1 Selección de arquitecturas y herramientas de programación

OBJETIVOS DEL CAPÍTULO

- ✓ Caracterizar y diferenciar los modelos de ejecución de código en un entorno cliente/servidor.
- ✓ Conocer los mecanismos de ejecución de código en función de las diferentes tecnologías web.
- ✓ Reconocer las ventajas de la generación dinámica de páginas web.
- ✓ Identificar y caracterizar los principales lenguajes y tecnologías de programación en entorno servidor.
- ✓ Conocer las distintas herramientas de programación web para lenguajes de servidor.

En este primer capítulo presentamos los conceptos necesarios para comprender el contexto del desarrollo de aplicaciones y sistemas de información web. Para ello, introducimos los modelos de programación más comunes en la Web, así como los diferentes lenguajes y tecnologías de programación, del lado del servidor, aplicables en este tipo de entornos. Por último, haremos un recorrido por algunas de las herramientas que el desarrollador tiene a su alcance para sacarle el máximo partido a la programación web.

1.1 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. Estos elementos abarcan desde los componentes físicos de Internet (*hubs*, repetidores, puentes, pasarelas, encaminadores, etc.) y los protocolos de comunicaciones (TCP, IP, HTTP, FTP, SMTP, etc.) hasta la utilización del sistema de nombres de dominio (DNS) para la búsqueda y recuperación de recursos o la utilización de software específico para proveer y consumir dichos recursos.

En este contexto, 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 denominado *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, esta relación está robustamente cimentada en el intercambio de mensajes como el único elemento de acoplamiento entre ambos.

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*. En un proceso habitual el cliente es el que 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 vayamos a transferir. La Figura 1.1 muestra este proceso genérico.

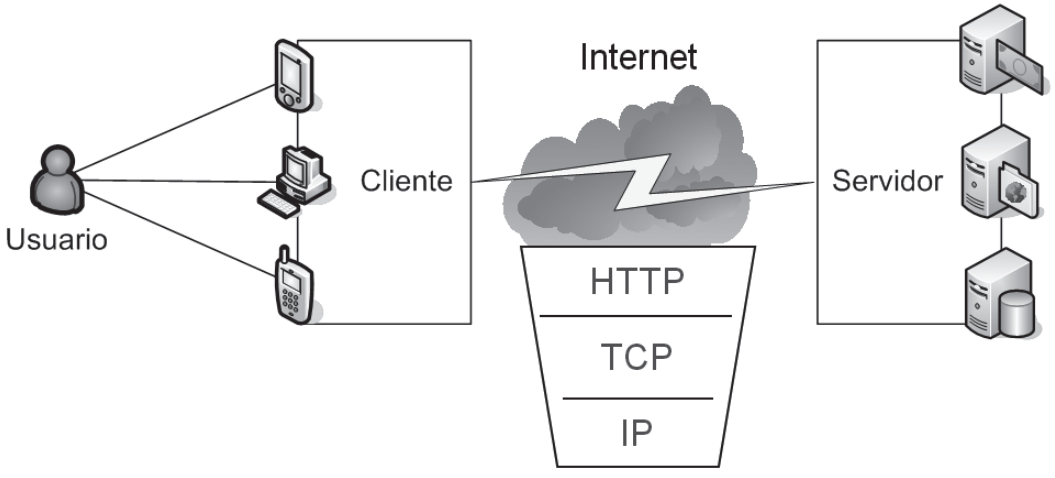


Figura 1.1. *Vista general del modelo cliente/servidor*

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. Si bien es posible encontrarse con divisiones más o menos especializadas, tradicionalmente se identifican tres tipos de capas fundamentales: *capa de presentación*, *capa de la lógica de negocio* y *capa de persistencia* (almacenamiento de datos). Los modelos arquitectónicos de programación se pueden clasificar en función de en qué lugar, si en el cliente o en el servidor, se sitúan cada una de estas capas. La decisión de dónde se sitúa cada una de estas capas dependerá del entorno de ejecución y, por tanto, de las tecnologías y lenguajes utilizados.

- **Capa de presentación**: esta capa es la 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 el formateo de la información enviada por el servidor y la captura de las acciones realizadas por el cliente.
- **Capa de negocio**: es la capa que conoce y gestiona las funcionalidades que esperamos 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.
- **Capa de persistencia o 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 cada una de estas capas influye a la hora de seleccionar uno u otro modelo de programación en la Web. Dichos modelos de programación cliente/servidor pueden clasificarse de este modo atendiendo a diferentes criterios. Así, es posible clasificarlos dependiendo del *tamaño* de los componentes, por la *naturaleza* del servicio ofrecido o según el *reparto de funciones* entre cliente y servidor:

- **Según el tamaño de los componentes**: esta primera clasificación hace referencia a qué elemento de la arquitectura web debe soportar más o menos carga de procesamiento. Se habla de configuraciones *Fat Client* (*thin Server*), donde el mayor peso de la aplicación se ejecuta en el cliente relegando al servidor a un mero administrador de datos; o *Fat Server* (*thin client*), donde la funcionalidad asociada al cliente está limitada a la presentación de la información enviada por el servidor.
- **Según la naturaleza del servicio ofrecido**: también es posible clasificar los entornos cliente/servidor en función de las capacidades ofrecidas por el servidor. De esta forma, podemos encontrar *servidores de ficheros*, donde el objetivo del cliente es el acceso a datos contenidos en ficheros; *servidores de bases de datos*, que se centran en la provisión y administración de sistemas gestores de bases de datos; *servidores de transacciones*, 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*, cuya principal característica es la utilización de objetos intercomunicados, tanto en el cliente como en el servidor; o *servidores web*, que conforman la base del modelo *World Wide Web* y que está fundamentado en la existencia de clientes simples que se comunican con servidores web utilizando HTTP como protocolo de comunicación.
- **Reparto de funciones entre cliente y servidor**: las diferentes tecnologías web existentes permiten gestionar y distribuir las responsabilidades de cada una 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*). La separación en dos o tres capas la podemos ver, además, tanto desde el punto de vista del software como del hardware.

1.2 GENERACIÓN DINÁMICA DE PÁGINAS WEB

El contenido que visualiza un cliente se obtiene debido a su interacción con una aplicación que es gestionada por el servidor. Las aplicaciones web pueden ser clasificadas de acuerdo a múltiples criterios: dependiendo de la *configuración* o modelo arquitectónico seleccionado, según la *tecnología* o *lenguaje utilizado* e, incluso, desde un punto de vista más *funcional*, es decir, observando la capacidad de comunicación con los usuarios en respuesta a sus interacciones con la interfaz que se les presenta. Son estos criterios los que dan lugar a la clasificación de las aplicaciones web en *estáticas*, *dinámicas* e *interactivas*.

- **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. Este tipo de aplicaciones suele realizarse utilizando el lenguaje HTML exclusivamente para la organización visual de la información.

- **Aplicaciones web dinámicas**: la programación de estas aplicaciones suele conocerse 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 incluyen, entre otros, HTML, CSS o las múltiples variaciones 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).

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 encontramos 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), entre otros.

En el lado del servidor es posible utilizar lenguajes embebidos en código HTML (como PHP, ASP o JSP), enlaces a ejecutables (tipo CGI o SSI), objetos (*Servlets*) e incluso lenguajes que separan la presentación de la lógica de negocio (ASP.Net de Microsoft).

1.3 LENGUAJES DE PROGRAMACIÓN EN ENTORNO SERVIDOR

Se entiende por lenguaje de programación correspondiente a un entorno servidor a 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. Una de ellas es utilizar lenguajes de *scripting* (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. Otra alternativa es el uso de 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. Otras técnicas de programación en el lado del servidor incluyen el uso de 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.

1.3.1 LENGUAJES DE SCRIPTING

Se entiende por lenguaje de *scripting* (o embebido) 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 pueden utilizarse 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, la posibilidad de ser portado y ejecutado en diferentes plataformas, etc. Se define como 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 necesitamos 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 incluso para propósitos empresariales.
- **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.

Existen muchos otros lenguajes de *scripting* en el lado del servidor enfocados a entornos más específicos y con características particulares, como ColdFusion Markup Language, Lua, Ruby, SMX, Lasso, WebDNA o Progress WebSpeed.

1.3.2 APLICACIONES CGI Y DERIVADOS

Una de las formas más simples 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*), especificado en la RFC3875, 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.

Como hemos dicho, CGI es la forma más simple de publicar contenido dinámico en la Web; 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 (módulos NSAPI o ISAPI por ejemplo), o el uso de *FastCGI* como forma de crear un único proceso que atienda a todas las peticiones CGI.

Otra de las alternativas que se han 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*.

Otra de las alternativas que un desarrollador tiene a su disposición cuando programa con Java es 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.

1.3.3 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 podríamos denominar como "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.

Adicionalmente, 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 como "híbrida". En algunos entornos de alto rendimiento y seguridad esta práctica es, generalmente, desaconsejada excepto para propósitos de enlace de datos, pues requiere más llamadas cuando se genera la página.

1.4 INTEGRACIÓN CON LOS SERVIDORES WEB

Hasta ahora hemos comentado que para el desarrollo de aplicaciones web en entornos cliente/servidor pueden utilizarse diferentes modelos, tecnologías y lenguajes. En cualquier caso, 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.

