Trabajo Práctico N°3 – Arquitectura y Conectividad

Profesor: Jorge Morales

Alumno: Raul Jara

Ejercicio N°1:

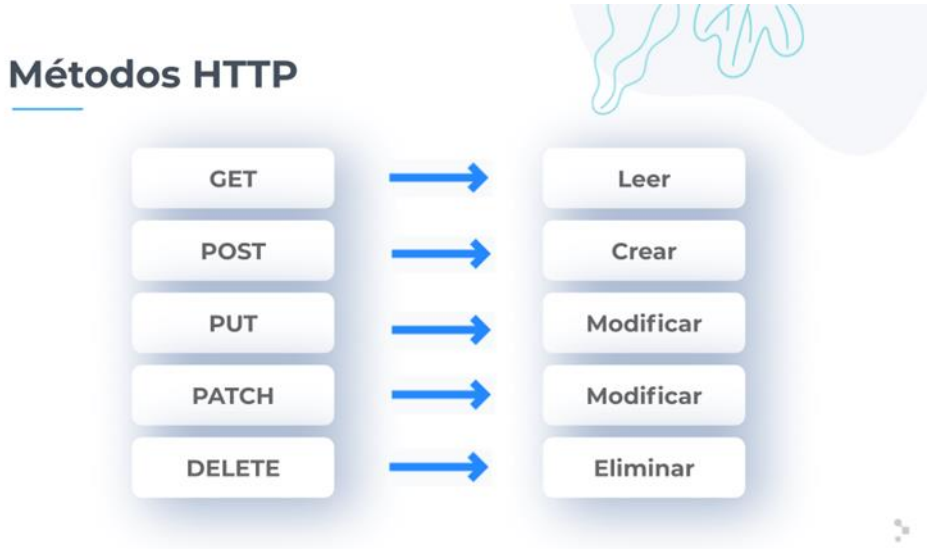
¿Cuáles son las Peticiones más comunes en el Protocolo HTTP? ¿Para qué se usan? Ejemplifique. (Indicar por lo menos 5).

Introducción

El protocolo HTTP (Hypertext Transfer Protocol) es la base de la comunicación en la web y en muchas aplicaciones IoT. Define un conjunto de métodos de solicitud que indican la acción deseada sobre un recurso específico.

Es un protocolo de aplicación que permite la comunicación entre clientes y servidores en la web. En el contexto del Internet de las Cosas (IoT), HTTP se utiliza comúnmente para intercambiar información entre dispositivos y plataformas en la nube. El protocolo define una serie de métodos o 'peticiones' que indican la acción que el cliente desea realizar sobre un recurso

A continuación, se describen las cinco peticiones más comunes en HTTP, sus usos y ejemplos prácticos.



