|  |  |
| --- | --- |
|  | 🞂**Protocolo HTTP**  **Hypertext Transfer Protocol** |
|  |  |
|  | Teleinformática y Comunicaciones  Trabajo Práctico de Investigación  Integrantes del grupo |
|  |  |
|  |  |
|  |  |
|  | * Gonzalez, Diego |
|  |  |
|  | * Gonzalez, Javier |
|  |  |
|  | * Hegykozi, Hernan |
|  |  |
|  | * Iturriaga, Ariel |

Fecha: 14 de Noviembre de 2012



## Índice

¿Cómo funciona HTTPS? 22

¿Cómo se resuelve el problema de seguridad de HTTP? 21

¿Qué es un tipo MIME? 15, 18

¿Qué problemas soluciona HTTPS? 21

¿SSL, cómo funciona? 23

15 preguntas con respuestas 25

1xx (Respuesta provisional) 10, 26

2xx (Correcto) 10

3xx (Redirigido) 11

4xx (Error de solicitud) 12, 26

5xx (Error del servidor) 14, 26

Algunas de las Ventajas de SOAP 19

Anatomía de un mensaje de SOAP 18

Bibliografía 28

Códigos de estado de HTTP 10

Conexiones 4

Ejemplo de un diálogo HTTP 5

Encabezados de mensajes 7

HTTP 4

Las propiedades más importantes 4

Mapa Conceptual Mental 29

Métodos 6

Principales características 4

Resumen 3

Protocolo HTTP

Hypertext Transfer Protocol

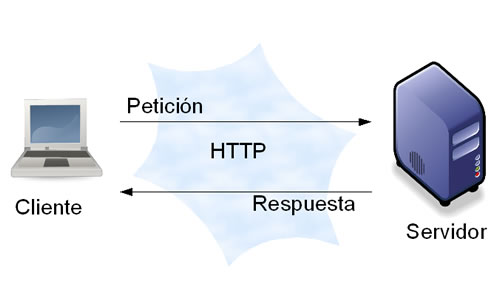
#### Resumen

Desde 1990, el protocolo HTTP (Protocolo de transferencia de hipertexto) es el protocolo más utilizado en Internet. La versión 0.9 solo tenía la finalidad de transferir los datos a través de Internet (en particular páginas Web escritas en HTML). La versión 1.0 del protocolo permite la transferencia de mensajes con encabezados que describen el contenido de los mensajes mediante la codificación MIME (extensiones multipropósito de correo en Internet).

Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión TCP/IP, y funciona de la misma forma que el resto de los servicios comunes de los entornos UNIX: un proceso servidor escucha en un puerto de comunicaciones TCP (por defecto, el 80), y espera las solicitudes de conexión de los clientes Web. Una vez que se establece la conexión, el protocolo TCP se encarga de mantener la comunicación y garantizar un intercambio de datos libre de errores.

El propósito del protocolo HTTP es permitir la transferencia de archivos (principalmente en formato HTML), entre un navegador (el cliente) y un servidor web (denominado entre otros, httpd en quipos UNIX) localizado mediante una cadena de caracteres denominada dirección URL (Uniform Resource Locator). (**Ver Figura 1**).

* **Figura 1**



Protocolo HTTP

Hypertext Transfer Protocol

## Principales características

Las principales características del protocolo HTTP son:

1. Toda la comunicación entre los clientes y servidores se realiza a partir de caracteres de 8 bits. De esta forma, se puede transmitir cualquier tipo de documento: texto, binario, etc., respetando su formato original.
2. Permite la transferencia de objetos multimedia. El contenido de cada objeto intercambiado está identificado por su clasificación MIME.
3. Existen tres verbos básicos (hay más, pero por lo general no se utilizan) que un cliente puede utilizar para dialogar con el servidor: GET, para recoger un objeto, POST, para enviar información al servidor y HEAD, para solicitar las características de un objeto (por ejemplo, la fecha de modificación de un documento HTML).
4. Cada operación HTTP implica una conexión con el servidor, que es liberada al término de la misma. Es decir, en una operación se puede recoger un único objeto.
5. No mantiene estado. Cada petición de un cliente a un servidor no es influida por las transacciones anteriores. El servidor trata cada petición como una operación totalmente independiente del resto.
6. Cada objeto al que se aplican los verbos del protocolo está identificado a través de la información de situación del final de la URL.

## HTTP

Especifica cuales mensajes pueden enviar los clientes a los servidores y qué respuestas obtienen. Cada interacción consiste en una solicitud ASCII, seguida por una respuesta tipo MIME del RFC 822. Todos los clientes y servidores deben obedecer este protocolo. Se define en el RFC 2616.

Las propiedades más importantes son:

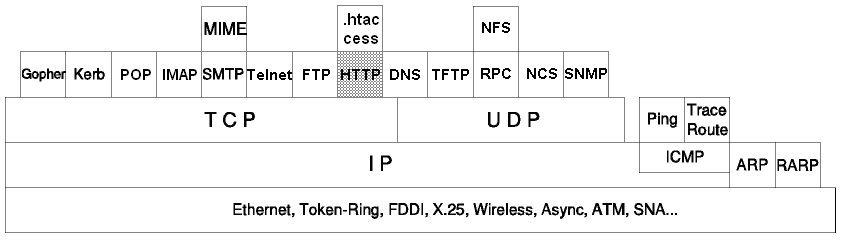
1. Conexiones.
2. Métodos.
3. Encabezados de mensajes.

## Conexiones

La forma común en que un navegador contacta a un servidor es estableciendo una conexión TCP con el puerto 80 de la máquina del servidor, El valor de utilizar TCP es que ni los navegadores ni los servidores tienen que preocuparse por los mensajes largos, perdidos o duplicados, ni por las confirmaciones de recepción. TCP maneja estos aspectos en su implementación (**Ver Figura 2**).

En HTTP 1.0, una vez que se establecía la conexión, se enviaba una solicitud y se obtenía una respuesta. Después se liberaba dicha conexión. Este método es adecuado si una página Web consiste por completo en texto HTML. Pero, todos sabemos que actualmente no es así. Una página Web actual contiene gran cantidad de iconos, imágenes, entre otras cosas, por lo que establecer una conexión TCP para transportar un solo icono es un método muy costoso.

* **Figura 2**



Debido a lo citado se diseñó HTTP 1.1, que soporta conexiones persistentes. Con ellas, es posible establecer una conexión TCP, enviar una solicitud y obtener una respuesta, y después enviar solicitudes adicionales y obtener respuestas adicionales. Al repartir la configuración y terminación de sesiones de TCP en múltiples solicitudes, la sobrecarga resultante de TCP es mucho menor por solicitud.

## Ejemplo de un diálogo HTTP cuando se quiere establecer una conexión

Para obtener un recurso con el [URL](http://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol#URL) <http://www.example.com/index.html>,

1. Se abre una conexión al host [www.example.com](http://www.example.com), puerto 80 que es el puerto por defecto para HTTP.
2. Se envía un mensaje en el estilo siguiente:

GET /index.html HTTP/1.1

Host: [www.example.com](http://www.example.com)

User-Agent: nombre-cliente

[Línea en blanco]

La respuesta del servidor está formada por encabezados seguidos del recurso solicitado, en el caso de una página web:

HTTP/1.1 200 OK

Date: Fri, 31 Dec 2003 23:59:59 GMT

Content-Type: text/html

Content-Length: 1221

<html>

<body>

<h1>Página principal de tuHost</h1>

(Contenido)

.

.

.

