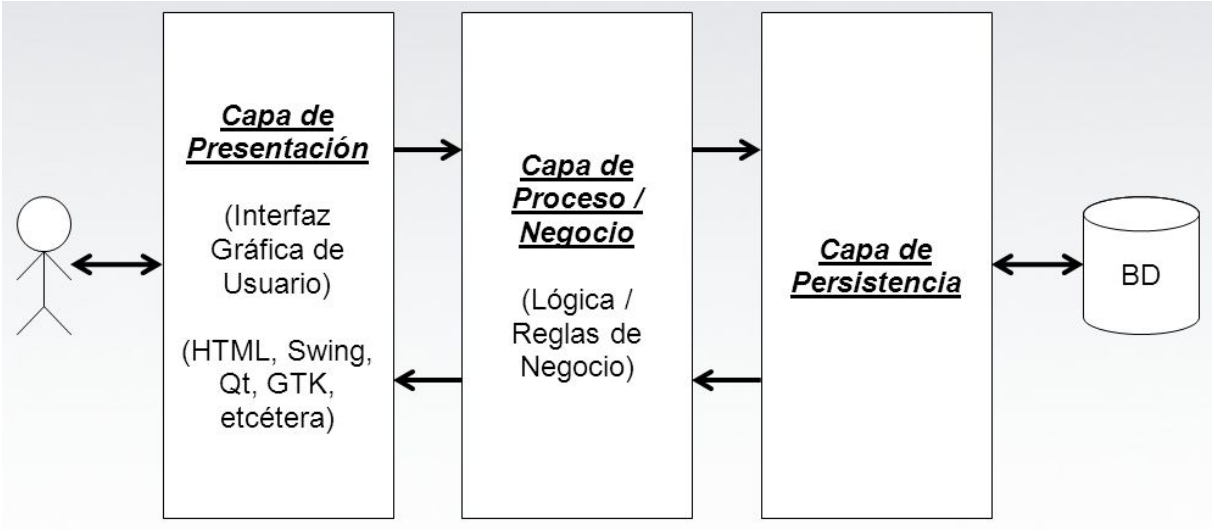


Las funcionalidades en los entornos cliente/servidor de la Web suelen estar agrupadas en diferentes capas, cada una centrada en la gestión de un aspecto determinado del sistema web. Tradicionalmente, hay tres tipos de capas fundamentales:

1) **Capa de Presentación**. Es la capa que ve el usuario. Le presenta una interfaz gráfica del recurso solicitado y sirve para recoger su interacción. Suele estar situada en el cliente. La programación de esta capa se centra en dar formato a la información enviada por el servidor y capturar las acciones realizadas por el cliente.

2) **Capa de Negocio.** Es la capa que conoce y gestiona las funcionalidades que se espera del sistema o aplicación web (lógica o reglas de negocio). Habitualmente, es donde se reciben las peticiones del usuario y desde donde se envían las respuestas apropiadas tras el procesamiento de la información proporcionada por el cliente. Al contrario que la capa de presentación, la lógica de negocio puede ser programada tanto en el entorno cliente como en el entorno servidor.

3) **Capa de Persistencia (Almacenamiento de Datos).** Es la capa donde residen los datos y la encargada de acceder a los mismos. Normalmente, está formada por uno o más gestores de bases de datos que realizan todo el proceso de administración de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.



El diseño de estas capas influye a la hora de seleccionar uno u otro modelo de programación en la Web. Dichos modelos de programación cliente/servidor se pueden clasificar de este modo atendiendo a diferentes criterios:

1) **Según el tamaño de los componentes.** Esto hace referencia a qué elemento de la arquitectura web debe soportar más o menos carga de procesamiento. Hay varias configuraciones:
   - *Fat Client (Thin Server).* El mayor peso de la aplicación se ejecuta en el cliente, relegando al servidor a un mero administrador de datos.
   - *Fat Server (Thin Client).* La funcionalidad asociada al cliente está limitada a la presentación de la información enviada por el servidor.
2) **Según la naturaleza del servicio ofrecido.** Los entornos cliente/servidor se pueden clasificar en función de las capacidades ofrecidas por el servidor. Por ejemplo, existen:
   - *Servidores de Ficheros.* El objetivo del cliente es el acceso a datos contenidos en ficheros.
   - *Servidores de Bases de Datos.* Se centran en la provisión y administración de sistemas gestores de bases de datos.
   - *Servidores de Transacciones.* Están centrados en el concepto de transacción con el objetivo de que los flujos de información con el cliente se realicen en un solo mensaje solicitud/respuesta.
   - *Servidores de Objetos.* Su principal característica es la utilización de objetos intercomunicados, tanto en el cliente como en el servidor.
   - *Servidores Web.* Conforman la base del modelo World Wide Web y están basados en la existencia de clientes simples que se comunican con servidores web utilizando HTTP como protocolo de comunicación.
3) **Según el reparto de funciones entre cliente y servidor.** Las tecnologías web existentes permiten gestionar y distribuir las responsabilidades de las prestaciones funcionales entre el cliente y el servidor. Lo más habitual es tener una configuración cliente/servidor de dos o tres capas, dependiendo de si las capas de negocio y datos se agrupan (*modelo en dos capas*) o si se separan (*modelo en tres capas*). Además, la separación en dos o tres capas se puede ver tanto desde el punto de vista del software como del hardware.

