CAPÍTULO 5: PROTOCOLO HTTP

En este proyecto, se establece que los clientes, a través de la aplicación instalada en sus terminales, accedan al servicio que le proporciona la transacción económica, de alguna forma. Puesto que tanto el servicio como el terminal móvil admiten conexiones HTTP, vamos a usar este protocolo para realizar la comunicación. Por ello se va a proceder a realizar una descripción de dicho protocolo.

El Protocolo de Transferencia de HiperTexto (*Hypertext Transfer Protocol*) es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP. La especificación completa del protocolo HTTP 1/1 está recogida en el RFC 2616. Fue propuesto por Tim Berners-Lee, atendiendo a las necesidades de un sistema global de distribución de información como el World Wide Web.

5.1.- Características y funcionamiento

Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión **TCP/IP**: 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.

HTTP se basa en sencillas operaciones de solicitud/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web es conocido por su URL.

Los recursos u objetos que actúan como entrada o salida de un comando **HTTP** están clasificados por su descripción MIME. De esta forma, el protocolo puede intercambiar cualquier tipo de dato, sin preocuparse de su contenido. La transferencia se realiza en modo binario, byte a byte, y la identificación MIME permitirá que el receptor trate adecuadamente los datos.

Las principales **características del protocolo HTTP** son:

- 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.

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

- 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).

- 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. En la actualidad se ha mejorado este procedimiento, permitiendo que una misma conexión se mantenga activa durante un cierto periodo de tiempo, de forma que sea utilizada en sucesivas transacciones. Este mecanismo, denominado *HTTP Keep Alive*, es empleado por la mayoría de los clientes y servidores modernos.

- 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.

- Cada objeto al que se aplican los verbos del protocolo está identificado a través de la información de situación del final de la URL.

Cada vez que un cliente realiza una petición a un servidor, se ejecutan los siguientes pasos:

1. Un usuario accede a una **URL**, seleccionando un enlace de un documento **HTML** o introduciéndola directamente en el campo Dirección del cliente Web.

2. El cliente Web descodifica la **URL**, separando sus diferentes partes. Así identifica el protocolo de acceso, la dirección DNS o IP del servidor, el posible puerto opcional (el valor por defecto es 80) y el objeto requerido del servidor.

3. Se abre una conexión **TCP/IP** con el servidor, llamando al puerto TCP correspondiente.

4. Se realiza la petición. Para ello, se envía el comando necesario (**GET**, **POST**, **HEAD**,…), la dirección del objeto requerido (el contenido de la URL que sigue a la dirección del servidor), la versión del protocolo HTTP empleada y un conjunto variable de información, que incluye datos sobre las capacidades del navegador, datos opcionales para el servidor, etc.

5. El servidor devuelve la respuesta al cliente. Consiste en un código de estado y el tipo de dato MIME de la información de retorno, seguido de la propia información.

6. Se cierra la conexión **TCP**. Si no se utiliza el modo *HTTP Keep Alive*, este proceso se repite para cada acceso al servidor **HTTP**.

El diálogo con los servidores HTTP se establece a través de mensajes formados por líneas de texto, cada una de las cuales contiene los diferentes comandos y opciones del protocolo. Sólo existen dos **tipos de mensajes**, uno para realizar peticiones y otro para devolver la correspondiente respuesta. La estructura general de los dos tipos de mensajes se puede ver en el siguiente esquema:

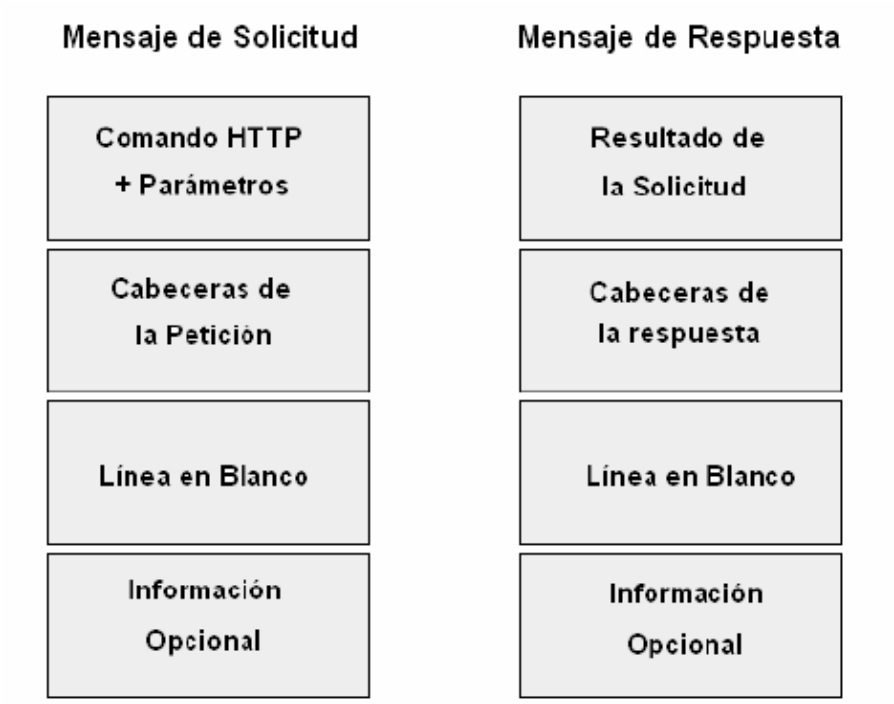

Figura 5. 1: Tipos de mensajes

La primera línea del mensaje de solicitud contiene el comando que se solicita al servidor **HTTP**, mientras que en la respuesta contiene el resultado de la operación que es un código numérico que permite conocer el éxito o fracaso de la operación. Después aparece, para ambos tipos de mensajes, un conjunto de cabeceras (unas obligatorias y otras opcionales), que condicionan y matizan el funcionamiento del protocolo.

La separación entre cada línea del mensaje se realiza con un par *CR-LF* (retorno de carro más nueva línea). El final de las cabeceras se indica con una línea en blanco, tras la cual se pueden incluir los datos transportados por el protocolo, por ejemplo, el documento HTML que devuelve un servidor.

5.2.- Comandos del protocolo HTTP

El protocolo HTTP consta de los siguientes comandos que permiten realizar las acciones pertinentes:

- **GET**, que se utiliza para recoger cualquier tipo de información del servidor. Se utiliza siempre que se pulsa sobre un enlace o se teclea directamente a una URL. Como resultado, el servidor HTTP envía el documento correspondiente a la URL.

Este comando puede ir acompañado de una cabecera con los siguientes parámetros:

- ***Accept***: indica los formatos de archivo que puede soportar el cliente.
- ***Accept-Charset***: indica los conjuntos de caracteres que acepta.
- ***Accept-Encoding***: el tipo de codificación que acepta.
- ***Accept-Language***: el lenguaje en el que se hará la respuesta de la cabecera.
- ***Authorization***: clave para tener acceso a lugares restringidos donde se requiere nombre de contraseña y el formato de autorización empleado.
- ***Connection***: especifica las opciones requeridas para una conexión.
- ***Date***: representa la fecha y la hora a la que se envió el comando.
- ***From***: campo opcional que incluye la dirección de correo electrónico del usuario del cliente.
- ***Host***: especifica la dirección del host del cliente y el puerto de conexión que ha empleado.
- ***If-Modified-Since***: condición de que sólo se responda a la solicitud si la fecha de última modificación del objeto es superior a la indicada en su parámetro.
- ***If-UnModified-Since***: sólo se responde a la solicitud si la fecha de última modificación del objeto no ha cambiado a partir de la fecha indicada en su parámetro.
- ***Pragma***: para incluir directivas de implementación.
- ***Proxy-Authorization***: para identificarse a un Proxy.
- ***Referer***: contiene la URL de la página que le ha dado acceso a la que está solicitando. Esto puede interesarle a los autores de páginas Web para saber en que lugares existen enlaces de su página.
- ***Range***: establecer un rango de Bytes del contenido de la respuesta.
- **Transfer-Encoding**: indica el tipo de codificación aplicada del contenido que responderá el servidor.
- ***Upgrade***: especifica qué protocolos suplementarios soporta el cliente, si también soportara FTP u otros.
- ***User-Agent***: especifica cual es el nombre del cliente y su versión. Tiene un segundo parámetro corresponde al sistema operativo donde corre el cliente.