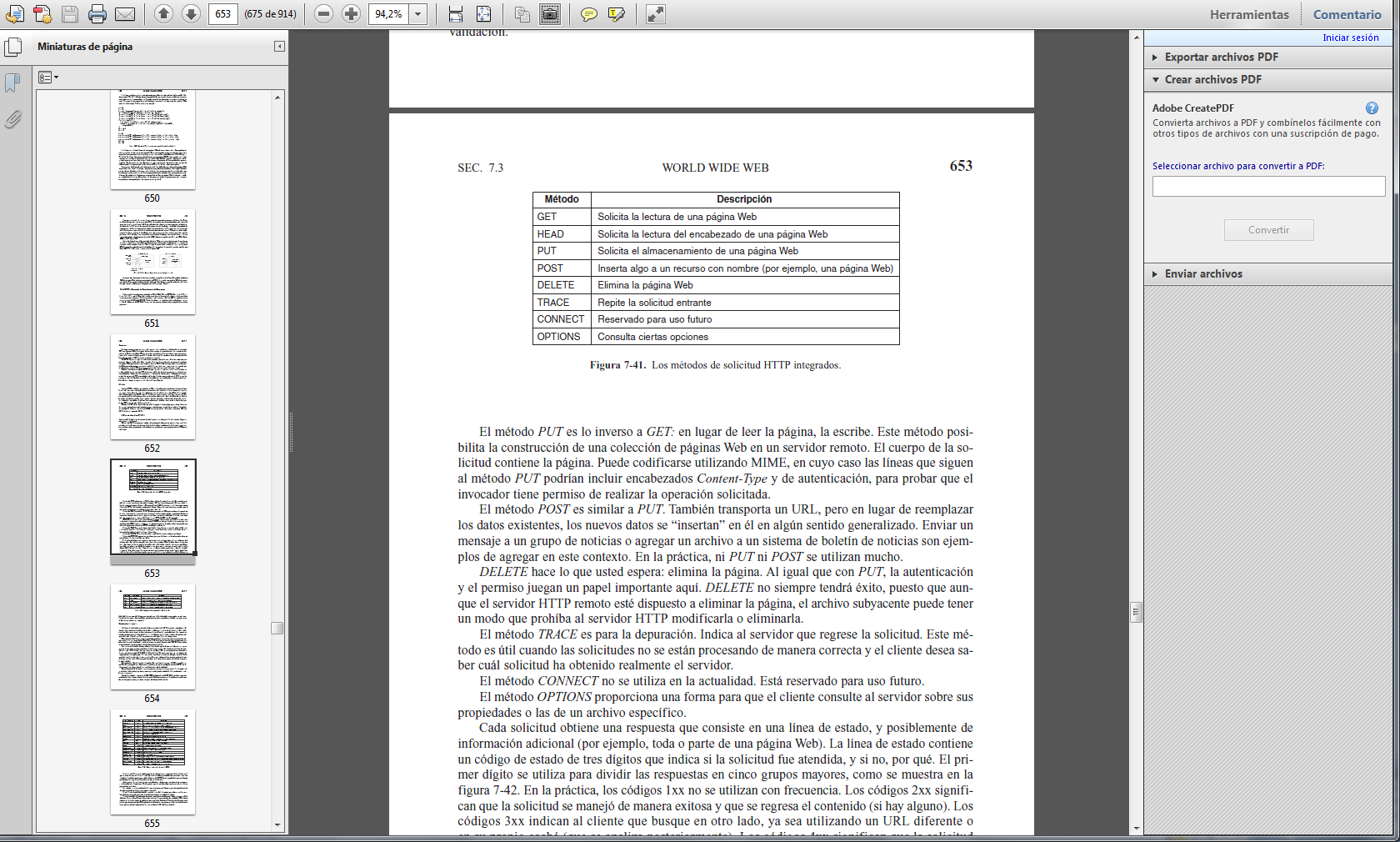
</body>

</html>

## Métodos

Aunque HTTP se diseñó para utilizarlo en Web, intencionalmente se ha hecho con miras a las aplicaciones orientadas a objetos futuras. Por ello, se soportan otras operaciones, llamadas métodos, Esta generalidad es lo que permite que SOAP (Simple Object Access Protocol) exista. Cada solicitud consiste en una o más líneas de texto ASCII, y la primera palabra de la primera línea es el nombre del método solicitado. Los nombres son sensibles a mayúsculas y minúsculas, por lo que GET es un método valido y get no lo es. Lista de métodos Ver figura 3.

* **Figura 3**



Veamos brevemente que significan cada uno de los métodos.

El método *GET* solicita al servidor que envíe la página. La página está codificada de forma adecuada en MIME. La mayoría de las solicitudes a servidores Web son GETs. La forma de GET común es:

GET nombredearchivo HTTP/1.1

El método *HEAD* simplemente solicita el encabezado del mensaje, sin la página real. Este método puede utilizarse para obtener la fecha de modificación de la página, para colectar información para propósitos de indización o para validar un URL.

El método *PUT* es lo inverso a GET: en lugar de leer la página, la escribe. Este método posibilita la construcción de una colección de páginas Web en un servidor remoto. El cuerpo de la solicitud contiene la página.

El método *POST* es similar a PUT. También transporta un URL, pero en lugar de reemplazar los datos existentes, los nuevos datos se insertan en él. Enviar un mensaje a un grupo de noticias o agregar un archivo a un sistema de boletín de noticias son ejemplos de agregar en este contexto.

*DELETE* justamente hace lo que estás pensando: elimina la página. DELETE no siempre tendrá éxito, puesto que aunque el servidor HTTP remota esté dispuesto a eliminar la página, el archivo puede tener un modo que prohíba al servidor HTTP modificarla o eliminarla.

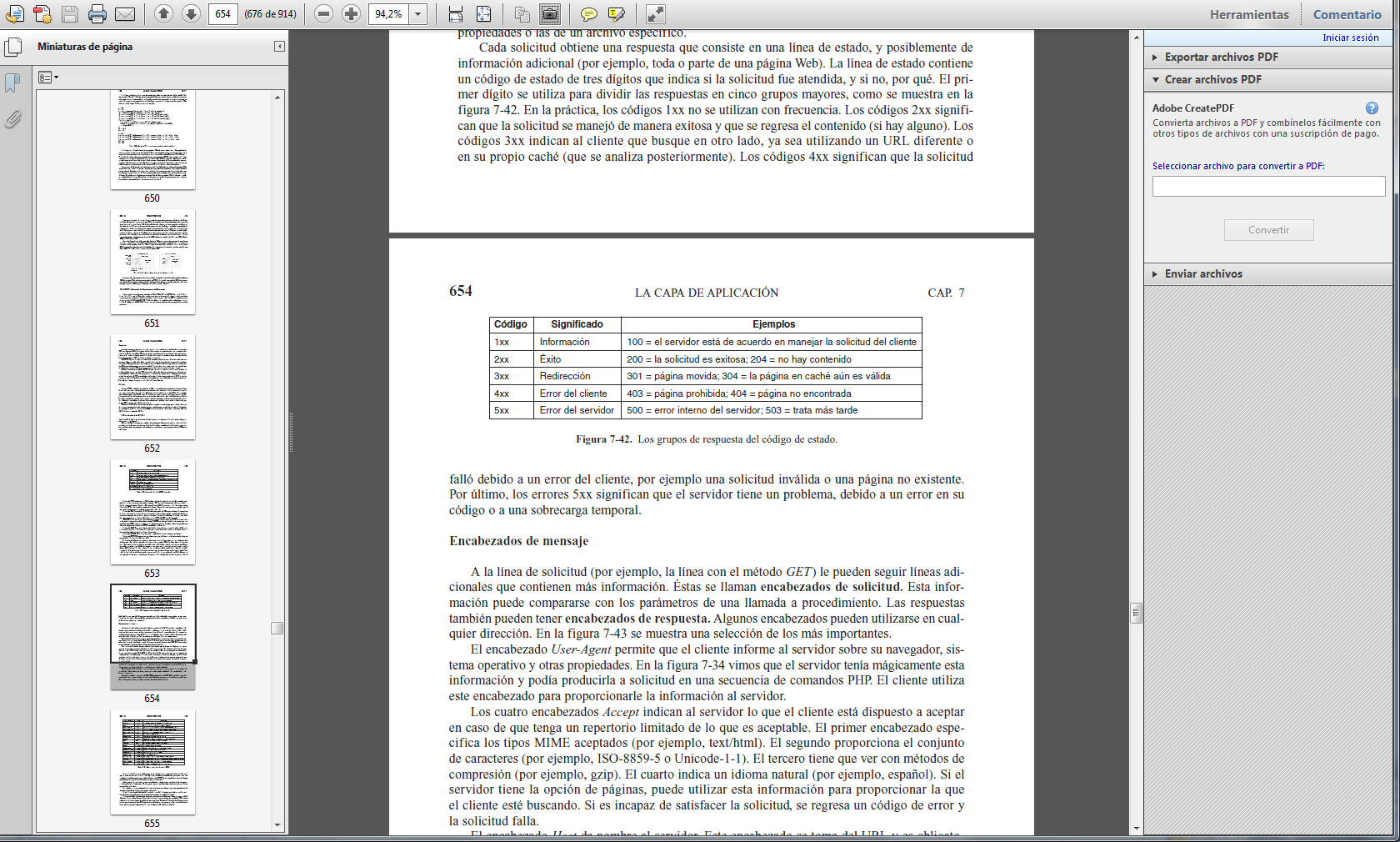
El método *TRACE* es para la depuración. Indica al servidor que regrese la solicitud. Este método es útil cuando las solicitudes no se están procesando de manera correcta y el cliente desea saber cual solicitud ha obtenido realmente el servidor.