2. PÁGINAS WEB Y APLICACIONES WEB

Un **sitio web** es una colección de páginas web relacionadas entre sí y comunes a un dominio o subdominio de la World Wide Web de Internet. Una **página web** es un documento:

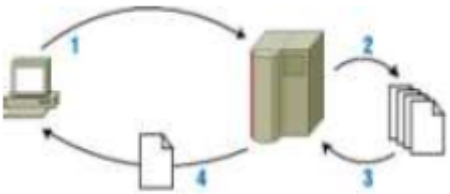
- Que está escrito en un lenguaje de marcas (HTML o XHTML), formado por etiquetas, que es accesible generalmente mediante el protocolo HTTP de Internet.
- Que puede contener una combinación de textos, imágenes, gráficos, audio, vídeo, hiperenlaces y otros materiales estáticos y dinámicos.
- Que incluye una hoja de estilos en cascada (CSS) que se utiliza para indicarle al navegador web las características de presentación de la página web.

2.1. PÁGINAS WEB ESTÁTICAS

Las **páginas web estáticas** se encuentran almacenadas en su forma definitiva, tal y como se crearon, y su contenido no varía. Son útiles para mostrar una información concreta y visualizarán esa misma información cada vez que se carguen. La única forma en que pueden cambiar consiste en que un programador las modifique para actualizar su contenido.

El **funcionamiento** de una página web estática consta de los siguientes pasos:

1) El cliente web (navegador) del ordenador solicita una página web (con extensión *.htm*, *.html* o *.xhtml*) a un servidor web mediante el protocolo HTTP.
2) El servidor web busca esta página web en un almacén de páginas (cada página web suele ser un fichero almacenado en un disco del servidor).
3) Si el servidor web encuentra esa página web, entonces la recupera.
4) El servidor web envía la página web al cliente web (navegador).
5) El cliente web (navegador) visualiza la página HTML con estilos CSS, imágenes, PDF, etc… (pero sin JavaScript).



Las páginas web estáticas tienen las siguientes **ventajas**:

- No es necesario saber programar para crear un sitio web que utilice únicamente páginas web estáticas. Simplemente habría que conocer HTML / XHTML y CSS, e incluso esto no es necesario, porque existen programas de diseño web que se pueden usar para generarlas.
- Su contenido nunca varía, por lo que son adecuadas cuando se destinan principalmente a mostrar una información permanente, donde el navegante se limita a obtener dicha información si poder interactuar con la página visitada.
- Son portables, es decir, funcionan en cualquier servidor.
- Tienen unos tiempos de acceso óptimos, pues tardan muy poco en cargarse.
- Facilitan el posicionamiento en motores de búsqueda como Google.

Las páginas web estáticas tienen los siguientes **inconvenientes**:

- La actualización de su contenido se debe hacer de forma manual editando las páginas que almacena el servidor web. Esto implica un mantenimiento que puede ser prohibitivo en sitios web con gran cantidad de contenidos.
- El visitante no tiene ninguna posibilidad de seleccionar, ordenar o modificar los contenidos o el diseño de la página a su gusto.
- No se pueden utilizar funcionalidades tales como bases de datos, foros, consultas on-line, correos inteligentes, etc.

Actualmente, las páginas web estáticas se usan para páginas personales, páginas de proyectos software y documentación técnica (JavaDoc en Java, sitio de Maven, etc.).

```
001  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
002  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es-ES" lang="es-ES">
003     <head>
004        <title>T&iacute;tulo de la p&aacute;gina</title>
005        <meta http-equiv="content-type" content="text/html; charset=utf-8">
006     </head>
007     <body>
008        <p>Ha accedido a las 9:20</p>
009     </body>
010  </html>
```

2.2. PÁGINAS WEB DINÁMICAS

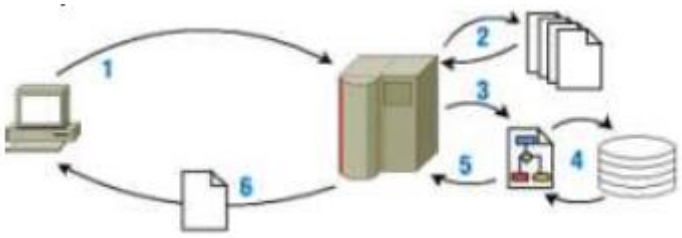
Las **páginas web dinámicas** se caracterizan porque su contenido cambia en función de diversas variables, como el navegador utilizado, el usuario con el que se ha iniciado sesión o las acciones que se han efectuado con anterioridad.

Se pueden distinguir dos tipos de páginas web dinámicas:

- **Aquellas que incluyen código que ejecuta el navegador web**. En estas páginas, el código ejecutable (normalmente escrito en lenguaje JavaScript) se incluye dentro del HTML o XHTML y se descarga junto con la página. Cuando el navegador muestra la página en pantalla, ejecuta el código que la acompaña. Este código puede incorporar diversas funcionalidades como:
  - Efectos gráficos que no se pueden implementar con CSS.
  - Mostrar u ocultar información en función de los elementos que se seleccionan (para documentos largos).
  - Menús desplegables.
  - Páginas adaptables (*responsive*) para dispositivos móviles.
- **Aquellas cuyo contenido se forma como resultado de la ejecución de un programa.** Estas páginas tienen extensiones *.php*, *.asp*, *.aspx*, *.jsp*, o *.cgi*, y no están almacenadas en el servidor web. El contenido que se descarga al navegador web es similar al de una página web estática (HTML o XHTML) y se genera como resultado de la ejecución de un programa en el servidor web (aunque no necesariamente por ese mismo servidor).

