

La programación web

La programación de aplicaciones web impone una nueva forma de pensar y programar aplicaciones. En este capítulo veremos qué es la programación web y algunas de sus características clave.

Introducción al mundo web	16
Internet	16
Sobre HTML	17
Páginas estáticas	18
Páginas dinámicas	19
Programación web vs. tradicional	20
HTTP	20
Sesiones	22
Pedidos HTTP	23
Algo sobre seguridad	24
Programando para la Web	25
Resumen	25
Actividades	26

INTRODUCCIÓN AL MUNDO WEB

En los últimos años, Internet dejó de ser un mero divertimento para pasar a ser un medio fundamental de desarrollo de negocios. Hoy en día, podemos hacer mucho más que visitar páginas web y chatear. Estamos acostumbrados a realizar muchas de nuestras tareas cotidianas vía Internet: pagar las cuentas, alquilar una película en el video club, revisar nuestros mails, reservar un hotel y pasajes para nuestras vacaciones, y muchas cosas más. Todo esto, independientemente del lugar donde estemos y del horario. En este contexto, estar en Internet es condición *sine qua non* para cualquier empresa.

En un principio, parecería suficiente con tener una mera página con información de contacto, en la que se muestren productos y servicios, pero enseguida se vuelve imprescindible proveer nuevos servicios a los potenciales clientes a través de la red de redes, para no perder competitividad en el mercado. Un banco ya no puede dejar de ofrecer home-banking, una línea aérea que no tenga página en Internet donde se puedan consultar los vuelos actualizados y reservar o comprar pasajes pierde mucho mercado. Y así con cualquier área de negocio en que pensemos.

Programar este tipo de servicios puede parecer muy complicado, y si bien hay algunas aplicaciones críticas, como por ejemplo, las bancarias —en donde la transferencia de efectivo no puede dar lugar a errores de sistemas y se requiere mucha inversión en materia de seguridad—, veremos que normalmente desarrollar una aplicación web segura no es más complicado que programar cualquier aplicación **stand-alone**. De hecho, dados los reducidos requerimientos y limitaciones de este tipo de desarrollos (que se verán más adelante) y la gran cantidad de software open source que hay disponible para ser usado gratuitamente, en muchos casos suele ser más simple hacer una aplicación web que una de escritorio.

Internet

Para poder meternos de lleno en la programación web, primero es fundamental entender cómo funciona, a grandes rasgos, Internet. Cuando nosotros abrimos

CGI

El primer sistema de páginas dinámicas fue desarrollado en 1993 y se llamó **CGI** (*Common Gateway Interface*). Estaba muy ligado al servidor de páginas web, y se encontraba escrito en **lenguaje C**. Eran sistemas muy complejos; sin embargo, este tipo de programación fue (e incluso todavía es) usado por muchos programadores por su rapidez y seguridad.

un navegador y lo apuntamos hacia la página web que deseamos ver, por ejemplo, **www.google.com**, en el fondo se está generando una comunicación entre dos programas: un **cliente** y un **servidor**.

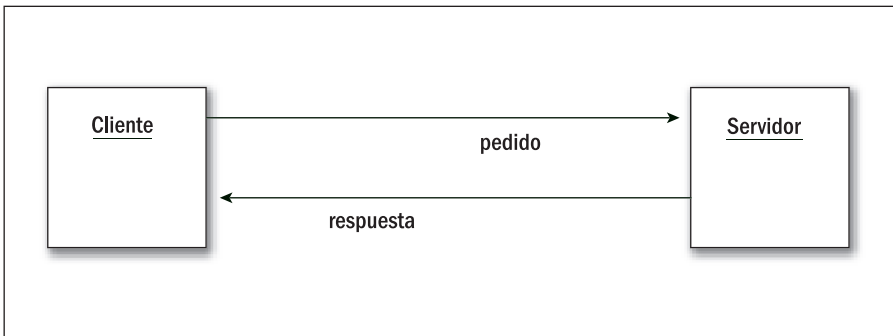


Figura 1. Al navegar por Internet básicamente estamos realizando una comunicación entre dos programas.