El método *CONNECT* no se utiliza en la actualidad. Está reservado para uso futuro.

El método *OPTIONS* proporciona una forma para que el cliente consulte al servidor sobre sus propiedades o las de un archivo específico.

Cada solicitud obtiene una respuesta que consiste en una línea de estado (Ver figura 4). La línea de estado contiene un código de estado de tres dígitos que indica si la solicitud fue atendida, y si no, por qué. El primer digito se utiliza para dividir las respuestas en cinco grupos mayores. Los códigos 1xx no se utilizan con frecuencia. Los códigos 2xx significan que la solicitud se manejó de manera exitosa y que se regresa contenido. Los códigos 3xx indican al cliente que busque en otro lado. Los códigos 4xx significan que la solicitud fallo debido a un error del cliente, por ejemplo una solicitud invalida o una página inexistente. Los errores 5xx significan que el servidor tiene un problema, debido a un error en su código o a una sobrecarga temporal.

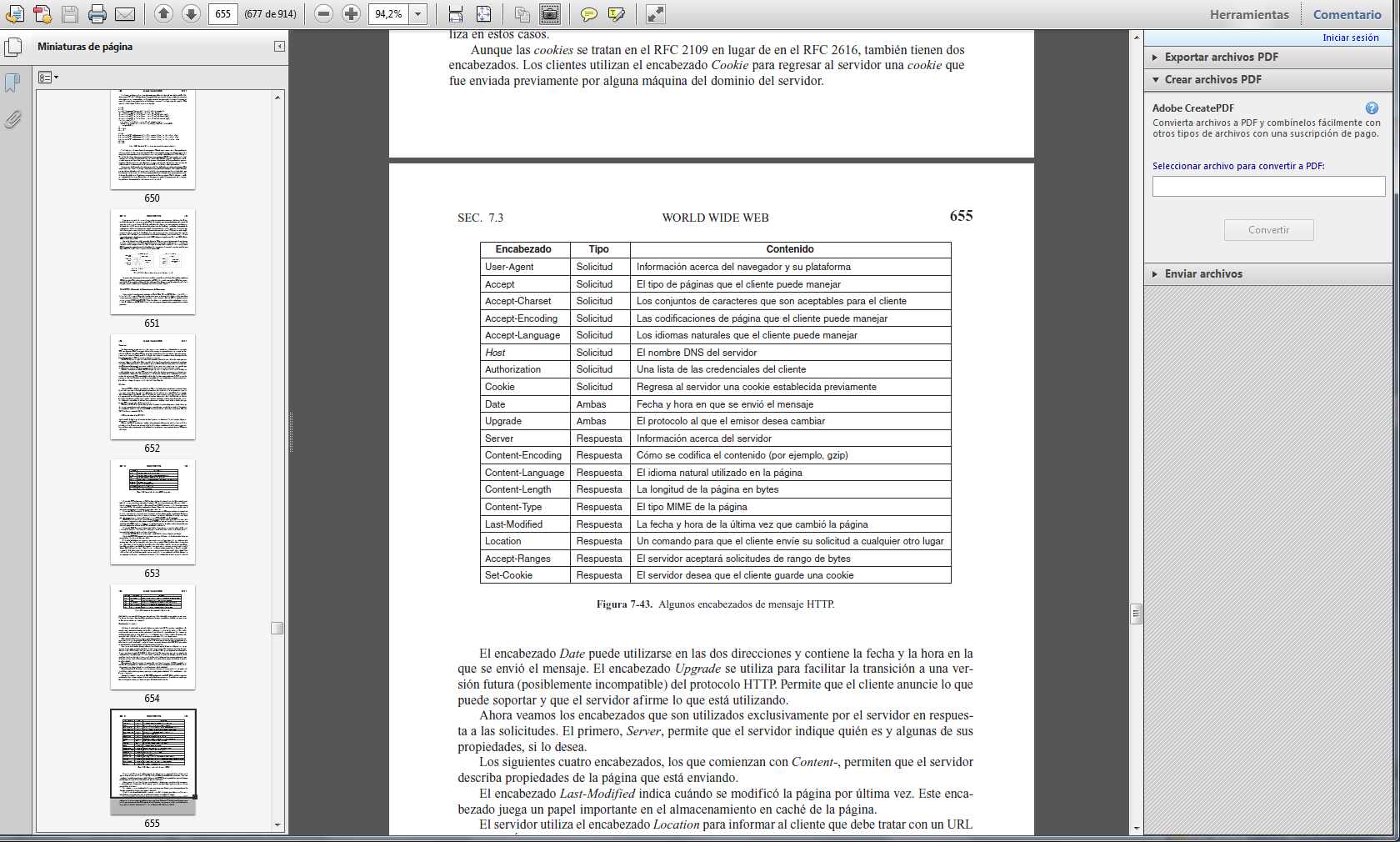
* **Figura 4**



## Encabezados de mensajes

A la línea de solicitud (por ejemplo, la línea con el método GET) le pueden seguir líneas adicionales que contienen mas información. Estas se llaman encabezados de solicitudes. Esta información pude compararse con los parámetros de una llamada a procedimiento. Las respuestas también pueden tener encabezados de respuesta. Ver figura 5.

* **Figura 5**



Veamos brevemente que significan cada uno de los encabezados.

El encabezado *User-Agent* permite que el cliente informe al servidor sobre su navegador, sistema operativo y otras propiedades.

Los cuatro encabezados *Accept* indican al servidor lo que el clienta está dispuesto a aceptar en caso de que tenga un repertorio limitado de lo que es aceptable. El primer encabezado especifica los tipos MIME aceptados (por ejemplo, txt/html). El segundo proporciona el conjunto de caracteres (por ejemplo, Unicode-1-1). El tercero tiene que ver con métodos de compresión (por ejemplo, gzip). El cuarto indica el idioma natural (por ejemplo, español). Si el servidor es incapaz de satisfacer la solicitud, se regresa un código de error y la solicitud falla.

El encabezado *Host* da nombre al servidor. Este encabezado se toma del URL y es obligatorio. Se utiliza porque algunas direcciones IP pueden proporcionar múltiples nombres DNS y el servidor necesita una forma de indicar a cual host entregarle la solicitud.

El encabezado *Authorization* es necesario para páginas que están protegidas. Por ejemplo, tal vez el cliente debe probar que tiene permiso para ver la página solicitada.

Las *cookies* también tienen dos encabezados. Los clientes utilizan el encabezado Cookie para regresar al servidor una cookie que fue enviada previamente por alguna maquina del dominio del servidor.