El **funcionamiento** de una página web dinámica consta de los siguientes pasos:

1) El cliente web (navegador) del ordenador solicita una página web a un servidor web.
2) El servidor web busca esa página web y la recupera.
3) Si se trata de una página web dinámica (su contenido debe ejecutarse para obtener el HTML que se va a devolver), entonces 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 (como una base de datos).
5) El resultado de la ejecución será una página web no dinámica en formato HTML.
6) El servidor web envía el resultado obtenido al cliente web (navegador), que la procesa y la muestra en pantalla.



Las páginas web dinámicas tienen las siguientes **ventajas**:

- Tienen una mayor interactividad con el usuario ya que éste puede alterar en parte el diseño, los contenidos o la presentación de la página a su gusto (tipos de letra, colores o fondos, tamaño de pantalla, etc.), siempre y cuando los desarrolladores de la página hayan activado dichas funcionalidades.
- Permiten almacenar y hacer actualizaciones de la información contenida en la página, así como realizar modificaciones dinámicas de la estructura y del diseño. El coste del mantenimiento es menor que el de las páginas estáticas.
- Disponen de un gran número de funcionalidades tales como bases de datos, foros, contenido dinámico, etc.

Las páginas web dinámicas tienen los siguientes **inconvenientes**:

- Tienen unos mayores requerimientos técnicos para su alojamiento en servidores web de pago.
- Los costes de alojamiento son mayores.
- En algunos casos, el coste de desarrollo es mayor.

```
001  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
002  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es-ES" lang="es-ES">
003          <head>
004                  <title>T&iacute;tulo de la p&aacute;gina</title>
005                  <meta http-equiv="content-type" content="text/html; charset=utf-8">
006          </head>
007          <body>
008                  <p>Ha accedido a las <script type='text/javascript'>var hora=new
     Date();document.write(hora.getHours()+':'+hora.getMinutes());</script></p>
009          </body>
010  </html>
```

2.3. APLICACIONES WEB

Las **aplicaciones web** emplean páginas web dinámicas para crear aplicaciones que se ejecutan en un servidor web y que se muestran en un navegador web.

Una de las primeras aplicaciones web que aparecieron son los clientes de correo, que permiten consultar los mensajes de correo recibido y enviar otros nuevos utilizando un navegador. Hoy en día, existen aplicaciones web para multitud de tareas como procesadores de texto, gestión de tareas, edición y almacenamiento de imágenes, etc.

Las aplicaciones web presentan las siguientes **ventajas**:

- Se instalan y se ejecutan solamente en un equipo: en el servidor. Esto es suficiente para que puedan ser utilizadas de forma simultánea desde muchos equipos clientes.
- Como solo se encuentran instaladas en un equipo, su gestión es más sencilla (hacer copias de seguridad de los datos, corregir errores, actualizar).
- Se pueden utilizar en todos los sistemas que dispongan de un navegador web, independientemente de sus características o de su sistema operativo.
- Se pueden utilizar desde cualquier lugar en el que se disponga de una conexión con el servidor. En muchos casos, esto posibilita el acceso a aplicaciones desde sistemas no convencionales, como los teléfonos móviles.

Las aplicaciones web presentan los siguientes **inconvenientes**:

- La interfaz de usuario de la aplicación web es la página que se muestra en el navegador. Esto restringe las características de la interfaz a aquellas de una página web.
- Dependen de una conexión con el servidor para su utilización. Si falla la conexión, no se puede acceder a la aplicación web.
- La información que se muestra en el navegador debe ser transmitida desde el servidor. Debido a esto, cierto tipo de aplicaciones no son adecuadas para implementarse como aplicación web (por ejemplo, las aplicaciones que manejan contenido multimedia, como las de edición de vídeo).

Las aplicaciones web se pueden clasificar de acuerdo a múltiples criterios, como la configuración o arquitectura seleccionada, la tecnología o lenguaje utilizado, e incluso la interactividad de los usuarios con la interfaz presentada.

- **Aplicaciones Web Estáticas.** Hacen referencia a aquellas aplicaciones web en las que el usuario recibe una página web cuya interacción no conlleva ningún tipo de acción, ni en la propia página, ni genera respuesta alguna por parte del servidor. Se suelen diseñar utilizando el lenguaje HTML exclusivamente para la organización visual de la información.
- **Aplicaciones Web Dinámicas.** La programación de estas aplicaciones se conoce con el nombre de HTML dinámico (DHTML) y se refiere a aquellas aplicaciones en las que la interacción del cliente con el recurso recibido por parte del servidor (página web) produce algún tipo de cambio en la visualización del mismo (cambios de formato, ocultación de partes del documento, creación de elementos nuevos, etc.). Los lenguajes involucrados en este tipo de aplicaciones son HTML, CSS o variantes de JavaScript (VBScript, JScript, Flash, etc.).
- **Aplicaciones Web Interactivas.** Al contrario que en los tipos de aplicaciones anteriores, donde la interacción del usuario produce un cambio en el recurso recibido, las aplicaciones web interactivas se basan en que dicha interacción hace que se genere un diálogo entre el cliente y el servidor. Desde el punto de vista del modelo de programación, la lógica asociada al inicio y gestión de dicho diálogo puede ser ejecutada tanto en el cliente como en el servidor (e incluso en ambos).

