Programación en Internet: clientes web

Sergio Luján Mora

Prefacio

En pocos años, Internet ha invadido casi todos los aspectos de la vida. Podemos comunicarnos a través de Internet de distintas formas (correo electrónico, radio y televisión *online*, telefonía IP). Podemos comprar diversos productos en Internet (libros, discos, entradas de cine). Podemos conocer gente a través de Internet (*chat*, foros de discusión). Para que todo ello funcione, hacen falta profesionales especializados en "programación en Internet".

En este libro se trata una pequeña porción de la "programación en Internet": la programación de aplicaciones web. Las aplicaciones web se encuadran dentro de las arquitecturas cliente/servidor. En concreto, en este libro se estudia como programar la parte cliente, los clientes web. Las tecnologías que se contemplan son HTML y JavaScript. Existen otras tecnologías, como ActiveX o *applets*, pero no están estandarizadas como HTML y JavaScript.

Este libro se complementa con otro de próxima aparición que tratará la programación de aplicaciones web desde el lado del servidor. En él se mostrarán las tecnologías que se emplean para programar los servidores web: CGI, ASP, JSP, etc.

Evidentemente, existen programas que permiten crear páginas HTML sin tener ni idea de HTML. Pero igual que se puede conducir un coche sin tener ni idea de mecánica o pilotar un avión sin tener ni idea de aerodinámica o de motores a reacción, conviene conocer "lo que se tiene entre manos" para poder obtener el máximo partido. La programación de aplicaciones web no consiste sólo en crear páginas web muy bonitas, con muchas imágenes y colores; hay que validar la entrada del usuario, acceder a bases de datos, etc.

Este no es un libro sobre programación básica, donde se expliquen conceptos como variable, bucle de repetición, expresión lógica o recursividad. Es necesario poseer un nivel mínimo de programación para poder abordar los temas que tratan sobre JavaScript. Sin embargo, cualquiera que haya programado en

algún lenguaje no tendrá problemas en asimilarlo.

El libro está estructurado en seis capítulos. Los dos primeros capítulos son introductorios: se repasan las arquitecturas cliente/servidor en general y se presenta un tipo concreto, las aplicaciones web. El tercer capítulo está dedicado al lenguaje HTML. En el cuarto capítulo se explican los lenguajes de *script* en general y en el siguiente capítulo se trata un lenguaje concreto: JavaScript. El libro finaliza con el modelo de objetos de documento, que permite acceder a los elementos de una página web desde un lenguaje de *script*. Por último, existen tres apéndices donde se resumen las etiquetas de HTML y se explica como trabajar con los colores en HTML y como depurar errores de JavaScript.

Aunque no figuran como autores (y no tienen "ni idea" de HTML o JavaScript), sin la participación de mis padres este libro no existiría, ya que su apoyo a lo largo de bastantes años me ha permitido llegar a escribir este libro.

Me gustaría agradecer a Marisa la paciencia (¿infinita o ínfima?) que ha tenido todas las veces que he llegado tarde debido a este libro.

Por último, mando un saludo a mis antiguos compañeros del Laboratorio Multimedia (mmlab), donde me introduje en el mundo de Internet, y a mis actuales compañeros del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Alicante.

Alicante, 8 de octubre de 2001 *Sergio Luján Mora*

Índice general

Índice de cuadros

Índice de figuras

Índice de acrónimos

ASP *Active Server Pages*
Tecnología de Microsoft que permite crear páginas web dinámicas en el servidor. Se puede decir que las páginas **ASP** son similares a los programas **CGI**. Las páginas **ASP** suelen estar programados en *VBScript*, aunque también se pueden programar en otros lenguajes.

ASCII *American Standard Code for Information Interchange*
Código binario utilizado para representar letras, números, símbolos, etc. A cada carácter se le asigna un número del 0 al 127 (7 bits). Por ejemplo, el código **ASCII** para la A mayúscula es 65. Existen códigos **ASCII** extendidos de 256 caracteres (8 bits), que permiten representar caracteres no ingleses como las vocales acentuadas o la eñe. Los caracteres de la parte superior (128 a 255) de estos códigos **ASCII** extendidos varían de uno a otro. Por ejemplo, uno de los más extendidos es ISO Latin-1 (oficialmente ISO-8859-1).