El encabezado *Date* puede utilizarse en las dos direcciones y contiene la fecha y la hora en la que se envió el mensaje.

El encabezado *Upgrade* se utiliza para facilitar la transición a una versión futura del protocolo HTTP. Permite que el cliente anuncie lo que puede soportar y que el servidor afirme lo que está utilizando.

Los encabezados que son utilizados exclusivamente por el servidor en respuestas a las solicitudes son:

El encabezado *Server*, permite que el servidor indique quién es y alguna de sus propiedades, si lo desea.

Los siguientes cuatro encabezados, los que comienzan con *Content-*, permiten que el servidor describa propiedades de la página que está enviando.

El encabezado *Last-Modified* indica cuándo se modificó la página por última vez. Este encabezado es muy importante en el almacenamiento en caché de la página.

El servidor utiliza el encabezado *Location* para informar al cliente que debe tratar con un URL diferente. Éste puede utilizarse si la página se movió o para permitir que múltiples URLs hagan referencia a la misma página.

Si una página es muy grande, tal vez un cliente pequeño no la quiera toda de una vez. Algunos servidores aceptan solicitudes de rangos de bytes, por lo que la página puede obtenerse en múltiples unidades pequeñas. El encabezado *Accept-Ranges* indica si el servidor está dispuesto a manejar este tipo de solicitudes parcial de páginas.

El segundo encabezado de *cookie*, *Set-Cookie*, es la forma en la que los servidores envían cookies a los clientes. Se espera que el cliente guarde la cookie y la regrese al servidor en solicitudes subsiguientes.

Códigos de estado de HTTP

Cuando se solicita al servidor una página de su sitio (por ejemplo, cuando un usuario accede a su página a través de un navegador), se muestra un código de estado de HTTP en respuesta a la solicitud.

Este código, que proporciona información acerca del estado de la solicitud, ofrece datos acerca del sitio y de la página solicitada.

Por ejemplo, algunos de los códigos de estado más frecuentes son:

1. **200** - El servidor ha mostrado la página correctamente.
2. **404** - La página solicitada no existe.
3. **503** - El servidor está temporalmente fuera de servicio.

A continuación se muestra una lista completa de códigos de estado de HTTP. Visite la página de W3C sobre códigos de estado de HTTP (http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html) para obtener más información al respecto.

## 1xx (Respuesta provisional)

Códigos de estado que indican una respuesta provisional y requieren que el solicitante realice una acción para poder continuar.

|  |  |
| --- | --- |
| **Código** | **Descripción** |
| 100 (Continuar) | El solicitante debe continuar con la solicitud. El servidor muestra este código para indicar que ha recibido la primera parte de una solicitud y que está esperando el resto. |
| 101 (Cambiando de protocolos) | El solicitante ha pedido al servidor que cambie los protocolos y el servidor está informando de que así lo hará. |

## **2xx** **(Correcto)**

Códigos de estado que indican que el servidor ha procesado la solicitud correctamente.

|  |  |
| --- | --- |
| **Código** | **Descripción** |
| 200 (Correcto) | El servidor ha procesado la solicitud correctamente. Generalmente, esto implica que el servidor ha proporcionado la página solicitada. Si aparece este estado al solicitar su archivo robots.txt, significa que el servidor lo ha recuperado correctamente. |
| 201 (Creado) | La solicitud se ha procesado correctamente y el servidor ha creado un nuevo recurso. |
| 202 (Aceptado) | El servidor ha aceptado la solicitud, pero todavía no la ha procesado. |
| 203 (Esta información no concede autorización) | El servidor ha procesado la solicitud correctamente, pero muestra información que puede proceder de otra fuente. |
| 204 (Sin contenido) | El servidor ha procesado la solicitud correctamente, pero no muestra ningún contenido. |
| 205 (Restablecer contenido) | El servidor ha procesado la solicitud correctamente, pero no muestra ningún contenido. A diferencia de la respuesta 204, esta requiere que el solicitante restablezca la vista del documento (por ejemplo, borrar los datos de un formulario para introducir nueva información). |
| 206 (Contenido parcial) | El servidor ha procesado una solicitud GET parcial correctamente. |

## **3xx** **(Redirigido)**

Es necesario llevar a cabo acciones adicionales para completar la solicitud. A menudo, estos códigos de estado se utilizan para el redireccionamiento. Puede utilizar Herramientas para webmasters para verificar si el servidor tiene problemas para rastrear sus páginas redireccionadas. En la página [Errores de rastreo](http://support.google.com/webmasters/bin/answer.py?answer=35156), dentro de la opción de estado, se muestran las URL que el servidor no pudo rastrear debido a errores de redireccionamiento.

|  |  |
| --- | --- |
| **Código** | **Descripción** |
| 300 (Varias opciones) | El servidor puede realizar varias acciones de acuerdo con la solicitud. Puede elegir una acción definida por el solicitante (user agent) o bien presentar una lista para que el solicitante elija una acción. |
| 301 (Movido permanentemente) | La página solicitada se ha movido definitivamente a una ubicación nueva. Cuando el servidor muestra esta respuesta (como respuesta a una solicitud GET o HEAD), dirige automáticamente al solicitante a la ubicación nueva. |
| 302 (Movido temporalmente) | El servidor responde a la solicitud con una página de otra ubicación, pero el solicitante debe seguir utilizando la ubicación original para solicitudes futuras. Este código es similar al 301 en que para una solicitud GET o HEAD, el sistema dirige automáticamente al solicitante a una ubicación diferente. Sin embargo, no se debe utilizar para comunicar al servidor que una página o un sitio se ha movido. |
| 303 (Ver otra ubicación) | El servidor muestra este código cuando el solicitante debe realizar una solicitud GET independiente a una ubicación diferente para poder obtener la respuesta. Para todas las solicitudes distintas de HEAD, el servidor dirige automáticamente al usuario a la ubicación nueva. |
| 304 (No modificado) | La página solicitada no ha sufrido cambios desde la última solicitud. Cuando el servidor muestra esta respuesta, no devuelve el contenido de la página.  Cuando una página no ha cambiado desde la última solicitud, debe configurar su servidor para que muestre esta respuesta (denominada cabecera "HTTP If-Modified-Since"). Esta función le ahorra ancho de banda y otros gastos, ya que su servidor puede comunicar que una página no ha cambiado desde la última vez que se rastreó. |
| 305 (Usar proxy) | El solicitante sólo puede acceder a la página solicitada mediante un proxy. Cuando el servidor muestra esta respuesta, también indica el proxy que debe utilizarse. |
| 307 (Redireccionamiento temporal) | El servidor responde a la solicitud con una página de otra ubicación, pero el solicitante debe seguir utilizando la ubicación original para solicitudes futuras. Este código es similar al 301 en que para una solicitud GET o HEAD, el sistema dirige automáticamente al solicitante a una ubicación diferente. |

## **4xx (Error de solicitud)**

Los códigos de estado siguientes indican que puede haberse producido un error en la solicitud que impidió al servidor procesarla.