El cliente es nuestro navegador (Internet Explorer, Firefox, etc.), que se comunica con el servidor. En este caso, el servidor es un **programa** que está constantemente escuchando **peticiones** de clientes y devuelve para cada pedido una **respuesta** acorde. La respuesta es tomada por nuestro navegador y se muestra en pantalla, y de esta forma navegamos, de pedido en pedido, respuesta tras respuesta, continuamente.

Sobre HTML

En este libro asumiremos que las páginas que devuelve el servidor son siempre páginas HTML. Si bien esto ocurre en la mayoría de los casos en los sitios de Internet, también hay muchos sitios que trabajan con otras tecnologías, de las cuales la más común es **Flash**.

HTML significa *Hypertext Markup Language*. Si lo traducimos al castellano: Lenguaje de **M**arcado de **H**ipertexto. Nos alcanza con saber que HTML es un lenguaje que le especifica ciertos atributos a su contenido, y los navegadores web



HTML

HTML es un lenguaje en constante evolución. En la página del W3C (World Wide Web Consortium) hay muchísima información sobre este lenguaje y sus variantes. Excelente lugar para mantenerse actualizado (www.w3.org/MarkUp).

saben muy bien cómo mostrar este contenido formateado según esos atributos. Por ejemplo, un documento HTML (o página HTML) puede especificar que en cierto lugar del documento haya que insertar una imagen, o que cierto texto dentro de su contenido vaya en negrita o con determinada tipografía. O sea, no difiere mucho de lo que podemos hacer con un documento Word, de hecho, cualquier procesador de texto permite exportar o guardar documentos como archivos de tipo HTML. La diferencia entre ambos documentos radica en el contexto en que van a utilizarse y a que HTML es un documento escrito enteramente en texto plano. Esto significa que también podemos crear documentos HTML en cualquier editor de texto sencillo. Ahora que sabemos un poco sobre lo que es HTML, vamos a desglosar su significado.

- **Hypertext:** se dice que un documento HTML contiene **hipertexto** en el sentido que un documento HTML puede referenciar a otro documento HTML. Cada vez que un documento referencia a otro, se dice que está conectado a ese otro documento mediante un enlace (o link).
- **Markup:** un documento HTML define sus secciones mediante **marcas** (etiquetas o tags) en su contenido.
- **Language:** ¡es un lenguaje! Los navegadores saben hablar este lenguaje y cuando un servidor les provee una página HTML, saben cómo mostrarla en pantalla.

Páginas estáticas

En muchos casos, el servidor es un programa que simplemente toma el pedido que recibe y devuelve una página (compuesta por uno o más archivos HTML, imágenes, animaciones, etc.) que está guardada en algún lugar del disco. En estos casos decimos que son páginas **estáticas**.

A no ser que el administrador del sitio actualice su contenido, al ingresar en la página web siempre obtendremos el mismo resultado, no importa cuándo ingresemos ni desde dónde. Para la gran mayoría de sistemas de negocios, este tipo de servicio no es muy útil, ya que se suele necesitar que se devuelva una página con contenido **dinámico**, que cambie según quién pidió la página, según la fecha, etc.



FLASH

Flash es una tecnología desarrollada por Macromedia, muy popular hoy día. Sirve para crear animaciones interactivas de todo tipo y es muy aplicable para hacer sitios web. Más información, en www.macromedia.com/software/flash/flashpro.

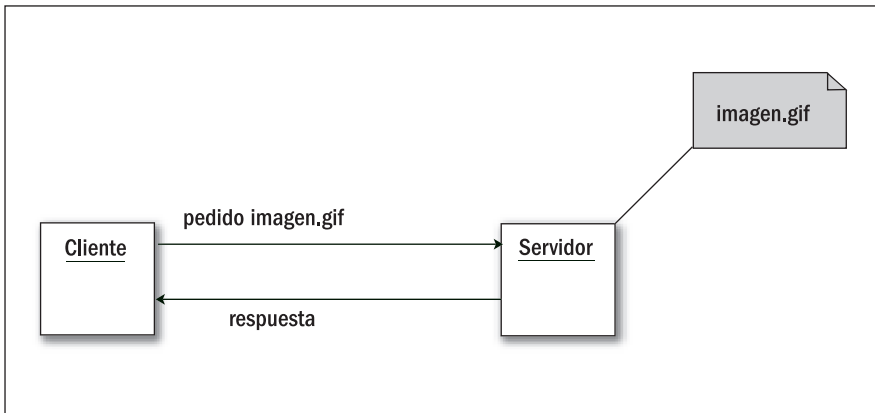


Figura 2. El servidor de páginas estáticas siempre devuelve el mismo recurso para el mismo pedido.