CGI *Common Gateway Interface*
Estándar que permite el intercambio de información entre un servidor y un programa externo al servidor. Un programa **CGI** es un programa preparado para recibir y enviar datos desde y hacia un servidor web según este estándar. Normalmente se programan en *C* o en *Perl*, aunque se puede usar cualquier lenguaje de propósito general.

DHTML *Dynamic HTML*
Conjunto de extensiones a **HTML** que permiten modificar el contenido de una página web en el cliente sin necesidad de establecer una nueva comunicación con el servidor. Se basa en el uso de **DOM** para acceder al contenido de la página.

DLL *Dynamic Link Library*
Fichero que almacena funciones ejecutables o datos que pueden ser usados por una aplicación en Microsoft Windows. Una **DLL** puede ser usada por varios programas a la vez y se carga en tiempo de ejecución (no en tiempo de compilación).

DOM *Document Object Model*
Especificación que define como se puede acceder a los objetos de un documento **HTML** (ventanas, imágenes, formularios) a través de un lenguaje de *script*. Básicamente define un jerarquía de objetos. **DOM** se encuentra en proceso de estandarización por **W3C**. **DHTML** depende de **DOM** para cambiar dinámicamente el contenido de una página web. Desgraciadamente, los dos navegadores mayoritarios poseen distintos modelos de objetos.

ECMA *European Computer Manufacturers Association*
ECMA es una asociación internacional que establece estándares relacionados con sistemas de comunicación y de información.

GIF *Graphics Interchange Format*
Formato gráfico de mapas de bit desarrollado por CompuServe. Incorpora compresión de datos, transparencias y animaciones. Existen dos versiones de este estándar gráfico: 87a y 89a.

HTML *HyperText Markup Language*
Lenguaje compuesto de una serie de etiquetas o marcas que permiten definir el contenido y la apariencia de las páginas web. Aunque se basa en **SGML**, no se puede considerar que sea un subconjunto. Existen cientos de etiquetas con diferentes atributos. **W3C** se encarga de su estandarización. El futuro sustituto de **HTML** es **XHTML**.

HTTP *HyperText Transfer Protocol*
Es el protocolo que emplea la **WWW**. Define cómo se tienen que crear y enviar los mensajes y qué acciones debe tomar el servidor y el navegador en respuesta a un comando. Es un protocolo *stateless* (sin estado), porque cada comando se ejecuta independientemente de los anteriores o de los posteriores. Actualmente, la mayoría de los servidores soportan **HTTP** 1.1 (**RFC** 2616 de junio de 1999). Una de las principales ventajas de esta versión es que soporta conexiones persistentes: una vez que el navegador

se conecta al servidor, puede recibir múltiples ficheros a través de la misma conexión, lo que aumenta el rendimiento de la transmisión hasta en un 20 %.

ISAPI *Internet Server Application Program Interface*
Un API para el servidor Microsoft Internet Information Server. Permite programar aplicaciones web.

ISO *International Organization for Standards*
Organización fundada en 1946, cuyos miembros son las organizaciones nacionales de normalización (estandarización) correspondientes a los países miembros. Entre sus miembros se incluyen a la ANSI (Estados Unidos), BSI (Gran Bretaña), AFNOR (Francia), DIN (Alemania) y UNE (España).

JPEG *Joint Photographic Experts Group*
Formato gráfico de mapas de bit. Incorpora compresión de datos con pérdidas y permite trabajar con 24 bits de color.

JSP *Java Server Pages*
Tecnología de SUN MICROSYSTEMS que permite crear páginas dinámicas en el servidor. Equivale a la tecnología **ASP** de Microsoft. Se programan en *Java*.