|  |  |
| --- | --- |
| **Código** | **Descripción** |
| 400 (Solicitud incorrecta) | El servidor no ha entendido la sintaxis de la solicitud. |
| 401 (No autorizado) | La solicitud requiere autenticación. El servidor puede mostrar esta respuesta para una página que requiera información de acceso. |
| 403 (Prohibido) | El servidor ha rechazado la solicitud. Si recibe este código de estado al intentar rastrear las páginas válidas del sitio (se puede comprobar en la página **Errores de rastreo** de las Herramientas para webmasters, dentro de la opción de **estado**), es posible que el servidor o el host estén bloqueando el acceso. |
| 404 (No se encuentra) | El servidor no encuentra la página solicitada. El servidor a menudo muestra este código cuando, por ejemplo, se realiza una solicitud de una página que no existe en el servidor. |
| 405 (Método no permitido) | No se permite el método especificado en la solicitud. |
| 406 (Inaceptable) | No se puede ofrecer la página solicitada con las características de contenido requeridas. |
| 407 (Se requiere autenticación de proxy) | Este código de estado es similar al 401 (No autorizado), aunque en este caso se especifica que el solicitante debe autenticarse mediante un proxy. Cuando el servidor muestra esta respuesta, también indica el proxy que debe utilizarse. |
| 408 (El tiempo de espera de la solicitud ha caducado) | Se ha excedido el tiempo de espera de respuesta de la solicitud. |
| 409 (Conflicto) | El servidor ha detectado un conflicto al llevar a cabo la solicitud, por lo que debe incluir la información correspondiente en la respuesta. El servidor podría mostrar este código como respuesta a una solicitud PUT que entre en conflicto con una solicitud anterior junto con una lista de diferencias entre ambas. |
| 410 (No disponible permanentemente) | El servidor muestra esta respuesta cuando el recurso solicitado se ha eliminado definitivamente. Es similar al código "404 (No se encuentra)", aunque en ocasiones se utiliza en su lugar para identificar aquellos recursos que existieron anteriormente. Si el recurso se ha movido permanentemente, debe utilizar un código 301 para especificar su nueva ubicación. |
| 411 (Requiere longitud) | El servidor no aceptará la solicitud sin el campo válido "Content-Length" (longitud del contenido) en la cabecera. |
| 412 (Error de condición previa) | El servidor no cumple con una de las condiciones previas que el solicitante ha especificado en la solicitud. |
| 413 (Entidad de solicitud demasiado larga) | El servidor no puede procesar la solicitud porque es demasiada larga. |
| 414 (URI solicitada demasiado larga) | La URI solicitada (generalmente una URL) es demasiado larga para que el servidor la procese. |
| 415 (Tipo de soporte incompatible) | La solicitud se encuentra en un formato que la página solicitada no admite. |
| 416 (Intervalo solicitado no válido) | El servidor muestra este código de estado cuando se realiza una solicitud de un rango que no se encuentra disponible para la página. |
| 417 (Error de expectativa) | El servidor no puede cumplir los requisitos del campo de expectativa de solicitud en la cabecera. |

## **5xx (Error del servidor)**

Los códigos de estado siguientes indican que se ha producido un error interno del servidor al intentar procesar la solicitud. Estos errores suelen afectar al servidor, no a la solicitud.

|  |  |
| --- | --- |
| **Código** | **Descripción** |
| 500 (Error interno del servidor) | Se ha producido un error en el servidor y no puede completar la solicitud. |
| 501 (No implementado) | El servidor no dispone de las funciones necesarias para completar la solicitud. Este código puede mostrarse, por ejemplo, cuando el servidor no reconozca el método de solicitud. |
| 502 (Pasarela incorrecta) | Al actuar como pasarela o proxy, el servidor ha recibido una respuesta no válida del servidor ascendente. |
| 503 (Servicio no disponible) | El servidor no está disponible en estos momentos, debido a tareas de mantenimiento o a una sobrecarga. Generalmente, este es un estado temporal. |
| 504 (El tiempo de espera de la pasarela ha caducado) | Al actuar como pasarela o proxy, el servidor no ha recibido una solicitud puntual del servidor ascendente. |
| 505 (Versión de HTTP no compatible) | El servidor no es compatible con la versión del protocolo HTTP utilizada en la solicitud. |

¿Qué es un tipo MIME?

El tipo MIME (Multipurpose Internet Mail Extensions) (Ver figura 6) (extensiones multipropósito de correo en Internet) es un estándar propuesto por los laboratorios *Bell Communications* en 1991 para ampliar las posibilidades del correo electrónico al incluir la posibilidad de insertar documentos (imágenes, sonidos y texto) en un mensaje.

Desde entonces, el tipo MIME se usa para dar formato tanto los documentos adjuntos en un mensaje como a los documentos transferidos a través del protocolo HTTP. Así, durante una transacción entre un servidor web y un explorador de Internet, el servidor web envía en primer lugar el tipo MIME del archivo enviado al explorador para que éste sepa cómo se mostrará el documento.

Un tipo MIME está compuesto de la siguiente manera:

Content-type: tipo\_mime\_principal/subtipo\_mime

Por ejemplo, una imagen GIF tiene el siguiente tipo MIME:

Content-type: image/gif

* **Figura 6**

|  |  |  |
| --- | --- | --- |
| **Tipos MIME** | **Tipo de archivo** | **Extensión asociada** |
| application/acad | Archivos de AutoCAD | dwg |
| application/clariscad | Archivos de ClarisCAD | ccad |
| application/drafting | Archivos de bosquejo preliminar de MATRA | drw |
| application/dxf | Archivos de AutoCAD | dxf |
| application/i-deas | Archivos de SDRC I-deas | unv |
| application/iges | Formato de intercambio CAO IGES | igs, iges |
| application/octet-stream | Archivos binarios no interpretados | bin |
| application/oda | Archivos ODA | oda |
| application/pdf | Archivos Adobe Acrobat | pdf |
| application/postscript | Archivos PostScript | ai, eps, ps |
| application/pro\_eng | Archivos de ProEngineer | prt |
| application/rtf | Formato de texto enriquecido | rtf |
| application/set | Archivos CAO SET | set |
| application/sla | Archivos de estereolitografía | stl |
| application/solids | Archivos solids de MATRA | dwg |
| application/step | Archivos de datos STEP | step |
| application/vda | Archivos de superficie | vda |
| application/x-mif | Archivos de Framemaker | mif |
| application/x-csh | Secuencia de comandos C-Shell (UNIX) | dwg |
| application/x-dvi | Archivos de texto dvi | dvi |
| application/hdf | Archivos de datos | hdf |
| **Tipos MIME** | **Tipo de archivo** | **Extensión asociada** |
| application/x-latex | Archivos de LaTEX | latex |
| application/x-netcdf | Archivos de NetCDF | nc, cdf |
| application/x-sh | Secuencia de comandos Bourne Shell | dwg |
| application/x-tcl | Secuencia de comandos Tcl | tcl |
| application/x-tex | Archivos tex | tex |
| application/x-texinfo | Archivos eMacs | texinfo, texi |
| application/x-troff | Archivos Troff | t, tr, troff |
| application/x-troff-man | Archivos Troff/macro man | man |
| application/x-troff-me | Archivos Troff/macro ME | me |
| application/x-troff-ms | Archivos Troff/macro MS | ms |
| application/x-wais-source | Fuente Wais | src |
| application/x-bcpio | CPIO binario | bcpio |
| application/x-cpio | CPIO Posix | cpio |
| application/x-gtar | Tar GNU | gtar |
| application/x-shar | Archivos Shell | shar |
| application/x-sv4cpio | CPIO SVR4n | sv4cpio |
| application/x-sv4crc | CPIO SVR4n con CRC | sc4crc |
| application/x-tar | Archivos tar comprimidos | tar |
| application/x-ustar | Archivos tar Posix comprimidos | man |
| application/zip | Archivos ZIP comprimidos | man |
| audio/basic | Archivos de audio básicos | au, snd |
| audio/x-aiff | Archivos de audio AIFF | aif, aiff, aifc |
| audio/x-wav | Archivos de audio Wave | wav |
| image/gif | Imágenes Gif | man |
| image/ief | Imágenes con formato de intercambio | ief |
| image/jpeg | Imágenes jpeg | jpg, jpeg, jpe |
| image/tiff | Imágenes tiff | tiff, tif |
| image/x-cmu-raster | Ráster cmu | cmu |
| image/x-portable-anymap | Archivos Anymap PBM | pnm |
| image/x-portable-bitmap | Archivos de mapa de bits PBM | pbm |
| image/x-portable-graymap | Archivos Graymap PBM | pgm |
| image/x-portable-pixmap | Archivos Pixmap PBM | ppm |
| image/x-rgb | Imágenes RGB | rgb |
| image/x-xbitmap | Imágenes X Bitmap | xbm |
| image/x-xpixmap | Imágenes X Pixmap | xpm |
| image/x-xwindowdump | Imágenes de volcado X Window | man |
| multipart/x-zip | Archivos Zip almacenados | zip |
| multipart/x-gzip | Archivos Zip GNU almacenados | gz, gzip |
| text/html | Archivos HTML | htm, html |
| text/plain | Archivos de texto sin formato | txt, g, h, c, cc, hh, m, f90 |
| text/richtext | Archivos de texto enriquecido | rtx |
| **Tipos MIME** | **Tipo de archivo** | **Extensión asociada** |
| text/tab-separated-value | Archivos de texto con separación de valores | tsv |
| text/x-setext | Archivos de texto struct | etx |
| video/mpeg | Video MPEG | mpeg, mpg, mpe |
| video/quicktime | Videos de QuickTime | qt, mov |
| video/msvideo | Videos de Microsoft Windows | avi |
| video/x-sgi-movie | Videos de MoviePlayer | movie |

¿Qué es SOAP?