Arquitectura	Cliente	Servidor
Página web estática	Estático. HTML y CSS	Estático. Recursos en disco duro
Página web interactiva	Dinámico. JavaScript	Estático. Recursos en disco duro
Aplicación web con cliente estático	Estático. HTML y CSS	Dinámico. Ejecución código
Aplicación web interactiva	Dinámico. JavaScript	Dinámico. Ejecución código
Aplicación web con AJAX	Dinámico. JavaScript	Dinámico. Ejecución código
Aplicación web SPA	Dinámico. JavaScript	Dinámico. Ejecución código

El tipo de aplicaciones web interactivas es el que más se utiliza actualmente en Internet y donde el número de tecnologías disponibles ha evolucionado más en los últimos años. Las tecnologías implicadas en este tipo de aplicaciones varían mucho en función de si son ejecutadas en el lado del cliente o en el lado del servidor.

En el lado del cliente se pueden utilizar:

- Lenguajes y tecnologías para la creación de aplicaciones interactivas como HTML (por ejemplo, en la forma de formularios que rellena el usuario y envía al servidor).
- Controles ActiveX (ejecución de comportamientos asociados a certificados).
- Objetos embebidos de tipo Flash (animaciones interactivas visualmente atractivas).
- Applets o AJAX (carga dinámica de contenidos).

En el lado del servidor se pueden utilizar:

- Lenguajes embebidos en código HTML (como PHP, ASP o JSP).
- Enlaces a ejecutables (tipo CGI o SSI).
- Objetos (Servlets).
- Lenguajes que separan la presentación de la lógica de negocio (ASP.Net de Microsoft).

2.4. EJECUCIÓN DE CÓDIGO EN EL SERVIDOR Y EN EL CLIENTE

Cuando el navegador solicita una página a un servidor web, es posible que antes de enviarla haya tenido que ejecutar, por sí mismo o por delegación, algún programa para obtenerla. Este programa es el que genera, en parte o en su totalidad, la página web que llega al navegador. En este caso, el código se ha ejecutado en el **entorno del servidor web**.

Además, cuando una página web llega al navegador, también es posible que incluya algún programa o fragmento de código que se debe ejecutar. Cuando ese código (normalmente en lenguaje JavaScript) se ejecuta en el navegador, puede modificar el contenido de la página o realizar otras acciones como la animación de textos u objetos de la página, o la comprobación de los datos que se introducen en un formulario. En este caso, el código se ha ejecutado en el **entorno del cliente web**.

Estas dos tecnologías se complementan una con la otra. En el ejemplo de la aplicación web de correo electrónico, el programa que se encarga de obtener los mensajes y su contenido de una base de datos se ejecuta en el entorno servidor, mientras que el navegador web ejecuta código encargado de avisar al usuario cuando quiere enviar un mensaje y se ha olvidado de poner un texto en el asunto.

Esta división es así porque el código que se ejecuta en el cliente web (navegador) no tiene acceso a los datos que se almacenan en el servidor. Por ejemplo, cuando en el navegador se desea leer un nuevo correo, el código JavaScript que se ejecuta no puede obtener de la base de datos el contenido de ese mensaje. La solución consiste en crear una nueva página web en el servidor con la información solicitada y enviarla de nuevo al navegador.

Sin embargo, recientemente ha surgido una técnica de desarrollo web con **AJAX**, que permite realizar programas en los que el código JavaScript que se ejecuta en el navegador puede comunicarse con el servidor para obtener información con la que, por ejemplo, modificar la página web actual.

En el ejemplo descrito anteriormente, cuando se pulsa con el ratón encima de un correo que se desea leer, la página puede contener código JavaScript que detecte la acción y consultar a través de Internet el texto que contiene ese mismo correo para mostrarlo en la misma página, modificando así su estructura si es necesario. Es decir, sin salir de una página web, se puede modificar su contenido en base a la información almacenada en un servidor de Internet.

3. TECNOLOGÍAS DE DESARROLLO WEB DEL LADO SERVIDOR

Los componentes principales necesarios para ejecutar una aplicación web en un servidor son los siguientes:

- **Servidor Web.** Recibe las peticiones del cliente web (navegador) y le envía la página solicitada (una vez generada, puesto que se trata de una página web dinámica). El servidor web debe conocer el procedimiento a seguir para generar la página web: qué modulo se encarga de la ejecución del código y cómo se debe comunicar con él.
- **Módulo o Programa.** Es el encargado de ejecutar el código y generar la página web resultante. Este módulo se debe integrar de alguna forma con el servidor web y depende del lenguaje y tecnología que se utiliza para programar la aplicación web.
- **Gestor de Base de Datos.** Normalmente también será un servidor. Este componente no es estrictamente necesario, pero en la práctica se utiliza en aquellas aplicaciones web que precisan utilizar grandes cantidades de datos.
- **Lenguaje de Programación.** Necesario para desarrollar la aplicación web.

3.1. ARQUITECTURAS Y PLATAFORMAS

Existen diferentes arquitecturas que se pueden utilizar:

1) **Java EE (Enterprise Edition**). También conocida con J2EE, es una plataforma orientada al desarrollo 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.

   Está apoyada por grandes empresas como Sun y Oracle (que mantienen Java) o IBM. Es una buena solución para el desarrollo de aplicaciones de tamaño mediano o grande. Tiene la ventaja de que existe una multitud de librerías para ese lenguaje y una gran base de programadores que lo conocen.