MIME *Multipurpose Internet Mail Extensions*
Se usa en el correo electrónico desde 1992 para enviar y recibir ficheros de distinto tipo. Se puede consultar el estándar en **RFC** 1341, **RFC** 1521 y **RFC** 1522.

PNG *Portable Network Graphics*
Formato gráfico de mapas de bit similar a **GIF**. **W3C** ha decidido sustituir **GIF** por **PNG** debido a que el primero emplea un algoritmo que está patentado, mientras que **PNG** es totalmente gratuito. Tanto Microsoft Internet Explorer como Netscape Communicator aceptan este formato.

RFC *Request for Comments*
Medio de publicar propuestas sobre Internet. Cada **RFC** recibe un número. Algunos se convierten en un estándar de Internet.

RGB *Red Green Blue*
Notación de los colores en la que cada color se representa como una combinación de los tres colores básicos (primarios) rojo (*red*), verde (*green*) y azul (*blue*). Se trata de un modelo aditivo (se parte del negro). Mediante la combinación adecuada de los tres colores básicos se consigue todo el espectro de colores. Además de **RGB** existen otras formas de representar los colores. Otra de las más corrientes es CMYK (*cyan*, *magenta*, *yellow*, *black*), que se trata de un modelo sustractivo.

SGML *Standard Generalized Markup Language*
Lenguaje que permite organizar y etiquetar los distintos elementos que componen un documento. Se emplea para manejar grandes documentos que sufren constantes revisiones y se imprimen en distintos formato. Desarrollado y estandarizado por **ISO** en 1986.

TCP/IP *Transmission Control Protocol/Internet Protocol*
Familia de protocolos que se emplean en las comunicaciones de Internet.

URL *Universal Resource Locator*
También conocido como *Uniform Resource Locator*. Sistema de direccionamiento de máquinas y recursos en Internet. Es decir, se trata de una dirección que permite localizar cualquier máquina o documento que se encuentre accesible a través de Internet.

W3C *World Wide Web Consortium*
Consorcio internacional de compañías involucradas en el desarrollo de Internet y en especial de la **WWW**. Su propósito es desarrollar estándares y "poner orden" en Internet.

WWW *World Wide Web*
Sistema de servidores web conectados a Internet (no todos los ordenadores conectados a Internet forman parte de la **WWW**). Su protocolo de comunicación es **HTTP**, su lenguaje de creación de documentos **HTML** y sus sistema de direccionamiento de los recursos **URL**. Los navegadores web (*browsers*) permiten navegar por la web.

XHTML *Extensible HyperText Markup Language*
HTML escrito según las normas que marca **XML**. Por tanto, se trata de una aplicación concreta de **XML** y no tienen que confundirse entre sí.

XML *Extensible Markup Language*
Metalenguaje de etiquetado basado en **SGML**. Diseñado específicamente para la **WWW** por **W3C**. Permite que un usuario diseñe sus propias etiquetas, con sus atributos y las reglas de construcción de documentos (sintaxis).

Capítulo 1

Arquitecturas cliente/servidor

Las aplicaciones web son un tipo especial de aplicaciones cliente/servidor. Antes de aprender a programar aplicaciones web conviene conocer las características básicas de las arquitecturas cliente/servidor.

Índice General

1.1. Introducción

Cliente/servidor es una arquitectura de red[1] en la que cada ordenador o proceso en la red es **cliente** o **servidor**. Normalmente, los servidores son ordenadores potentes dedicados a gestionar unidades de disco (servidor de ficheros), impresoras (servidor de impresoras), tráfico de red (servidor de red), datos (servidor de bases de datos) o incluso aplicaciones (servidor de aplicaciones), mientras que los clientes son máquinas menos potentes y usan los recursos que ofrecen los servidores.

Esta arquitectura implica la existencia de una relación entre procesos que solicitan servicios (**clientes**) y procesos que responden a estos servicios (**servidores**). Estos dos tipos de procesos pueden ejecutarse en el mismo procesador o en distintos.