A diferencia de DCOM y CORBA, que son binarios, SOAP (Simple Object Access Protocol) usa el código fuente en XML, que facilita la eliminación de errores, pero es menos efectivo. El intercambio de mensajes se realiza mediante tecnología de componentes (software componentry). El término Object en el nombre significa que se adhiere al paradigma de la programación orientada a objetos.

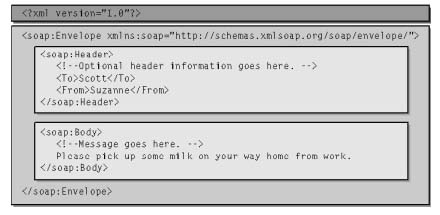
SOAP es un marco extensible y descentralizado que permite trabajar sobre múltiples pilas de protocolos de redes informáticas. Los procedimientos de llamadas remotas pueden ser modelados en la forma de varios mensajes SOAP interactuando entre sí.

SOAP corre sobre cualquier protocolo de Internet, generalmente HTTP, que es el único homologado por el W3C. SOAP tiene como base XML, con un diseño que cumple el patrón Cabecera-Desarrollo de diseño de software, como otros muchos diseños, verbigracia HTML. La cabecera Header es opcional y contiene metadatos sobre enrutamiento (routing), seguridad o transacciones. El desarrollo Body contiene la información principal, que se conoce como carga útil (payload). La carga útil se acoge a un XML Schema propio.

## Anatomía de un mensaje de SOAP

SOAP proporciona un mecanismo estándar de empaquetar un mensaje. Un mensaje SOAP se compone de un sobre que contiene el cuerpo del mensaje y cualquier información de cabecera que se utiliza para describir el mensaje. (**Ver Figura 7**)

* **Figura 7**



El elemento raíz del documento es el elemento Envelope. El ejemplo contiene dos subelementos, Body y Header. Un ejemplo de SOAP válido también puede contener otros elementos hijo en el sobre. El sobre puede contener un elemento Header opcional que contiene información sobre el mensaje. En el ejemplo anterior, la cabecera contiene dos elementos que describen a quien compuso el mensaje, y posible receptor del mismo. El sobre debe contener un elemento body el elemento body (cuerpo) contiene la carga de datos del mensaje. En el ejemplo el cuerpo contiene una simple cadena de caracteres.

Un mensaje debe estar dentro de sobre de SOAP bien construido. Un sobre se compone de un único elemento Envelope. El sobre puede contener un elemento Header y puede contener un elemento Body. Si existe, la cabecera debe ser el elemento hijo inmediato del sobre, con el cuerpo siguiendo inmediatamente a la cabecera. El cuerpo contiene la carga de datos del mensaje y la cabecera contiene los datos adicionales que no pertenecen necesariamente al cuerpo del mensaje.   
Además de definir un sobre de SOAP, la especificación de SOAP define una forma de codificar los datos contenidos en un mensaje. La codificación de SOAP proporciona un mecanismo estándar para serializar tipos de datos no definidos en la parte 1 de la especificación del esquema de XML.   
La especificación de SOAP también proporciona un patrón de mensaje estándar para facilitar el comportamiento de tipo RPC. Se emparejan dos mensajes de SOAP para facilitar la asociación de un mensaje de petición con un mensaje de respuesta. La llamada a un método y sus parámetros se serializan en el cuerpo del mensaje de petición en forma de una estructura. El elemento raíz tiene el mismo nombre que el método objetivo, con cada uno de los parámetros codificado como un subelemento.

El mensaje de respuesta puede contener los resultados de la llamada al método o una estructura de fallo bien definida. Los resultados de la llamada a un método se serializan en el cuerpo de la petición como una estructura. Por convenio, el elemento raíz tiene el mismo nombre que el método original al que se añade result. Los parámetros de retorno se serializan como elementos hijo, con el parámetro de retorno en primer lugar. Si se encuentra un error el cuerpo del mensaje de respuesta contendrá una estructura de fallo bien definida. Por convenio, el elemento raíz tiene el mismo nombre que el método original al que se añade result. Los parámetros de retorno se serializan como elementos hijo, con el parámetro de retorno en primer lugar. Si se encuentra un error el cuerpo del mensaje de respuesta contendrá una estructura de fallo bien definida.

## Algunas de las Ventajas de SOAP

No está asociado con ningún lenguaje: los desarrolladores involucrados en nuevos proyectos pueden elegir desarrollar con el último y mejor lenguaje de programación que exista pero los desarrolladores responsables de mantener antiguas aflicciones heredadas podrían no poder hacer esta elección sobre el lenguaje de programación que utilizan. SOAP no especifica una API, por lo que la implementación de la API se deja al lenguaje de programación, como en Java, y la plataforma como Microsoft .Net.

* No se encuentra fuertemente asociado a ningún protocolo de transporte: La especificación de SOAP no describe como se deberían asociar los mensajes de SOAP con HTTP. Un mensaje de SOAP no es más que un documento XML, por lo que puede transportarse utilizando cualquier protocolo capaz de transmitir texto.
* No está atado a ninguna infraestructura de objeto distribuido La mayoría de los sistemas de objetos distribuidos se pueden extender, y ya lo están alguno de ellos para que admitan SOAP.
* Aprovecha los estándares existentes en la industria: Los principales contribuyentes a la especificación SOAP evitaron, intencionadamente, reinventar las cosas. Optaron por extender los estándares existentes para que coincidieran con sus necesidades. Por ejemplo, SOAP aprovecha XML para la codificación de los mensajes, en lugar de utilizar su propio sistema de tipo que ya están definidas en la especificación esquema de XML. Y como ya se ha mencionado SOAP no define un medio de trasporte de los mensajes; los mensajes de SOAP se pueden asociar a los protocolos de transporte existentes como HTTP y SMTP.
* Permite la interoperabilidad entre múltiples entornos: SOAP se desarrollo sobre los estándares existentes de la industria, por lo que las aplicaciones que se ejecuten en plataformas con dicho estándares pueden comunicarse mediante mensaje SOAP con aplicaciones que se ejecuten en otras plataformas. Por ejemplo, una aplicación de escritorio que se ejecute en una PC puede comunicarse con una aplicación del back-end ejecutándose en un mainframe capaz de enviar y recibir XML sobre HTTP.

¿Cómo se resuelve el problema de seguridad de HTTP?

Si asistes a una charla sobre seguridad en internet seguro que, más bien pronto que tarde, aparecerá una frase parecida a: *asegúrate que, cuando entras a tu banco, al principio de la* barra de dirección puedes ver las siglas HTTPS. Y es cierto, tenemos que tenerlo en cuenta, pero ¿por qué?

Muchas veces la explicación se reduce a decir que se establece una conexión segura, y no es falso, pero la verdad es que podremos entender con un poquito más de profundidad en qué consiste HTTPS, y no hace falta hacerlo con palabras complicadas.