- ***Via***: sirve para obtener la ruta de routers que ha recorrido el mensaje.

- **HEAD**, que es igual que el comando GET, y puede usar las mismas cabeceras, pero este comando le pide al servidor que le envíe sólo la cabecera como respuesta. Es utilizado por los gestores de cachés de páginas o los servidores Proxy, para conocer cuándo es necesario actualizar la copia que se mantiene de un fichero.

- **POST**, que tiene un funcionamiento idéntico al comando HEAD, pero además, este comando envía datos de información al servidor, normalmente originaria de un formulario Web para que el servidor los administre o los añada a una base de datos.

En la versión 1.1 se han definido algunos comandos adicionales, que sólo están disponibles en determinadas versiones de servidores HTTP, con motivos eminentemente experimentales, que se pueden utilizar, por ejemplo, para editar las páginas de un servidor Web trabajando en remoto.

Cuando el servidor envía la respuesta al cliente, le envía la información solicitada y una cabecera con una serie **campos**, estos campos son:

- ***Age***: tiempo transcurrido desde que se creó la respuesta.

- ***Allow***: especifica qué comandos puede utilizar el cliente respecto al objeto pedido, como pueden ser GET o HEAD.

- ***Expires***: fecha en la que caducará el objeto adjunto.

- ***Last-Modified***: fecha de la última modificación del objeto.

- ***Location***: se usa para redirigir la petición a otro objeto. Informa sobre la dirección exacta del objeto al que se ha accedido.

- ***Proxy-Authenticate***: indica el esquema de autentificación en caso de que un Proxy lo requiera.

- ***Public***: da una lista de los comandos que reconoce el servidor.

- ***Retry-After***: si no puede ofrecer un objeto provisionalmente, indica la fecha para que se vuelva a hacer la petición.

- ***Server***: especifica el tipo y versión del servidor.

- ***Vary***: indica que hay más de una posible respuesta a la petición pero solo elige una.

- ***Warning***: específica información adicional sobre el estado de la respuesta.

- ***WWW-Autenticate***: usado cuando se accede a un lugar restringido, sirve para informar de las maneras de autentificarse que soporta el objeto del servidor.

Hay otros **parámetros de información de cabeceras**, que son válidos tanto para mensajes del cliente como para respuestas del servidor. Estas cabeceras son:

- ***Content-Type***: descripción MIME de la información que contiene el mensaje para dar el tratamiento conveniente a los datos que se envían.

- ***Content-Length***: longitud en Bytes de los datos enviados.

- ***Content-Encoding***: especifica el formato en el que están codificados los datos enviados.

- ***Date***: fecha local de la operación. Las fechas deben incluir la zona horaria en que reside el sistema que genera la operación. Por ejemplo: *Sunday, 12- Dec-96 12:21:22 GMT+01*. No existe un formato único en las fechas; incluso es posible encontrar casos en los que no se dispone de la zona horaria correspondiente, con los problemas de sincronización que esto produce. Los formatos de fecha a emplear están recogidos en las *RFC 1036* y *1123*.

- ***Pragma***: Permite incluir información variada relacionada con el protocolo HTTP en la solicitud o respuesta que se está realizando. Por ejemplo, un cliente envía un *Pragma: no-caché* para informar de que desea una copia nueva del recurso especificado.