La arquitectura cliente/servidor implica la realización de aplicaciones distribuidas. La principal ventaja de esta arquitectura es que permite separar las funciones según su servicio, permitiendo situar cada función en la plataforma más adecuada para su ejecución. Además, también presenta las siguientes ventajas:

- Las redes de ordenadores permiten que múltiples procesadores puedan ejecutar partes distribuidas de una misma aplicación, logrando concurrencia de procesos.

- Existe la posibilidad de migrar aplicaciones de un procesador a otro con modificaciones mínimas en los programas.

- Se obtiene una escalabilidad de la aplicación. Permite la ampliación horizontal o vertical de las aplicaciones. La **escalabilidad horizontal** se refiere a la capacidad de añadir o suprimir estaciones de trabajo que hagan uso de la aplicación (clientes), sin que afecte sustancialmente al rendimiento general. La **escalabilidad vertical** permite la migración hacia servidores de mayor o menor capacidad y velocidad o de un tipo diferente.

[1] Otro tipo de arquitectura de red es *peer-to-peer* (entre pares o de igual a igual), en la que cada ordenador de la red posee responsabilidades equivalentes.

- Posibilita el acceso a los datos independientemente de donde se encuentre el usuario.

1.2. Separación de funciones

La arquitectura cliente/servidor nos permite la separación de funciones en tres niveles, tal como se muestra en la Figura 1.1:

- **Lógica de presentación**. La presentación de los datos es una función independiente del resto.
- **Lógica de negocio (o aplicación)**. Los flujos de trabajo pueden cambiarse según las necesidades existentes de un procesador a otro.
- **Lógica de datos**. La gestión de los datos debe ser independiente para poder ser distribuida según las necesidades de la empresa en cada momento.

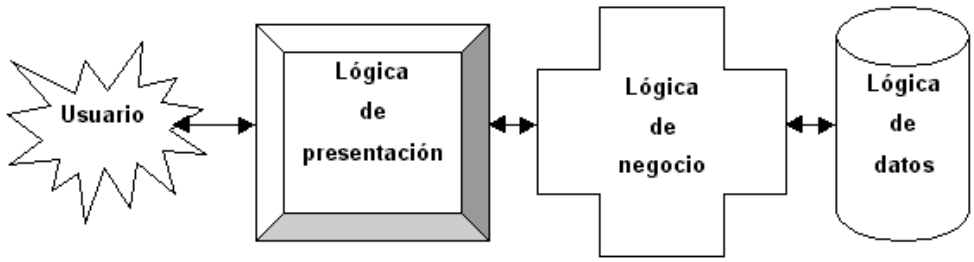

Figura 1.1: Separación de funciones

Si un sistema distribuido se diseña correctamente, los tres niveles anteriores pueden distribuirse y redistribuirse independientemente sin afectar al funcionamiento de la aplicación.

1.3. Modelos de distribución en aplicaciones cliente/-servidor

Según como se distribuyan las tres funciones básicas de una aplicación (presentación, negocio y datos) entre el cliente y el servidor, podemos contemplar

tres modelos: presentación distribuida, aplicación distribuida y datos distribuidos.

1.3.1. Presentación distribuida

El cliente sólo mantiene la presentación, el resto de la aplicación se ejecuta remotamente (Figura 1.2). La presentación distribuida, en su forma más simple, es una interfaz gráfica de usuario a la que se le pueden acoplar controles de validación de datos, para evitar la validación de los mismos en el servidor.

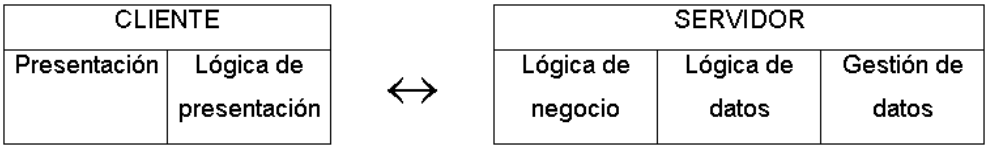

Figura 1.2: Presentación distribuida

1.3.2. Aplicación distribuida

Es el modelo que proporciona máxima flexibilidad, puesto que permite tanto a servidor como a cliente mantener la lógica de negocio realizando cada uno las funciones que le sean más propias, bien por organización, o bien por mejora en el rendimiento del sistema (Figura 1.3).