Páginas dinámicas

Pensemos, por ejemplo, en el sistema de un banco. Si ingreso en el sitio del banco y quiero consultar mi saldo, debería devolverme mi saldo actual al momento exacto de pedirlo. Si el sitio web sólo provee contenido estático, entonces deberían tener una página guardada en el disco para cada saldo posible. Esto, obviamente, es impracticable. Y tampoco es viable que haya una persona que esté actualizando las páginas de saldos de todos los usuarios a medida que van realizando operaciones sobre sus cuentas. En esos casos, el servidor efectúa operaciones (yendo a buscar datos a una base de datos, consultando con otro servidor o accediendo a otro tipo de servicios de negocios) y devuelve una página **dinámica**; esto es, una página que no está guardada en ningún lugar dentro del servidor, sino que fue creada en el momento para quien la pidió. El servidor accede a los datos variables, en este caso, el saldo del usuario, luego construye en memoria la página con este dato y se la devuelve al cliente, que la muestra en pantalla.

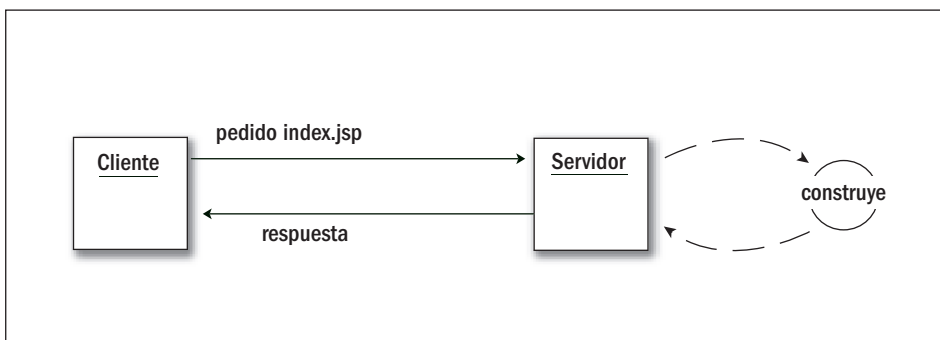


Figura 3. El servidor web dinámico construye una respuesta distinta para cada pedido.

La programación web entonces, consiste en escribir programas que, dada una petición web realizada por un cliente (un navegador), procesen el pedido y generen y devuelvan un resultado.

Como en todo proceso de desarrollo de software, hay mucho de reutilización. Podremos ver que, por ejemplo, la parte de recibir el pedido y enviar la respuesta por lo general no cambia, solamente cambia la lógica de negocios de cada pedido y el contenido de la respuesta generada. En este libro utilizaremos herramientas ya desarrolladas de código abierto que nos ayudarán en la creación de aplicaciones web.

Programación web vs. tradicional

La programación web es un nuevo paradigma, dado que impone ciertas restricciones que pueden resultar confusas en un principio. Estas restricciones se basan en que Internet (al menos hasta ahora) trabaja sobre el protocolo **HTTP** (*HyperText Transfer Protocol*). Cada vez que escribimos en un navegador **http://direccion.web.com**, estamos indicándole explícitamente que se conecte usando dicho protocolo. Si no lo escribimos, por lo general el navegador se encarga de rellenarlo automáticamente, aunque los navegadores a menudo también implementan otros tipos de comunicaciones entre ellos y un servidor.

Por ejemplo, **Internet Explorer** y **Firefox** soportan comunicaciones de tipo **FTP** (*File Transfer Protocol*), que es un protocolo para la transferencia de archivos. Para acceder a un determinado recurso mediante ese otro protocolo, basta con escribir la dirección en el navegador (por ejemplo, **ftp://ftp.uba.ar**) y éste sabrá automáticamente que debe establecer una conexión con el servidor mediante el protocolo FTP.