   Dentro de esta arquitectura, hay 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.

2) **AMP.** Es la combinación de tres componentes: el servidor web Apache, el servidor de bases de datos MySQL y el lenguaje de programación utilizado PHP, Perl o Python.

   Según el sistema operativo que se use 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*). Es una plataforma de programación que permite desarrollar aplicaciones de tamaño pequeño o mediano con un aprendizaje sencillo. Su gran ventaja es la comunidad que la soporta y la multitud de aplicaciones de código libre disponibles.

   Existen paquetes software que incluyen en una única instalación una plataforma AMP completa. Algunos ni siquiera es necesario instalarlos e incluso disponen de versiones para distintos sistemas operativos como Windows, Linux o Mac. Uno de los más conocidos es XAMPP.

3) **CGI/Perl.** Es la combinación de dos componentes: el potente lenguaje de código libre Perl, creado originariamente para la administración de servidores, y el estándar CGI que permite al servidor web ejecutar programas genéricos, que están escritos en cualquier lenguaje (también se puede utilizar C o Python) y que devuelven páginas web (HTML) como resultado de su ejecución.

   Esta arquitectura es la más primitiva. El principal inconveniente de esta combinación es que CGI es lento (aunque existen métodos para acelerarlo). Por otro lado, Perl es un lenguaje muy potente con una amplia comunidad de usuarios y mucho código libre disponible.

4) **ASP.Net.** Es la arquitectura comercial propuesta por Microsoft para el desarrollo de aplicaciones. Proviene de la evolución de la anterior tecnología de Microsoft, ASP (*Active Server Pages*), y forma parte de la plataforma .Net, destinada a la generación de páginas web dinámicas.

   La arquitectura utiliza el servidor web de Microsoft, IIS (*Internet Information Server*), puede obtener información de varios gestores de bases de datos (como Microsoft SQL Server) y se pueden utilizar los lenguajes de programación Visual Basic.Net o C#.

   Una de las mayores ventajas de la arquitectura .Net es que incluye todo lo necesario para el desarrollo y el despliegue de aplicaciones. Por ejemplo, tiene su propio entorno de desarrollo, Visual Studio, aunque hay otras opciones disponibles. La principal desventaja es que se trata de una plataforma comercial de código propietario.

Hay muchas decisiones que se deben tomar antes de comenzar el desarrollo de una aplicación web, como por ejemplo la arquitectura de la aplicación, el lenguaje de programación, el entorno de desarrollo, el gestor de bases de datos o el servidor web.

Para seleccionar la arquitectura de programación web más adecuada, se deben considerar los siguientes aspectos:

- ¿Qué tamaño tiene el proyecto?
- ¿Qué lenguajes de programación se conocen? ¿Vale la pena el esfuerzo de aprender uno nuevo?
- ¿Se van a utilizar herramientas de código abierto o herramientas privativas? ¿Cuál es el coste de usar soluciones comerciales?
- ¿Se va a programar la aplicación de forma individual o se va a formar parte de un grupo de programadores?
- ¿Se dispone de algún servidor web o gestor de bases de datos o se puede decidir libremente utilizar el que se crea necesario?
- ¿Qué tipo de licencia se va a aplicar a la aplicación que se desarrolle?

Estudiando las respuestas a estas preguntas, se podrá ver qué arquitecturas se adaptan mejor a la aplicación web y cuáles no son viables.

3.2. INTEGRACIÓN CON EL SERVIDOR WEB

En muchas ocasiones, la posibilidad de responder a una petición hecha por un cliente depende de las capacidades que tenga el servidor web y los módulos o extensiones que tenga instalados.

El objetivo principal de un servidor web es proveer de contenido estático a un cliente que ha realizado una petición a través de un navegador. Para ello, carga un archivo y lo sirve a través de la red al navegador del solicitante. Este es el funcionamiento habitual cuando lo que se implementa en el servidor es una aplicación web estática (HTML) o dinámica (DHTML), es decir, el servidor se convierte en un instrumento que proporciona un lugar para guardar y administrar los recursos HTML, que pueden ser accesibles por los usuarios de la red a través de navegadores.

Para que el servidor pueda entender la petición del cliente, éste tiene que realizar una petición formal. Es decir, la petición tiene que constar de unos elementos concretos y especificados en un orden determinado. Las direcciones de las peticiones suelen ser de tipo **URL (*Uniform Resource Locator*)**, localizador uniforme de recursos.

Un URL contiene la referencia a un cierto protocolo de red (http, https, ftp, etc.) y la dirección de un recurso concreto en forma de ruta al objeto que se desea descargar:

```
esquema://máquina.directorio.archivo
```

Opcionalmente, se pueden incluir otros datos, como el nombre y la contraseña del usuario que accede y el número de puerto por el que el servidor está escuchando las peticiones del cliente:

```
esquema://usuario:contraseña@máquina:puerto.directorio.archivo
```

Respecto a las peticiones de los clientes, existen distintos modos o métodos para intercambiar información entre cliente y servidor:

- **GET.** Es un método de invocación en el que el cliente le solicita al servidor web que le devuelva la información identificada en la propia URL. Lo más común es que las peticiones se refieran a un documento HTML o a una imagen, aunque también pueden hacer referencia a un programa de base de datos. En tal caso, el servidor ejecuta ese programa y le devuelve al cliente el resultado generado tras esa petición.
- **POST.** Mientras que el método GET se utiliza para recuperar información, el método POST se usa habitualmente para enviar información a un servidor web. Estos casos suelen darse al enviar el contenido de un formulario de autenticación, así como entradas de datos o especificar parámetros para algún tipo de componente ejecutado en el servidor.