Empezando por la funcionalidad básica de un servidor web, su cometido principal 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, que 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 (*Localizador Uniforme de Recurso*), que contiene una dirección, la referencia a un cierto protocolo (HTTP, FTP, etc.) y la descripción de un recurso concreto en forma de ruta al objeto que queremos descargar. Opcionalmente podemos incluir un campo adicional (puerto) indicando el número de puerto por el que el servidor está escuchando las peticiones del cliente.

Con respecto a las peticiones de los clientes es importante destacar que existen distintos modos o métodos para intercambiar información entre cliente y servidor y que deberemos tenerlo en cuenta en el desarrollo de aplicaciones web en el entorno del servidor.

- **Método 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 se puede referir 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.
- **Método 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.

Como ya se ha comentado anteriormente, la potencia de los servidores web aparece en el momento en que queremos desarrollar aplicaciones web interactivas y con una cierta complejidad. En ese caso, podemos 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 suelen instalarse como extensiones en el software del servidor e incluyen soporte para *scripts* CGI, proporcionan seguridad SSL, permiten la ejecución de *scripts*, interactúan con la máquina virtual de Java, etc.

1.5 HERRAMIENTAS DE PROGRAMACIÓN

La selección de un entorno de desarrollo para la programación web debe tener en cuenta los diferentes escenarios, modelos y configuraciones que intervienen en este contexto. Las aplicaciones web están formadas por un conjunto de páginas HTML, *scripts* de lenguajes web, programas escritos en diferentes lenguajes, bases de datos y documentos de diferentes formatos y pueden residir en varias ubicaciones o sitios web, es decir, en diferentes servidores.

El proceso de desarrollo, sin embargo, no tiene por qué realizarse en el mismo equipo en el que finalmente se despliegue y ejecute la aplicación web que se está desarrollando. El mismo razonamiento es aplicable al proceso de pruebas en el que se simulen las peticiones al servidor web desde un cliente determinado. Para el proceso de desarrollo deberemos tener en cuenta, en el caso del servidor, el conjunto de tecnologías y lenguajes seleccionados, el tipo de servidor, los módulos y extensiones que tiene configurado, los permisos de acceso y ejecución o la localización de los datos utilizados por la aplicación web. Desde el punto de vista del cliente habrá que tener en cuenta, entre otros, la versión y tipo de navegador, la resolución gráfica o el sistema de permisos del navegador en el equipo cliente.

A la hora de seleccionar las herramientas que un desarrollador tiene a su disposición es necesario hacer una primera clasificación de los instrumentos involucrados en función de sus capacidades:

- **Navegadores**. 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.
- **Editores de documentos**. Este grupo está formado por editores de texto que permiten escribir código HTML directamente, sin ninguna ayuda ni facilidad adicional.
- **Entornos de programación**. 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 para 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.

En el ámbito del desarrollo de aplicaciones web las herramientas más importantes son los llamados *entornos de programación*, que son programas, aplicaciones o simples utilidades destinadas a la programación web, haciendo mucho más fácil para el programador su tarea. Desde el punto de vista de las herramientas que se pueden utilizar para la construcción de programas o *scripts*, podemos emplear una gran variedad de utilidades que van desde simples editores de texto hasta los entornos integrados que facilitan enormemente la construcción de aplicaciones web en lenguajes específicos.

1.5.1 MARCADORES DE TEXTO

Los marcadores de texto, o *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 ellas 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**. Completo editor de texto para programación. Soporta múltiples formatos con colores configurados para cada lenguaje. Es uno de los múltiples editores de pago que podemos encontrar en el mercado.

1.5.2 HERRAMIENTAS GENÉRICAS

Dentro de los entornos de programación existen herramientas que ofrecen funcionalidades avanzadas aparte del reconocimiento de la sintaxis del lenguaje. Estas aplicaciones están mucho más desarrolladas que los marcadores de texto y ofrecen capacidades tales como la sugerencia de estructuras o funciones predeterminadas o la posibilidad de validar la corrección del código escrito. A continuación describimos algunas de estas herramientas a modo de ejemplo:

- **Microsoft FrontPage**. Es una herramienta de construcción y edición de páginas web para el sistema operativo Windows. Si bien tuvo bastante éxito hace unos años, actualmente muchos desarrolladores consideran que el código HTML generado por esta aplicación es un poco descuidado y muchas veces reiterativo, especialmente en versiones antiguas. Forma parte del paquete ofimático Microsoft Office y es de pago.
- **Eclipse**. Se trata de un entorno de programación gratuito escrito en Java que permite desarrollar aplicaciones en varios lenguajes (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.
- **Dreamweaver**. Es un editor de código HTML profesional de pago para el diseño visual y la administración de sitios y páginas web. También incluye numerosas herramientas y funciones de edición de código: referencias HTML, CSS y JavaScript, un depurador JavaScript y editores de código (la vista de código y el inspector de código) que permiten editar JavaScript, XML y otros documentos de texto directamente en Dreamweaver.