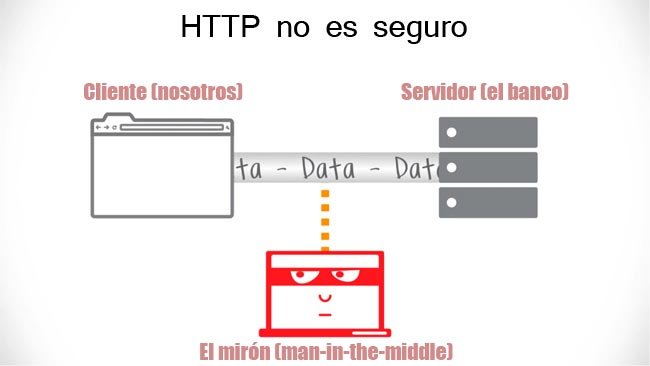
## ¿Qué problemas soluciona HTTPS?

Muchas veces, para comprender el porqué alguien se ha molestado en inventar algo complicado es mejor pensar para qué lo ha ideado. ¿Cuáles son los problemas que han motivado la aparición de este HTTPS? El principal problema es una falta de privacidad en nuestra navegación normal.

Normalmente, cuando navegamos por internet lo hacemos utilizando el protocolo HTTP, que simplemente establece unas directrices acerca de cómo se va a comunicar nuestro ordenador (cliente) con un servidor (por ejemplo, el ordenador donde está alojada la página de Gmail). Establece cómo se transfieren los datos, y en este caso, los datos se transfieren sin ninguna modificación, según los estás viendo ahora mismo.

Podemos imaginar que hay una tubería entre nuestro ordenador y el servidor por la que pasan los datos. Pero en esa tubería, alguien podría hacer un pequeño agujero por el que poder ver la información que viaja de un lado al otro (**Ver Figura 8**), o incluso algo peor, modificar la información que tu envías. En ámbitos de seguridad esto se refiere como un ataque man-in-the-middle (Hombre en el medio, literalmente).

* **Figura 8**



Esto no es un problema si alguien “mironea” mientras visitamos Gmail, pero posiblemente sea un problema mientras estás consultando los movimientos de tu tarjeta de crédito. Además, si *nuestro man-in-the-middle* se hace pasar por tu entidad bancaria… ¿cómo podrías estar seguro de que te responde tu banco? Aquí aparece HTTPS.

## ¿Cómo funciona HTTPS?

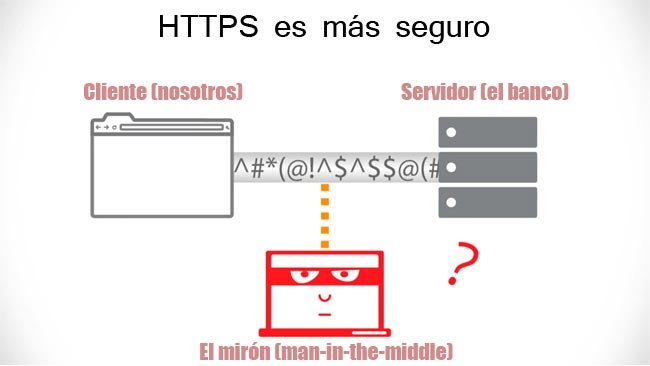
Ya que evitar “mirones” en una comunicación es algo que se puede catalogar como muy difícil o imposible, la solución ha consistido en hacer que nuestro amigo mirón no pueda comprender lo que mira a través de ese agujerito (**Ver Figura 9**) que ha hecho en la tubería, y que tampoco pueda meter en la tubería mensajes haciéndose pasar por nuestro banco (servidor) o por nosotros mismos (cliente).

Al menos en este asunto de las conexiones seguras, la magia no existe. Antes de empezar esa conexión segura, se establece una comunicación entre el cliente y el servidor en la que se acuerdan los detalles necesarios. Esta fase se denomina acuerdo, o en inglés handshake (apretón de manos), y es donde todo se complica un poco.

El handshake entre cliente y servidor puede variar dependiendo de los modos de autenticación (verificación de la identidad del remitente de un mensaje) y cifrado que se vayan a utilizar durante la conexión segura. Suena difícil, pero los sistemas utilizados no difieren mucho de los utilizados durante la II Guerra como por ejemplo el famoso código Enigma.

Imaginamos que somos capaces de crear dos códigos (o claves) especiales. De forma que un mensaje se puede cifrar con la primera y descifrar con la segunda, y viceversa. Así, un mensaje cifrado con la primera, ÚNICAMENTE podrá ser descifrado con la segunda, y un mensaje cifrado con la segunda, sólo podrá ser descifrado con la primera.

* **Figura 9**



Ahora viene el truco. La primera de esas claves la haces pública, esa misma es la que se debe enviar a todos las personas que desean descifrar mensajes tuyos. Pero la segunda es secreta, se debe guardar como un mayor tesoro. Con este pequeño truco podemos hacer dos cosas imprescindibles dentro del protocolo HTTPS:

1. **Verificar nuestra identidad**: Si un mensaje (o parte) lo ciframos con nuestro código secreto, cualquier persona será capaz de leerlo, pero nadie más habrá sido capaz de crearlo (necesitarían nuestra clave privada). Así es como se firma con SSL (Secure Socket Layer).
2. **Evitar mirones**: Si me quieren enviar un mensaje, se debe utilizar mi código público para cifrarlo. De esta forma sólo quien tenga mi código secreto será capaz de leerlo, es decir, sólo yo podré saber lo que significa. De hecho, ni siquiera Uds. serán capaces de leerlo una vez que esté cifrado.

Este esquema HTTPS es el que toda transacción comercial en Internet debe de usar. A este esquema se lo conoce como SSL.

## ¿SSL, cómo funciona?

En general, con tres elementos esenciales:

1. Protocolo de comunicaciones,
2. Certificado como un conjunto de pruebas para establecer la identidad y
3. Tercero confiable o Autoridad de Certificación que da fe sobre la legitimidad de las pruebas de identidad.

El protocolo de seguridad Secure Sockets Layer (SSL) permite las comunicaciones cifradas entre servidores y navegadores para garantizar los movimientos de los clientes y visitantes en los portales web de las Organizaciones.

Un Certificado SSL opera como una credencial en la industria electrónica. De esta manera, realiza la identificación de un dominio específico y un servidor web. La validez de esta credencial depende de la confianza en la Autoridad de Certificación que la emitió. A su vez, las Autoridades de Certificación como Symantec™ tienen métodos rigurosos para verificar la información proporcionada por las personas y Organizaciones que desean adquirir un Certificado de Seguridad.

En este sentido, Symantec™ destaca por ocupar el primer lugar como proveedor de seguridad. Dado que miles de millones de veces al día ayuda a las Organizaciones, visitantes y consumidores de todo el mundo a participar en las comunicaciones y el comercio de confianza.

Cuando algún usuario visita un sitio web seguro, el Certificado SSL proporciona información de identificación del servidor web y establece una conexión cifrada. Este proceso ocurre, instantáneamente, en fracciones de segundo. Mientras tanto, entre el navegador y el servidor web se da la siguiente secuencia:

1. El navegador intenta conectarse a un sitio web con SSL y solicita la identificación del servidor.
2. El servidor envía al navegador una copia de su certificado SSL.
3. El navegador comprueba si es posible confiar en el Certificado SSL y una vez confirmado envía un mensaje al servidor.
4. El servidor emite un acuse de recibo, firmado digitalmente, para iniciar una sesión SSL cifrada.
5. De esta forma, los datos encriptados se comparten entre navegador y servidor.

En resumen, SSL garantiza la integridad de las comunicaciones en las redes mediante Certificados Digitales que implican procesos de encriptación, autenticación y verificación. Es decir, proporciona autenticación de las partes que participan en las transacciones en línea y encripta las sesiones de comunicación.

15 preguntas con respuestas

1. ¿Qué es HTTP?

