HTTP REQUESTs

HTTP request

HTTP define un **conjunto de métodos de petición** para indicar la acción que se desea realizar para un recurso determinado.

Los más utilizados son GET, POST, PUT, DELETE y HEAD.

Método GET

El método GET se emplea para **leer** una representación de un recurso.

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).

Método GET

Los formularios también pueden usarse con el método GET, donde se añaden los keys y values buscados a la URL del header:

```
<form action="formulario.php" method="get">
    Nombre: <input type="text" name="nombre"><br>
    Apellido: <input type="text" name="apellido"><br>
    <input type="submit" value="Enviar">
</form>
```

Luego, la URL quedaría:

mi-proyecto.com/formulario.php?nombre=julian&apellido=butti%40proyecto.com

Método POST

El método POST es similar al GET pero el contenido se envía en el body del request, nada aparece en la url.

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

En caso de **respuesta positiva** devuelve 201 (created).

En caso de **respuesta negativa** devuelve 404 (not found) o 400 (bad request).

Método PUT

El método PUT es similar al POST y generalmente se utiliza para actualizar, pero también puede usarse para crear. Tampoco muestra ninguna información en la URL.

En caso de **respuesta positiva** devuelve 201 (created, en caso de que la acción haya creado un elemento) o 204 (no response, si el servidor no devuelve ningún contenido).

En caso de **respuesta negativa** devuelve 404 (not found) o 400 (bad request).

A diferencia de POST es idempotente, si se crea o edita un resource con PUT y se hace el mismo request otra vez, el recurso 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 recurso, la llamada ya no es idempotente, ya que se cambian contenidos.

Método delete

El método DELETE elimina un recurso identificado en la URI.

En caso de **respuesta positiva** devuelve 200 junto con un body response o 204 sin body.

En caso de **respuesta negativa** devuelve 404 (not found) o 400 (bad request).

DELETE, al igual que PUT y GET, también es idempotente.

Método HEAD

El método HEAD es idéntico 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.

CÓDIGOS DE ESTADO HTTP

Códigos de estado HTTP

Los códigos de estado HTTP describen de forma abreviada la respuesta HTTP.

El **primer dígito del código de estado** especifica uno de los 5 tipos de respuesta, el mínimo para que un cliente pueda trabajar con HTTP es que reconozca estas 5 clases.

- 1XX Respuestas informativas
- 2XX Peticiones correctas
- 3XX Redirecciones
- 4XX Errores del cliente
- 5XX Errores de servidor

1xx Respuestas informativas

Este tipo de código de estado indica una respuesta provisional.

100 Continue. El servidor ha recibido los headers del request y el cliente debería proceder a enviar el cuerpo de la respuesta.

101 Switching Protocols. El requester ha solicitado al servidor conmutar protocolos.

102 Processing (WebDAV; RFC 2518). Usado en requests para reanudar peticiones PUT o POST abortadas.

2xx Peticiones correctas

Este código de estado indica que la acción solicitada por el cliente ha sido recibida, entendida, aceptada y procesada correctamente.

200 OK. El request es correcto. Esta es la respuesta estándar para respuestas correctas.

201 Created. El request se ha completado y se ha creado un nuevo recurso.

202 Aceptada. El request se ha aceptado para procesarlo, pero el proceso aún no ha terminado.

203 Non-Authoritative Information. El request se ha procesado correctamente, pero devuelve información que podría venir de otra fuente.

204 No Content. El request se ha procesado correctamente, pero no devuelve ningún contenido.

205 Reset Content. El request se ha procesado correctamente, pero no devuelve ningún contenido y se requiere que el requester recargue el contenido.

206 Partial Content. El servidor devuelve sólo parte del recurso debido a una limitación que ha configurado el cliente (se usa en herramientas de descarga como wget).

207 Multi-Status (WebDAV; RFC 4918). El cuerpo del mensaje es XML y puede contener un número de códigos de estado diferentes dependiendo del número de sub-requests.

3xx Redirecciones

El cliente ha de tomar una acción adicional para completar el request. Muchos de estos estados se utilizan para redirecciones. El user-agent puede llevar a cabo la acción adicional sin necesidad de que actúe el usuario sólo si el método utilizado en la segunda petición es GET o HEAD. Un user-agent no debería redireccionar automáticamente un request más de cinco veces, ya que esas redirecciones suelen indicar un infinite loop.

300 Multiple Choices. Es una lista de enlaces. El usuario puede seleccionar un enlace e ir a esa dirección. Hay un máximo de cinco direcciones.

301 Moved Permanently. La página solicitada se ha movido permanentemente a una nueva URI.

302 Found. La página solicitada se ha movido temporalmente a una nueva URI.

303 See Other. La página solicitada se puede encontrar en una URI diferente.

304 Not Modified. Indica que la página solicitada no se ha modificado desde la última petición.

305 Use Proxy (desde HTTP/1.1). El recurso solicitado sólo está disponible a través de proxy, cuya dirección se proporciona en la respuesta. Muchos clientes HTTP como Mozilla o Internet Explorer no manejan bien estas respuestas con estos códigos de estado, sobre todo por seguridad.

307 Temporary Redirect (desde HTTP/1.1). La página solicitada se ha movido temporalmente a otra URL. En este caso el recurso debería de repetirse con otra URI, sin embargo, futuros requests deberán usar la URI original. Al contrario que con la 302, el método request no puede cambiar cuando se reubique el request original.

308 Permanent Redirect (RFC 7538). El request y futuros requests deberían repetirse usando otro URI. Este también es similar al 301, pero no permite al método HTTP que cambie.