HTTP

El protocolo **HTTP**, como su nombre lo indica, fue diseñado para transferir documentos de hipertexto (documentos HTML). En sus orígenes, cuando Internet era **ARPANET** y era muchísimo más pequeña de lo que es ahora, y cuando la velocidad de conexión era enormemente inferior, texto plano era lo único que se

HTTP

La actual versión del protocolo HTTP es la 1.1. En www.w3.org/Protocols/rfc2616/rfc2616.html está la especificación completa. Como el **World Wide Web Consortium** considera que la versión 1.1 supera las deficiencias de la versión 1.0, no está trabajando en nuevas versiones, aunque sí están trabajando en un protocolo relacionado que integre HTTP con el nuevo protocolo **XML Protocol**.

transfería por ella. A nadie se le ocurría siquiera publicar un documento con imágenes o sonidos. A medida que avanzó el tiempo y la red se volvió más veloz, estas necesidades se hicieron evidentes y el protocolo fue mejorado para poder transferir cualquier tipo de datos, incluyendo voz, imágenes, video, etc.

HTTP tiene varias otras características, pero la que más nos va a interesar es que es un protocolo que no guarda el estado (**stateless**). Esto significa que no se mantiene una conexión constantemente entre el cliente y el servidor, sino que el cliente manda el pedido y corta la conexión, sin guardar información sobre pedidos anteriores. De esta forma, el servidor trata cada pedido en forma independiente del anterior, simplemente porque no puede saber si el pedido proviene del mismo cliente, aunque hayan ocurrido muy cerca en el tiempo. Esto puede sonar muy confuso. Una analogía que aclara bastante las cosas es la siguiente: el servidor es una persona sentada detrás de una puerta. Esta persona recibe papelitos con preguntas por debajo de la puerta, los mira, escribe en ellos una respuesta y los envía de vuelta, y esa es toda la comunicación que tiene con el mundo exterior. Esta persona no sabe quién mandó cada papelito, sólo los recibe y responde. Quizá del otro lado de la puerta hay una sola persona que es la misma siempre y manda diferentes preguntas, o hay cientos de personas, cada cual con sus inquietudes esperando ser respondidas; pero este humilde servidor no lo sabe, ni puede saberlo. Esto hace complicado mantener un hilo de conversación entre los clientes (quienes mandan los papelitos) y el servidor (quien los recibe y responde). Por ejemplo, si una persona hace una pregunta y al obtener la respuesta quiere hacer otra pregunta relacionada, tiene que formular la pregunta sabiendo que quien responde la tomará como si fuera una pregunta totalmente nueva.

Esta problemática también hace imposible saber si alguien que acaba de mandar un papelito sigue estando tras la puerta o se fue hace rato. Un sitio web que se comunique con el cliente únicamente mediante documentos HTML a través del protocolo HTTP no tiene forma de saber (sin acudir a otras técnicas de programación como applets, Flash, componentes ActiveX, etc.) si el usuario sigue navegando su sitio o si se fue a otra página o cerró el navegador.

III HTTPS

El protocolo **HTTPS** fue inventado por Netscape y funciona en la capa de presentación del modelo OSI/ISO. Esto hace que sea fácil aplicarlo a protocolos del nivel de aplicación (como HTTP o FTP) sin hacer muchos cambios. La encriptación se basa en un algoritmo de clave pública y privada que encripta información entre dos partes sin que necesiten acordar previamente la clave que utilizarán.

Sesiones

Esta restricción del protocolo HTTP es un impedimento muy grande. Prácticamente toda aplicación necesita superar este inconveniente y poder mantener un registro de la conversación mantenida entre el usuario y el servidor. Para ello se define el concepto de **sesión**, que se maneja de la siguiente forma:

- El cliente realiza un pedido al servidor.
- El servidor responde el pedido y, a su vez, le devuelve un identificador al cliente.
- El cliente deberá, en los sucesivos pedidos, incluir este identificador en cada pedido que realice al servidor.
- El servidor, al reconocer el identificador, puede mantener un estado de pedidos de un mismo cliente.

De esta forma se puede solucionar el problema que acarrea la naturaleza **sin estado** del protocolo HTTP.