La potencia de los servidores web aparece cuando se quiere desarrollar aplicaciones web interactivas y con una cierta complejidad. En este caso, se pueden utilizar varias tecnologías implementadas en módulos específicos en el servidor para aumentar su potencia, más allá de su capacidad de entregar páginas HTML. Dichas tecnologías se suelen instalar como extensiones en el software del servidor e incluyen soporte para scripts CGI, proporcionan seguridad SSL, permiten la ejecución de scripts e interactúan con la máquina virtual de Java.

4. LENGUAJES DE PROGRAMACIÓN EN ENTORNO SERVIDOR

Un lenguaje de programación correspondiente a un entorno servidor es aquel cuyo código, bien sea como objeto precompilado o bien como código interpretado, es ejecutado por un software específico en el componente que actúa como servidor.

Existen múltiples alternativas a la hora de ejecutar código en el servidor:

1) Utilizar **lenguajes de scripting o embebidos** (SSI, LiveWire, ASP, PHP, etc.), cuya característica principal es que el código es interpretado y que se intercala con una plantilla de código HTML con la estructura básica de la página que se envía al cliente.
2) Utilizar **enlaces a programas y componentes ejecutables** (CGI, JSP, EJB, etc.), de tal forma que el código ejecutado por el servidor está almacenado en unidades precompiladas y ejecutadas de forma independiente, que generan las páginas enviadas al cliente.
3) Utilizar **estrategias "híbridas" basadas en técnicas de respaldo** (denominadas *code-behind*), como ASP.Net de Microsoft, en el que algunos elementos de código se intercalan con la lógica de presentación mientras que se mantiene la funcionalidad de dichos elementos en ficheros o librerías independientes en el servidor.

4.1. LENGUAJES DE SCRIPTING O EMBEBIDOS

Un **lenguaje de scripting o embebido** es aquel que se intercala con el código HTML que forma una aplicación web. La idea básica es crear una serie de plantillas HTML en las que se insertan instrucciones de programación en diferentes lenguajes que son ejecutadas por un servidor para determinar las partes dinámicas de las páginas web. Para que estas porciones de código sean ejecutadas, el agente que actúa de servidor (servidor web) debe tener instalado un módulo (*scripting engine*) que sea capaz de reconocer e interpretar el lenguaje en el que se programa dicho código.

Existen diversos lenguajes de scripting que se pueden utilizar para la creación de aplicaciones web dinámicas. Entre ellos destacan PHP, ASP, Perl, Python y JSP:

- **PHP (Hypertext Processor).** Es uno de los lenguajes más extendidos actualmente. Cuenta con una comunidad de desarrolladores muy importante debido a sus características de gratuidad, código abierto, y la posibilidad de ser portado y ejecutado en diferentes plataformas. Es un lenguaje imperativo de tipos dinámicos, con la posibilidad de utilizar construcciones orientadas a objetos. Es soportado por la gran mayoría de servidores web actuales.
- **ASP (Active Server Pages).** Se trata de una tecnología propietaria y de código cerrado para el lado del servidor de Microsoft. Su última versión es la 3.0 y data del año 2002, año a partir del cual se empezó a sustituir progresivamente por la versión ASP.Net. Aunque puede ser ejecutado en otros servidores web, ASP está diseñado especialmente para ser utilizado con IIS (Internet Information Server). El lenguaje utilizado con ASP es Visual Basic en su versión de scripting (VBScript), aunque también pueden usarse otros (JScript).
- **Perl.** Fue inicialmente concebido como un lenguaje de manipulación de cadenas de caracteres basado en un estilo de programación por bloques, como C o AWK. Perl es un lenguaje imperativo, con variables, expresiones, asignaciones, bloques de código delimitados por llaves, estructuras de control y subrutinas actualmente orientado a la generación dinámica de páginas web. Fue uno de los primeros lenguajes en ser utilizados para la programación web (para crear aplicaciones CGI). Es de código abierto y cuenta con una comunidad bastante numerosa de seguidores.
- **Python.** Es un lenguaje de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programas, desde aplicaciones Windows a servidores de red, o incluso páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo. Esto ofrece ventajas, como la rapidez de desarrollo, e inconvenientes, como una menor velocidad. En los últimos años, el lenguaje se ha hecho muy popular gracias a la cantidad de librerías, tipos de datos y funciones incorporadas en el propio lenguaje o la sencillez y velocidad con la que se crean los programas. Además, Python es un lenguaje gratuito, es decir, no necesita de ningún software de pago adicional para su ejecución.

- **JSP (Java Server Pages).** El funcionamiento de Java como lenguaje de script es similar al de PHP o ASP: se trata de porciones de código Java intercalado con código HTML estático. Sin embargo, la forma de interpretarlo es diferente. Una vez que el módulo encargado de ejecutar la página JSP llega a una porción de código Java, lo transforma a un *servlet* (porción de código Java cargado en la memoria de un servidor web) y lo ejecuta obteniendo el código que ha de enviarse al cliente. La diferencia con otros lenguajes es que el *servlet* queda cargado en memoria por si llegara otra petición a la misma página JSP, con lo que el rendimiento se ve mejorado notablemente.

