**Los métodos más importantes de HTTP (especialmente para hacer aplicaciones REST) son POST, GET, PUT, DELETE y HEAD.**

**GET**

El método GET se emplea para leer una representación de un resource. En caso de respuesta positiva (200 OK), GET devuelve la representación en un formato concreto: HTML, XML, JSON o imágenes, JavaScript, CSS, etc. En caso de respuesta negativa devuelve 404 (not found) o 400 (bad request). Por ejemplo en la carga de una página web, primero se carga la url solicitada:

GET php.net/docs HTTP/1.1

En este caso devolverá HTML. Y después los demás resources como CSS, JS, o imágenes:

GET php.net/images/logo.png HTTP/1.1

Los formularios también pueden usarse con el [método GET](http://diego.com.es/get-y-post-en-php), donde se añaden los keys y values buscados a la URL del header:

<**form** action="formget.php" method="get">

Nombre: <**input** type="text" name="nombre"><**br**>

Email: <**input** type="text" name="email"><**br**>

<**input** type="submit" value="Enviar">

</**form**>

La URL con los datos rellenados quedaría así:

GET ejemplo.com/formget.php?nombre=pepe&email=pepe%40ejemplo.com HTTP/1.1

**POST**

Aunque se puedan enviar datos a través del método GET, en muchos casos se utiliza POST por las limitaciones de GET. En caso de respuesta positiva devuelve 201 (created). Los POST requests se envían normalmente con formularios:

<**form** action="formpost.php" method="post">

Nombre: <**input** type="text" name="nombre"><**br**>

Email: <**input** type="text" name="email"><**br**>

<**input** type="submit" value="Enviar">

</**form**>

Rellenar el formulario anterior crea un HTTP request con la request line:

POST /formpost.php HTTP/1.1

El contenido va en el body del request, no aparece nada en la URL, aunque se envía en el mismo formato que con el método GET. Si se quiere enviar texto largo o cualquier tipo de archivo este es el método apropiado.

Le siguen los headers, donde se incluyen algunas líneas específicas con información de los datos enviados:

Content-Type: application/x-www-form-urlencoded

Content-Length: 43

A los headers le siguen una línea en blanco y a continuación el contenido del request:

formpost.php?nombre=pepe&email=pepe%40ejemplo.com

**PUT**

Utilizado normalmente para actualizar contenidos, pero también pueden crearlos. Tampoco muestra ninguna información en la URL. En caso de éxito devuelve 201 (created, en caso de que la acción haya creado un elemento) o 204 (no response, si el servidor no devuelve ningún contenido). A diferencia de POST es idempotente, si se crea o edita un resource con PUT y se hace el mismo request otra vez, el resource todavía está ahí y mantiene el mismo estado que en la primera llamada. Si con una llamada PUT se cambia aunque sea sólo un contador en el resource, la llamada ya no es idempotente, ya que se cambian contenidos.

PUT ejemplo.com/usuario/peter HTTP/1.1

**DELETE**

Simplemente elimina un resource identificado en la URI. Si se elimina correctamente devuelve 200 junto con un body response, o 204 sin body. DELETE, al igual que PUT y GET, también es idempotente.

DELETE ejemplo.com/usuario/peter HTTP/1.1

**HEAD**

Es idéntido a GET, pero el servidor no devuelve el contenido en el HTTP response. Cuando se envía un HEAD request, significa que sólo se está interesado en el código de respuesta y los headers HTTP, no en el propio documento. Con este método el navegador puede comprobar si un documento se ha modificado, por razones de caching. Puede comprobar también directamente si el archivo existe.

Por ejemplo, si tienes muchos enlaces en tu sitio web, puedes enviar un HEAD request a todos los enlaces para comprobar los que estén rotos. Es bastante más rápido que hacerlo con GET.