1.5.3 HERRAMIENTAS ESPECÍFICAS

En muchas ocasiones, la utilización de un lenguaje de programación determinado exige que el desarrollador tenga instalada una plataforma de desarrollo concreta. Tal es el caso de la programación en Java que requiere la instalación de la Máquina Virtual de Java (JVM) o de el entorno .Net Framework de Microsoft, necesaria para la programación con ASP.Net. Las herramientas que mostramos a continuación se encuentran dentro de la categoría de herramientas específicas para una tecnología concreta.

- **Microsoft 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.
- **NetBeans IDE**. 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.

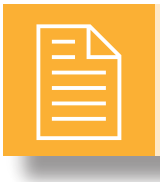
ACTIVIDADES 1.1

- Los modelos de programación por capas presentados en este capítulo se centran en la asignación de funcionalidades a nivel software. Proponemos que el alumno complemente estos modelos investigando las diferentes ventajas y desventajas que pueden existir en el caso de realizar una asignación de responsabilidades en diferentes nodos hardware.
- La arquitectura de un servidor web varía en función de la plataforma sobre la que se vaya a utilizar y las capacidades que se requieren. Algunos de los servidores web más conocidos son *Apache*, de la *Apache Foundation,* o *Internet Information Server de Microsoft.* Proponemos que el alumno busque la última versión de estos servidores y describa tanto su arquitectura básica como los mecanismos de extensión que ofrecen para el soporte de diferentes tecnologías de programación en entorno del servidor.
- Los lenguajes del entorno del servidor presentados han ido evolucionando históricamente incluyendo cada vez más funcionalidades. Proponemos que el alumno elija uno de esos lenguajes y realice una descripción detallada de su evolución, indicando cuáles son las influencias recibidas de otros lenguajes y sobre qué otros lenguajes ha influido.
- Al igual que los servidores cuentan con extensiones, las herramientas de programación web también pueden ser extendidas. Busque información sobre uno de los editores mencionados y amplíe su información indicando los métodos en los que pueden ser extendidos.

RESUMEN DEL CAPÍTULO

Para poder abordar la creación de aplicaciones web es necesario conocer antes el contexto en el que se desarrollan estas aplicaciones. Así, podemos ver que el modelo arquitectónico en el que se basan es el denominado "cliente/servidor", donde un cliente, a través de un navegador, realiza peticiones a un servidor, que será el encargado de gestionar su petición y devolver una respuesta adecuada. Las aplicaciones desarrolladas en este contexto pueden ser muy sencillas (aplicaciones web estáticas) o muy complejas (aplicaciones web dinámicas e interactivas). La selección de una u otra tecnología, o lenguaje, dependerá principalmente del modelo de programación escogido, es decir, del reparto de funcionalidades entre el cliente y el servidor.

Los lenguajes utilizados en el entorno del servidor se agrupan en lenguajes de *scripting* (interpretados como PHP, ASP, Perl o Python), lenguajes para aplicaciones ejecutadas de forma independiente del servidor web (CGI y derivados) o lenguajes que optan por una programación "híbrida" (ASP.Net). Finalmente, para el desarrollo óptimo de aplicaciones web se suelen utilizar herramientas específicas que ayudan al desarrollador en su labor.



TEST DE CONOCIMIENTOS

1 ¿Cuál de los siguientes lenguajes no puede utilizarse como lenguaje de *scripting*?
- **a)** PHP.
- **b)** Perl.
- **c)** HTML.
- **d)** JSP.

2 ¿Cuál de las siguientes estrategias representa una estrategia de programación válida en entornos cliente/servidor?
- **a)** *Fat Client/Thin Server*.
- **b)** Programación en dos capas.
- **c)** Uso de servidores de objetos.
- **d)** Todos los anteriores.

3 ¿Cuál de las siguientes herramientas no permite la validación de la sintaxis de código?
- **a)** UltraEdit.
- **b)** Microsoft Visual Studio.
- **c)** NetBeans IDE.
- **d)** Ninguno los anteriores.

4 Señale la respuesta correcta con respecto a CGI:
- **a)** Es un lenguaje de programación del lado del servidor.
- **b)** Es un estándar que indica la forma en la que un servidor debe ejecutar un programa externo.
- **c)** Puede programarse en HTML.
- **d)** Es una herramienta de carácter genérico.

5 Señale la respuesta correcta con respecto a la integración de código con servidores web:
- **a)** Las peticiones a una aplicación web pueden hacerse siguiendo un formato “informal” (en lenguaje natural).
- **b)** Las peticiones de tipo POST se utilizan para recuperar información del servidor.
- **c)** El código de una aplicación web estática no requiere la ejecución de ningún programa externo por parte del servidor.
- **d)** Ninguna de las anteriores.