Ante cada transacción con un servidor HTTP, éste devuelve un **código numérico** que informa sobre el resultado de la operación, como primera línea del mensaje de respuesta. Estos códigos aparecen en algunos casos en la pantalla del cliente, cuando se produce un error.

Los **códigos** de estados están clasificados en **cinco categorías**:

- ***1xx***: mensajes informativos.

- ***2xx***: mensajes asociados con operaciones realizadas correctamente.

- ***3xx***: mensajes de redirección, que informan de operaciones complementarias que se deben realizar para finalizar la operación.

- ***4xx***: errores del cliente; el requerimiento contiene algún error, o no puede ser realizado.

- ***5xx***: errores del servidor, que no ha podido llevar a cabo una solicitud.

La lista de códigos es la que podemos ver en la tabla siguiente:

Código	***Comentario***	***Descripción***
100	Continue	Se usa para informar al cliente que la parte inicial de la petición se ha recibido y no ha sido rechazada por el servidor.
101	Switching Protocols	El servidor está de acuerdo con el cambio de protocolo expuesto en cabecera *Upgrade.*
200	OK	Operación realizada satisfactoriamente.
201	Created	La operación ha sido realizada correctamente, y como resultado se ha creado un nuevo objeto, cuya URL de acceso se proporciona en el cuerpo de la respuesta. Este nuevo objeto ya está disponible. Puede ser utilizado en sistemas de edición de documentos.
202	Accepted	La operación ha sido realizada correctamente, y como resultado se ha creado un nuevo objeto, cuya URL de acceso se proporciona en el cuerpo de la respuesta. El nuevo objeto no está disponible por el momento. En el cuerpo de la respuesta se debe informar sobre la disponibilidad de la información.
204	No Content	La operación ha sido aceptada, pero no ha producido ningún resultado de interés. El cliente no deberá modificar el documento que está mostrando en este momento.
301	Moved Permanently	El objeto al que se accede ha sido movido a otro lugar de forma permanente. El servidor proporciona, además, la nueva URL en la variable *Location* de la respuesta. Algunos browsers acceden automáticamente a la nueva URL. En caso de tener capacidad, el cliente puede actualizar la URL incorrecta.
302	Moved Temporarily	El objeto al que se accede ha sido movido a otro lugar de forma temporal. El servidor proporciona, además, la nueva URL en la variable *Location* de la respuesta. Algunos browsers acceden automáticamente a la nueva URL. El cliente no debe modificar ninguna de las referencias a la URL errónea.
304	Not Modified	Cuando se hace un GET condicional, y el documento no ha sido modificado, se devuelve este código de estado.
400	Bad Request	La petición tiene un error de sintaxis y no es entendida por el servidor.
401	Unauthorized	La petición requiere una autorización especial, que normalmente consiste en un nombre y clave que el servidor verificará. El campo *WWW-Autenticate* informa de los protocolos de autentificación aceptados para este recurso.

403	Forbidden	Está prohibido el acceso a este recurso. No es posible utilizar una clave para modificar la protección.
404	Not Found	La URL solicitada no existe.
500	Internal Server Error	El servidor ha tenido un error interno, y no puede continuar con el procesamiento.
501	Not Implemented	El servidor no tiene capacidad, por su diseño interno, para llevar a cabo el requerimiento del cliente.
502	Bad Gateway	El servidor, que está actuando como Proxy o pasarela, ha encontrado un error al acceder al recurso que había solicitado el cliente.
503	Service Unavailable	El servidor está actualmente deshabilitado, y no es capaz de atender el requerimiento.

Tabla 5. 1: Lista de códigos de estado HTTP

El protocolo **HTTP** está muy extendido en el mundo de Internet, y cualquier usuario de Internet posee un navegador Web, con el que se podrá conectar con el servidor Web implementado sin tener que realizar ninguna otra operación que solicitar una página Web como se hace normalmente. Así pues se ha optado por adoptar el protocolo HTTP para la comunicación entre cliente y servidor.