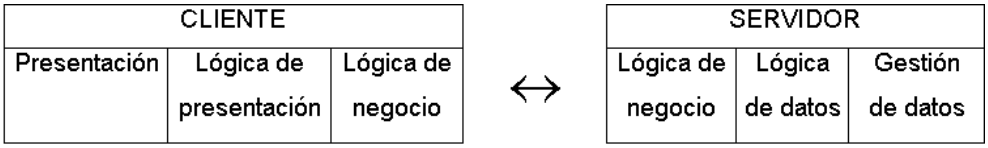

Figura 1.3: Aplicación distribuida

1.3.3. Datos distribuidos

Los datos son los que se distribuyen, por lo que la lógica de datos es lo que queda separada del resto de la aplicación (Figura 1.4). Se puede dar de dos formas: ficheros distribuidos o bases de datos distribuidas.

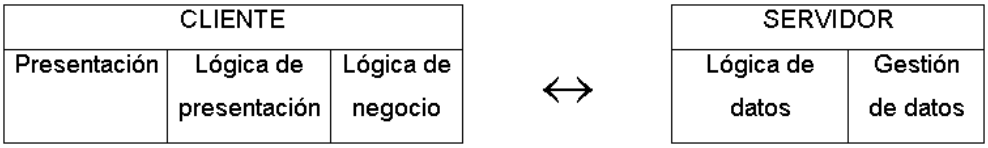

Figura 1.4: Datos distribuidos

1.4. Arquitecturas de dos y tres niveles

La diferencia entre las aplicaciones de dos y tres niveles estriba en la forma de distribución de la aplicación entre el cliente y el servidor.

Una arquitectura de dos niveles está basada en un sistema gestor de bases de datos donde el cliente mantiene la lógica de la presentación, negocio, y de acceso a los datos, y el servidor únicamente gestiona los datos. Suelen ser aplicaciones cerradas que supeditan la lógica de los procesos cliente al gestor de base de datos que se está usando.

En las arquitecturas de tres niveles, la lógica de presentación, la lógica de negocio y la lógica de datos están separadas, de tal forma que mientras la lógica de presentación se ejecutará normalmente en la estación cliente, la lógica de negocio y la de datos pueden estar repartidas entre distintos procesadores.

El objetivo de aumentar el número de niveles en una aplicación distribuida es lograr una mayor independencia entre un nivel y otro, lo que facilita la portabilidad en entornos heterogéneos.

1.5. Descripción de un sistema cliente/servidor

Un sistema cliente/servidor suele presentar las siguientes características:

1. Una combinación de la parte cliente (también llamada *front-end*) que interactúa con el usuario (hace de interfaz entre el usuario y el resto de la aplicación) y la parte servidor (o *back-end*) que interactúa con los recursos compartidos (bases de datos, impresoras, módems).

2. La parte cliente y servidor tienen diferentes necesidades de recursos a la hora de ejecutarse: velocidad de procesador, memoria, velocidad y capacidad de los discos duros, dispositivos de entrada/salida, etc.

3. El entorno suele ser heterogéneo y multivendedor. El hardware y sistema operativo del cliente y el servidor suelen diferir. El cliente y el servidor se suelen comunicar a través de unas API[2] y RPC[3] conocidas (por ejemplo, ODBC[4] para acceder a bases de datos).

4. Normalmente la parte cliente se implementa haciendo uso de una interfaz gráfica de usuario, que permite la introducción de datos a través de teclado, ratón, lápiz óptico, etc.

[2] *Application Program Interface*, interfaz de programación de aplicaciones.
[3] *Remote Procedure Call*, llamada a procedimiento remoto.
[4] *Open Database Connectivity*, conectividad de bases de datos abierta.