Los lenguajes de scripting o embebidos 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.

4.2. APLICACIONES CGI Y DERIVADOS

Un **lenguaje compilado a código nativo** es aquel en el cual el código fuente se traduce a código binario, dependiente del procesador, antes de ser ejecutado. El servidor web almacena los programas en su modo binario, que ejecuta directamente cuando se invocan.

La forma más simple de generar páginas dinámicas es delegar la creación de las mismas a un programa externo, que reciba ciertos parámetros de entrada y devuelva como resultado el contenido que debe visualizar el cliente. El estándar **CGI (Common Gateway Interface)** define precisamente este comportamiento, estableciendo los vínculos que hay que establecer con una aplicación independiente o fichero ejecutable. Puesto que el programa externo no depende del código a generar, el lenguaje elegido puede ser cualquiera (como C o C++). La localización del *script* o fragmento de programa a ejecutar se indica en la URL que forma la petición HTTP del cliente.

Sin embargo, esta forma de crear aplicaciones web presenta ciertas desventajas, siendo la principal de ellas el **escaso rendimiento a la hora de responder a múltiples peticiones** CGI simultáneamente, puesto que para cada una se tiene que crear un proceso nuevo en el servidor con los costes de memoria y uso de procesador que conlleva. Respecto a este problema, han surgido algunas alternativas como son:

- El uso de extensiones para cada uno de los lenguajes en los que se programe el CGI, pero como parte integrante del servidor web.
- El uso de FastCGI como forma de crear un único proceso que atienda a todas las peticiones CGI.

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. Por ejemplo, no se reutilizan los procesos para atender a varias peticiones. Además, los programas no son portables entre distintas plataformas.

Un **lenguaje compilado a código intermedio** es aquel en el cual el código fuente se traduce a un código intermedio, independiente del procesador, antes de ser ejecutado. Esto posibilita que las aplicaciones se puedan ejecutar en varias plataformas distintas.

Otra alternativa que se ha propuesto como solución a los problemas de CGI es el uso de **plataformas independientes de ejecución de código**. El principio de estas plataformas es equivalente al uso de extensiones para servidores web. Es el caso de la programación con Java en el lado del servidor. En este contexto, la arquitectura de un servidor que es capaz de ejecutar programas Java (*Servlets*, *JavaBeans*, etc.) está formada por el propio servidor web en sí mismo y la Máquina Virtual de Java (JVM), que es capaz de ejecutar un programa especial Java (llamado *Servlet container*) encargado de gestionar datos de sesión y *Servlets*.

Cuando un desarrollador programa en Java, tiene a su disposición usar los denominados **Enterprise Java Beans (EJB)**, que son componentes programados en Java que implementan la lógica de negocio de la aplicación utilizando las características de la plataforma J2EE. Su especificación detalla cómo los servidores de aplicaciones proveen objetos desde el lado del servidor que son, precisamente, los EJB. El uso de EJB está especialmente indicado en los casos en los que se requiera mantener la transaccionalidad de las peticiones hechas por un cliente en un entorno web.

Los lenguajes compilados a código intermedio ofrecen un equilibrio entre las dos opciones anteriores. Su rendimiento es muy bueno y se pueden portar entre distintas plataformas en las que exista una implementación de la arquitectura.

4.3. APLICACIONES HÍBRIDAS DE CÓDIGO REPARTIDO

Como alternativa a los lenguajes de scripting (interpretados) y a las aplicaciones CGI y derivadas (ejecutadas por un programa o módulo independiente del servidor web), en los últimos años ha surgido una tecnología intermedia que se puede denominar híbrida. La solución más representativa es la plataforma de Microsoft .Net Framework a través de ASP.Net.

ASP.Net es una tecnología totalmente orientada a objetos que puede ser escrita en cualquier lenguaje soportado por el entorno .Net Framework (VB.Net; C# y JScript.Net). Desde el punto de vista del rendimiento, la aplicación se precompila en una sola vez al lenguaje nativo y, luego, en cada petición tiene una compilación *just in time*, es decir, se compila desde el código nativo, lo que permite mucho mejor rendimiento.

Las páginas de ASP.Net, conocidas oficialmente como **Web Forms (formularios web)**, son el principal medio de construcción para el desarrollo de aplicaciones web desde el punto de vista de Microsoft. Los formularios web están contenidos en archivos con una extensión ASPX que son los que el cliente solicita a través de una URL al servidor. Estos ficheros ASPX contienen código HTML o estático y también etiquetas propias de la plataforma .Net. Estas etiquetas definen *controles web* que se procesan del lado del servidor y *controles de usuario* donde los desarrolladores colocan todo el código estático y dinámico requerido por la página web.

Además, el código dinámico que se ejecuta en el servidor puede ser colocado en una página dentro de un bloque “<% -- código dinámico -- %>”, que es muy similar a otras tecnologías de *scripting* como PHP, JSP y ASP. Esta circunstancia es otra de las razones por las que se considera la programación con ASP.Net híbrida. Generalmente, esta práctica se desaconseja (excepto para propósitos de enlace de datos), pues requiere más llamadas cuando se genera la página.