HTTP son las siglas en inglés de Hypertext Transfer Protocol. Es el protocolo más utilizado de Internet que permite la transferencia de texto en html para navegar.

1. Nombre 2 de las 6 características principales.

a) Permite la transferencia de objetos multimedia. El contenido de cada objeto intercambiado está identificado por su clasificación MIME.

b) No mantiene estado. Cada petición de un cliente a un servidor no es influida por las transacciones anteriores. El servidor trata cada petición como una operación totalmente independiente del resto.

1. ¿Sobre cuál protocolo se transporta HTTP? Amplíe.

HTTP se transporta sobre TCP. La forma común en que un navegador contacta a un servidor es estableciendo una conexión TCP con el puerto 80 de la máquina del servidor. El valor de utilizar TCP es que ni los navegadores ni los servidores tienen que preocuparse por los mensajes largos, perdidos o duplicados, ni por las confirmaciones de recepción. TCP maneja estos aspectos en su implementación

1. ¿Cuáles son las propiedades más importantes?

Las propiedades más importantes son tres: Conexiones, Métodos y Encabezados de Mensajes.

1. Nombre tres métodos utilizados por HTTP.

El método *GET* solicita al servidor que envíe la página.

El método *HEAD* simplemente solicita el encabezado del mensaje, sin la página real.

El método *TRACE* es para la depuración. Indica al servidor que regrese la solicitud.

1. ¿Explique a qué se refiere con Encabezados de Mensajes?

Encabezados de Mensajes se refiere a la línea de solicitud (por ejemplo, la línea con el método GET) le pueden seguir líneas adicionales que contienen mas información. Estas se llaman encabezados de solicitudes. Esta información pude compararse con los parámetros de una llamada a procedimiento.

1. Nombre tres encabezados utilizados por HTTP.

El encabezado *User-Agent* permite que el cliente informe al servidor sobre su navegador, sistema operativo y otras propiedades.

Los clientes utilizan el encabezado *Cookie* para regresar al servidor una cookie que fue enviada previamente por alguna maquina del dominio del servidor.

El encabezado *Server*, permite que el servidor indique quién es y alguna de sus propiedades, si lo desea.

1. ¿Qué es un código de estado de HTTP?

Cuando se solicita al servidor una página de su sitio (por ejemplo, cuando un usuario accede a su página a través de un navegador), se muestra un código de estado de HTTP en respuesta a la solicitud.

1. ¿Cuáles son los código de estado más frecuentes?

Los códigos de estado más frecuentes son:

**200** - El servidor ha mostrado la página correctamente.

**404** - La página solicitada no existe.

**503** - El servidor está temporalmente fuera de servicio.

1. Explique el significado de al menos dos grupos de los códigos de estados.

1xx (Respuesta provisional): son los códigos de estado que indican una respuesta provisional y requieren que el solicitante realice una acción para poder continuar.

4xx (Error de solicitud): son los códigos de estado que indican que puede haberse producido un error en la solicitud que impidió al servidor procesarla.

5xx (Error del servidor): son los códigos de estado que indican que se ha producido un error interno del servidor al intentar procesar la solicitud. Estos errores suelen afectar al servidor, no a la solicitud.

1. ¿Qué es un tipo MIME?

El tipo MIME de sus siglas en inglés Multipurpose Internet Mail Extensions (extensiones multipropósito de correo en Internet) es un estándar propuesto por los laboratorios *Bell Communications* en 1991 para ampliar las posibilidades del correo electrónico al incluir la posibilidad de insertar documentos (imágenes, sonidos y texto) en un mensaje.

1. ¿Qué es SOAP?

SOAP por sus siglas en inglés Simple Object Access Protocol es un marco extensible y descentralizado que permite trabajar sobre múltiples pilas de protocolos de redes informáticas. SOAP es un protocolo “liviano” basado en XML, para invocar procedimientos o servicios web en forma remota. Utiliza cualquier protocolo que permita transportar mensajes de texto, siendo HTTP el más utilizado.

1. ¿Qué problema importante presenta HTTP?

Un problema importante de HTTP es que si se escucha la red mientras se está estableciendo una conexión HTTP, los datos viajan en texto plano. Eso quiere decir que cualquier intruso puede robar esa información para hacerse pasar por uno.

1. ¿Qué es HTTPS?

HTTPS por sus siglas en inglés Hypertext Transfer Ptotocol Secure hace que HTTP sea seguro porque utiliza SSL estableciendo encriptación entre el cliente y el servidor web.

1. ¿Qué es y cómo opera SSL?

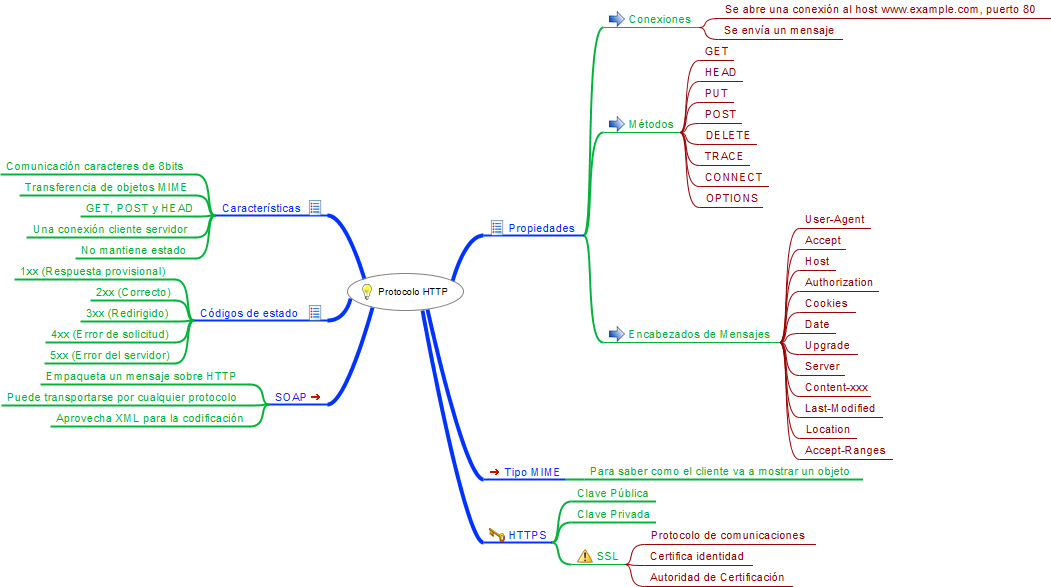
El protocolo de seguridad SSL por sus siglas en inglés Secure Sockets Layer permite las comunicaciones cifradas entre servidores y navegadores.

Un Certificado SSL opera como una credencial. De esta manera, realiza la identificación de un dominio específico y un servidor web. La validez de esta credencial depende de la confianza en la Autoridad de Certificación que la emitió.

Esto hace que HTTP se convierta en un medio seguro utilizando HTTPS.

Bibliografía utilizada

* Tanenbaum, Andrew S. (2003). Redes de computadoras (4ª ed.).
* Estándar Hypertext Transfer Protocol -- HTTP/1.1. Recuperado de <http://www.w3.org/Protocols/rfc2616/rfc2616.html> .
* ¿Qué es un tipo MIME? Recuperado de <http://es.kioskea.net/contents/systemes/mime.php3> .
* Descripción de HTTP. Recuperado de <http://www.cicei.com/ocon/gsi/tut_tcpip/3376c426.html> .
* Definición de Códigos de Estado. Recuperado de <http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html> y <http://support.google.com/webmasters/bin/answer.py?hl=es&answer=40132> .
* Protocolo SOAP. Recuperado de <http://www.gxtechnical.com/gxdlsp/pub/genexus/internet/docum/releasenotes/7.5/callsoap.htm> y <http://www.desarrolloweb.com/articulos/1557.php> .
* ¿SSL, cómo funciona? Recuperado de <http://www.certsuperior.com/SSLComoFunciona.aspx> .

Mapa Conceptual Mental