Hay dos formas de lograr que el navegador incluya este identificador de sesión en cada pedido al servidor. La primera es mediante el uso de **cookies**. Las cookies (¡galletitas!) son pequeños archivos con información que el servidor envía al navegador para que éste guarde, y que el navegador vuelve a enviar en cada pedido que realiza al servidor. De este modo, el navegador puede guardar información específica sobre el sitio que se está visitando, como puede ser el identificador de sesión, las preferencias del usuario (por ejemplo, el idioma, los colores con que prefiere visualizar la página), etc.

La segunda forma, llamada **URL rewriting**, hace que el navegador sobrescriba todos los enlaces que vuelven al servidor agregándoles como parámetro el identificador de sesión. Esto es, cada acción que el usuario pueda hacer desde el navegador que vuelva al servidor (y que no sea un enlace externo a otro sitio), va a tener agregado un parámetro con el identificador, de esta forma, al hacer clic sobre cualquiera de los links que tenga la página, estará pasando a su vez el identificador de sesión, y así el servidor podrá identificarlo.



IDEMPOTENCIA

La especificación del protocolo HTTP dice que los métodos **GET**, **HEAD**, **PUT** y **DELETE** deben ser **idempotentes**. Esto significa que pedidos de estos tipos no deben tener efectos secundarios. Más formalmente: para una sucesión de pedidos idénticos, el resultado debe ser siempre el mismo.

Es importante destacar que todo tipo de información de estado se guarda en el servidor. Es éste quien mantiene los datos asociados con la **sesión** del usuario. Por ejemplo, en el clásico sistema de compra online donde hay un carrito de compras virtual, cada vez que un usuario agrega un producto, se agrega a la lista que ya contiene el carrito del usuario, pero este carrito y su información residen en el servidor, que los mantiene asociados con el identificador de sesión del cliente; el cliente simplemente manda pedidos y se le muestran resultados, sin saber lo que ocurre del otro lado.

Pedidos HTTP

Hemos visto que los clientes realizan pedidos mediante el protocolo HTTP a servidores web. Lo que no vimos hasta ahora es que los pedidos HTTP pueden ser de varios tipos (también son llamados **métodos**). La **Tabla 1** muestra los diferentes tipos de pedidos que existen actualmente para el protocolo HTTP versión 1.1.

TIPO DE PEDIDO	DESCRIPCIÓN
OPTIONS	Se usa para preguntarle al servidor las diferentes formas de comunicación que soporta.
GET	Pide un recurso al servidor.
HEAD	Igual que el GET, pero el servidor sólo devuelve el encabezado de lo pedido.
POST	Método que se usa para enviar información al servidor.
PUT	Usado para enviar recursos al servidor.
DELETE	Borra recursos del servidor.
TRACE	Se usa para pedir un rastreo del pedido.
CONNECT	Método reservado.

Tabla 1. Los diferentes métodos que define el protocolo HTTP.

Estos ocho métodos definidos en la especificación del protocolo permiten establecer conexiones muy avanzadas entre clientes y servidores, pero que no son usadas por lo general por las aplicaciones web estándar. El método **TRACE**, por ejemplo, se usa para testear que el servidor esté recibiendo los datos correctamente y para depurar las conexiones. La mayoría de los servidores en producción directamente tienen deshabilitados muchos de estos métodos o están asociados con algún

III COOKIES

Ha habido y hay todavía bastante controversia sobre las cookies. Hay gente que las defiende y gente que dice que son una invasión a la privacidad, dado que el servidor guarda información en nuestra PC y la puede usar para recabar datos sin nuestro consentimiento. Por eso, la mayoría de los navegadores modernos permiten definir si queremos aceptar cookies o no y varias opciones más.

tipo de directiva de seguridad que impide que cualquier usuario los ejecute. Caso contrario, cualquier usuario desde algún lugar remoto del planeta podría ejecutar un **DELETE** de una página y la borraría del servidor. En este libro vamos a referirnos únicamente a pedidos de los tipos **POST** y **GET**. Si bien a priori **POST** y **GET** parecen totalmente distintos entre sí, vamos a ver que pueden usarse con los mismos propósitos, aunque hay casos en los que se evidencia que uno es más idóneo que el otro. La especificación recomienda distintos usos para cada uno de ellos. Dice que el método **GET** debería usarse solamente para obtener datos del servidor y que **POST** debe usarse para enviar información al servidor, como ser una orden de compra o una actualización de un dato. Así y todo, podemos usar **GET** para enviar información al servidor (con ciertas limitaciones) y **POST** para obtenerla.