5. HERRAMIENTAS DE PROGRAMACIÓN

En función de sus capacidades y funcionalidades, las distintas herramientas de las que dispone un desarrollador se pueden clasificar:

- **Navegadores Web.** Son las aplicaciones que se ejecutan en el entorno del cliente y que permiten visualizar los documentos escritos en lenguaje HTML y capturar las interacciones del usuario.
- **Marcadores de Texto.** Son editores de texto que permiten escribir código en cualquier lenguaje de programación directamente, sin ninguna ayuda ni facilidad adicional.
- **Entornos Integrados de Desarrollo (IDE).** Son entornos integrados que nos permiten editar, compilar y ejecutar los programas generados a partir de diferentes lenguajes usados en el desarrollo de las aplicaciones web.
- **Herramientas de Tratamiento de Imágenes.** La mayoría de las páginas web muestran contenido gráfico de una u otra manera. Por ello, es necesario el uso de este tipo de herramientas para adecuar las características de las imágenes a su transmisión por la red y su posterior visualización.
- **Herramientas de la Creación y Administración de Bases de Datos.** Se trata de herramientas para la carga de datos y el mantenimiento posterior de los datos almacenados.

5.1. MARCADORES DE TEXTO

Los **marcadores de texto** (*text markers*) son simples editores de texto, aunque dirigidos al ámbito de la programación. Al escribir el programa, y dependiendo del lenguaje de programación utilizado, el editor de texto nos ayuda a identificar mejor la sintaxis del lenguaje, cambiando de color las etiquetas, realizando tabulaciones en el texto, etc. Algunas de estas herramientas son:

- **Arachnophilia.** Es un robusto editor de texto, ideal para programar sitios web en diferentes lenguajes. Soporta HTML, JavaScript, C++, CGI, Perl, Java, entre otros. También incluye un cliente inteligente de FTP para subir automáticamente los archivos modificados.
- **Notepad++.** Es un editor gratuito de código fuente. Soporta varios lenguajes de programación y se ejecuta en Windows.
- **UltraEdit.** Es un completo editor de texto para programación. Soporta múltiples formatos con colores configurados para cada lenguaje. Es uno de los múltiples editores de pago que existen en el mercado.
- **Sublime Text.** Es un editor de texto para código fuente. Soporta muchos lenguajes de programación y de marcas de forma nativa. No es software libre o de código abierto, pero permite descargar una versión de evaluación de forma gratuita que es plenamente funcional y que no tiene fecha de caducidad.

5.2. ENTORNOS INTEGRADOS DE DESARROLLO

En el ámbito del desarrollo de aplicaciones web, las herramientas más importantes son los **entornos integrados de desarrollo (IDE)**, que agrupan en un único producto software el conjunto de herramientas necesarias para el desarrollo de aplicaciones (editor de textos, compilador o intérprete, depurador, interfaz gráfica), haciendo mucho más fácil para el programador su tarea.

Los entornos de desarrollo integrados (IDE) aportan las siguientes **características**:

- **Resaltado de Texto.** Muestra con distinto color o tipo de letra los diferentes elementos del lenguaje: sentencias, variables, comentarios, etc. También genera indentado automático para diferenciar de forma clara los distintos bloques de un programa.
- **Completado Automático.** Detecta qué se está escribiendo y cuando es posible, muestra distintas opciones para completar el texto.

- **Navegación en el Código.** Permite buscar de forma sencilla elementos dentro del texto, por ejemplo, definiciones de variables.
- **Comprobación de Errores al Editar.** Reconoce la sintaxis del lenguaje y revisa el código en busca de errores mientras se escribe.
- **Generación Automática de Código.** Ciertas estructuras, como la que se utiliza para las clases, se repiten varias veces en un programa. La generación automática de código puede encargarse de crear la estructura básica, para que sólo se tenga que rellenarla.
- **Ejecución y Depuración.** Esta característica es una de las más útiles. El IDE se puede encargar de ejecutar un programa para poder probar su funcionamiento. Además, cuando algo no funciona, permite su depuración con herramientas como la ejecución paso a paso, el establecimiento de puntos de ruptura o la inspección de los valores que se almacenan en las variables.
- **Gestión de Versiones.** En conjunción con un sistema de control de versiones, el entorno de desarrollo puede ayudar a guardar copias del estado del proyecto a lo largo del tiempo, para que, si es necesario, se puedan revertir los cambios realizados.

Algunos entornos de desarrollo son específicos de una plataforma o de un lenguaje, como ocurre con Visual Studio, el IDE de Microsoft para desarrollar aplicaciones en lenguaje C# o Visual Basic para la plataforma .Net. Otros entornos de desarrollo, como Eclipse o NetBeans, permiten personalizar el entorno para trabajar con diferentes lenguajes y plataformas.

- **Eclipse.** Se trata de un entorno de programación gratuito escrito en Java que permite desarrollar aplicaciones en varios lenguajes (C, C++, Java, PHP). Una de sus principales características es que permite la extensión de sus funcionalidades mediante la instalación de múltiples módulos para diferentes propósitos específicos.

- **NetBeans.** Es una aplicación de código abierto diseñada para el desarrollo de aplicaciones fácilmente portables entre distintas plataformas haciendo uso de la tecnología Java. Se trata de un entorno de desarrollo gratuito optimizado para el desarrollo de aplicaciones con Java.

- **Visual Studio.** Es el entorno de desarrollo más conocido para el diseño de aplicaciones en los sistemas operativos de Microsoft. Está destinado al desarrollo de aplicaciones web con lenguajes como Visual Basic o C# y permite la publicación de la aplicación web que se está desarrollando directamente desde su interfaz.