4xx Errores del cliente

Excepto cuando se responde a un HEAD request, el servidor debe incluir una entidad que contiene una explicación del error, y si es temporal o permanente. Son aplicables a cualquier método de solicitud (GET, POST...). Los user agents deben mostrar cualquier entidad al usuario.

400 Bad Request. El servidor no puede o no va a procesar el request por un error de sintaxis del cliente.

401 Unauthorized (RFC 7235). Similar al error 403, pero se usa cuando se requiere una autentificación y ha fallado o todavía no se a facilitado.

402 Payment Required. Reservado para futuro uso. La intención original fue para pago con tarjeta o micropago, pero eso no ha ocurrido, y este código apenas se usa.

403 Forbidden. El request fue válido pero el servidor se niega a responder.

404 Not Found. El recurso del request no se ha podido encontrar pero podría estar disponible en el futuro. Se permiten requests subsecuentes por parte del cliente.

405 Method Not Allowed. Se ha hecho un request con un recurso usando un método request no soportado por ese recurso (por ejemplo usando GET en un formulario que requiere POST).

4xx Errores del cliente

406 Not Acceptable. El recurso solicitado solo genera contenido no aceptado de acuerdo con los headers Accept enviados en el request.

407 Proxy Authentication Required (RFC 7235). El cliente se debe identificar primero con el proxy.

408 Request Timeout. El cliente no ha enviado un request con el tiempo necesario con el que el servidor estaba preparado para esperar. El cliente podría repetir el request sin modificaciones más tarde.

409 Conflict. Conflicto en el request, como cuando se actualizan al mismo tiempo dos recursos.

410 Gone. El recurso solicitado no está disponible ni lo estará en el futuro. Un buscador eliminará antes una página 410 que una 404.

411 Length Required. El request no especificó la longitud del contenido, la cual es requerida por el recurso solicitado.

412 Precondition Failed (RFC 7232). El servidor no cumple una de las precondiciones que el requester añade en el request.

4xx Errores del cliente

413 Request Entity Too Large. El request es más largo que el que está dispuesto a aceptar el servidor.

414 Request-URI Too Long. El URI es muy largo para que el servidor lo procese.

415 Unsupported Media Type. La entidad request tiene un media type que el servidor o recurso no soportan.

416 Requested Range Not Satisfiable (RFC 7233). El cliente ha solicitado una porción de archivo, pero el servidor no puede ofrecer esa porción.

417 Expectation Failed. El servidor no puede cumplir los requerimientos del header del request Expect.

5xx Errores del servidor

El servidor ha fallado al completar una solicitud aparentemente válida. Cuando los códigos de estado empiezan por 5 indica casos en los que el servidor sabe que tiene un error o realmente es incapaz de procesar el request. Salvo cuando se trata de un request HEAD, el servidor debe incluir una entidad conteniendo una explicación del error, y de si es temporal o permanente. Igualmente los user-agents deberán mostrar cualquier entidad al usuario. Estos códigos de respuesta se aplican a cualquier método request.

500 Internal Server Error. Error genérico, cuando se ha dado una condición no esperada y no se puede concretar el mensaje.

501 Not Implemented. El servidor o no reconoce el método del request o carece de la capacidad para completarlo. Normalmente es algo que se ofrecerá en el futuro, como un nuevo servicio de una API.

502 Bad Gateway. El server actuaba como puerta de entrada o proxy y recibió una respuesta inválida del servidor upstream.

503 Service Unavailable. El servidor está actualmente no disponible, ya sea por mantenimiento o por sobrecarga.

504 Gateway Timeout. El servidor estaba actuando como puerta de entrada o proxy y no recibió una respuesta oportuna por parte del servidor upstream.

505 HTTP Version Not Supported. El servidor no soporta la versión del protocolo HTTP usada en el request.

511 Network Authentication Required (RFC 6585). El cliente necesita autentificarse para poder acceder a la red.

AJAX

Ajax

AJAX son las siglas de Asynchronous JavaScript And XML, (Javascript asíncrono y XML).

La ventaja de ajax respecto a otros lenguajes de programación web es la asincronía. Esto consiste en que cuando queremos intercambiar datos con el servidor (por ejemplo enviar o comprobar un formulario, consultar una base de datos, etc), la página no se queda parada esperando la respuesta, sino que se pueden seguir ejecutando acciones mientras tanto.

Ventajas de usar Ajax

Mejor experiencia de usuario. Ajax permite que las páginas se modifiquen sin tener que volver a cargarse, dándole al usuario la sensación de que los cambios se producen instantáneamente. Este comportamiento es propio de los programas de escritorio a los que la mayoría de los usuarios están más acostumbrados. La experiencia se vuelve mucho más interactiva.

Optimización de recursos. Al no recargarse la página se reduce el tiempo implicado en cada transacción. También se utiliza menos ancho de banda.

Alta compatibilidad. Ajax es soportado por casi todas las plataformas Web.

Desventajas de usar Ajax

Problemas de acceso. Normalmente, si un usuario refina una consulta a una base de datos a través de muchos criterios (por ejemplo, categoría, precio, forma de pago, etc.), la página se recargará con una URL que reflejará los parámetros ingresados. El usuario puede guardar esa URL para volver a acceder a los resultados ya filtrados fácilmente. Pero con Ajax la URL no se modifica ante la consulta, por lo que deberemos volver a ingresar cada filtro manualmente cuando queramos recuperar los resultados deseados. Existen métodos para modificar este comportamiento, pero agregan dificultad al desarrollo y peso al sitio.

Problemas de SEO. Los buscadores tienen dificultades al analizar el código escrito en JavaScript. El hecho de que se no se generen nuevas URL elimina un importante factor de posicionamiento.

Dificultad. Las aplicaciones con Ajax suelen requerir de un mayor tiempo de desarrollo.