Algo sobre seguridad

El tema de seguridad en aplicaciones web excede ampliamente los contenidos de este libro. Aunque es una cuestión muy tratada y sobre la que se ha escrito muchísimo, vamos a dar un pequeño pantallazo del tema, de su problemática y de algunas formas de mantenerlo bajo control.

Los sistemas web implementan numerosos procesos en los que deben ofrecer seguridad. Más adelante veremos una aplicación simple pero poderosa que se utiliza para brindar seguridad en lo referido a autenticaciones y autorizaciones.

Autenticación es el proceso que se encarga de verificar que un usuario es realmente quien dice que es. Es común en muchos sitios web. Cada vez que nos piden que ingresemos nuestro nombre de usuario y contraseña, básicamente nos están pidiendo que demostremos que somos el usuario dueño de la contraseña, y para esa verificación la ingresamos.

Autorización consiste en verificar que un usuario dado (que suponemos ya se ha autenticado) tenga permisos para efectuar determinada operación. Por ejemplo, en determinado sistema sólo el usuario administrador podrá borrar información, de manera que cuando el servidor identifique que se está pidiendo borrar algo, primero deberá verificar que el usuario que está pidiendo el borrado sea un usuario de tipo administrador, y en ese caso autorizará la acción.

Sin embargo, el principal problema de seguridad que plantean las aplicaciones web (al igual que toda aplicación distribuida) es el tema del transporte de datos.

El protocolo HTTP, como dijimos, transporta información entre computadoras, y con altísima probabilidad esta información viajará a través de muchos nodos por Internet hasta llegar al servidor al cual se dirige. Esto trae muchos proble-

mas de seguridad, ya que alguien puede, con relativa facilidad, tener acceso a la comunicación que se establece entre cliente y servidor y puede leer los datos que se transmiten, y estos datos pueden incluir números de tarjetas de crédito, claves, información confidencial, etc.

La solución que se usa hoy en día (aunque en materia de seguridad informática nada es ciento por ciento seguro) es un protocolo llamado **HTTPS**. Este protocolo lo que especifica es que las comunicaciones se siguen haciendo igual que con HTTP, pero antes de enviar la información el cliente, la **encripta**. El servidor la recibe, la **desencripta** y luego la procesa. Así, si alguien logra interceptar la comunicación, es casi imposible que pueda descifrar su contenido.

Programando para la Web

Ahora que hemos visto de qué se trata la programación web, con sus limitaciones y modelos desde un punto de vista teórico, vamos a darle un vistazo a la parte práctica.

RESUMEN

Hemos visto en este capítulo introductorio las oportunidades que Internet brinda al permitir ofrecer sitios web dinámicos que provean servicios de valor para los usuarios. Describimos también sus limitaciones y lo que se hace para intentar subsanarlas. Nótese que hemos analizado la programación web independientemente del lenguaje de programación que vayamos a utilizar para implementar nuestras aplicaciones.



TEST DE AUTOEVALUACIÓN

- 1 ¿Cuáles son los dos actores fundamentales en la navegación por Internet?

- 2 ¿En qué se diferencia un servidor de páginas estáticas con uno de páginas dinámicas?

- 3 ¿Qué característica del protocolo HTTP es la que más limita la programación web?

- 4 ¿Qué solución se aplica a este problema?

- 5 ¿De qué formas el navegador puede enviar al servidor el identificador de sesión?

- 6 Usando telnet y simulando ser un navegador web, ingrese a su diario online favorito y obtenga la página principal.

- 7 ¿Qué es una cookie?

- 8 ¿Por qué la mayoría de los servidores web tienen deshabilitado el método HTTP DELETE?

- 9 ¿Cuál es la diferencia entre autenticación y autorización?

- 10 ¿Por qué es necesario encriptar la información confidencial cuando es enviada por Internet?

EJERCICIOS PRÁCTICOS

- ✓ Conéctese usando un programa tipo Telnet a un servidor web (port 80).

- ✓ Lea la especificación de http.

- ✓ Use su navegador para observar las cookies alojadas en él.