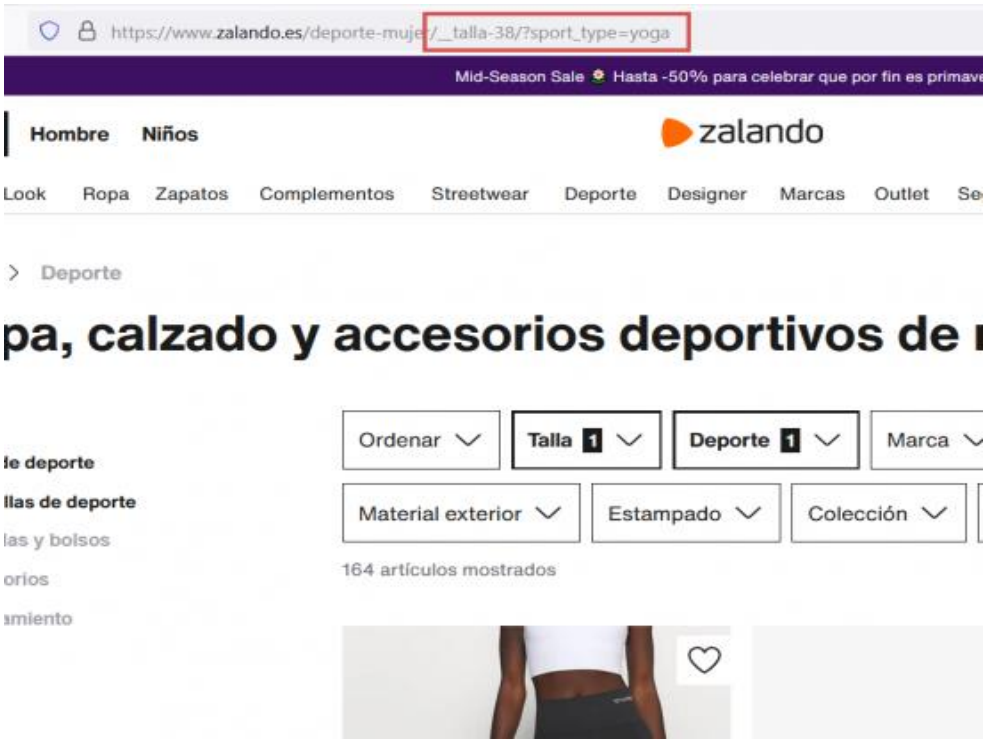
1. ***GET***

El método GET se utiliza para solicitar datos del servidor. Es el método más utilizado en HTTP y permite recuperar información sin realizar ningún cambio en el recurso solicitado.

Características:
- Es seguro: no modifica el estado del servidor.
- Es idempotente: realizar la misma solicitud múltiples veces produce el mismo resultado.
- Se puede almacenar en caché.

Ejemplo de uso en IoT: un sistema de monitoreo de clima realiza una solicitud GET para obtener los datos de temperatura actuales desde un sensor remoto.
Solicitud: GET /api/temperatura
Respuesta: {"temperatura": "22.5°C"}

Los parámetros GET siempre inician con un signo de interrogación `?`. A este le siguen el nombre de la variable y su valor correspondiente, separados por un `=`. Si una dirección contiene más de un parámetro, estos se separan por un “et”, conocido en inglés como ampersand `&`.



2. ***POST***

El método POST se utiliza para enviar datos al servidor con el fin de crear un nuevo recurso. Es comúnmente usado en formularios web y sistemas IoT para registrar datos nuevos.

Características:
- No es idempotente: múltiples solicitudes pueden crear múltiples recursos.
- Los datos se envían en el cuerpo de la solicitud.

Ejemplo de uso en IoT: enviar los datos de humedad de un sensor a un servidor central.
Solicitud: POST /api/humedad
Cuerpo: {"valor": "60%", "timestamp": "2025-04-13T10:00:00Z"}

Diferencias entre GET y Post

	GET	POST
Los parámetros se colocan dentro	URI	Cuerpo
Propósito	Recuperación de documentos	Actualización de datos
Resultados de consulta	Capaz de ser marcado como favorito.	No puede ser marcado como favorito
Seguridad	Vulnerable, como presente en texto plano	Más seguro que el método GET
Restricciones de tipo de datos de formulario	Solo se permiten caracteres ASCII.	No se permiten restricciones, incluso datos binarios.
Longitud de datos de formulario	Debe mantenerse lo más mínimo posible.	Podría estar en cualquier rango.
Visibilidad	Puede ser visto por cualquier persona.	No muestra variables en la URL.
Tamaño variable	Hasta 2000 caracteres.	Hasta 8 Mb
Almacenamiento en caché	Los datos del método se pueden almacenar en caché.	No almacena en caché los datos.

3. PUT

El método PUT se usa para actualizar completamente un recurso existente. A diferencia de POST, PUT reemplaza la representación actual del recurso por la proporcionada.

Características:
- Es idempotente: múltiples solicitudes idénticas producen el mismo efecto.
- Especifica completamente el nuevo estado del recurso.

Ejemplo de uso en IoT: actualizar la configuración de un sensor para cambiar su frecuencia de

muestreo.
Solicitud: PUT /api/sensores/1
Cuerpo: {"frecuencia": "15s", "ubicacion": "invernadero"}

El método PUT es usado para solicitar que el servidor almacene el cuerpo de la entidad en una ubicación específica dada por el URL.

Ejemplo:

Se solicita al servidor que guarde el cuerpo de la entidad dada en index.htm en la raíz del servidor:

```
PUT /index.htm HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.yosoy.dev
Accept-Language: es-mx
Connection: Keep-Alive
Content-type: text/html
Content-Length: 182
```

```
<html>
<body>
<h1>Howdy, Michelle!</h1>
</body>
</html>
```

Y el servidor por su parte, responde con lo siguiente al cliente (habiendo ya guardado el cuerpo de la entidad en e index.htm):

```
HTTP/1.1 201 Created
Date: Mon, 27 Nov 2017 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Content-type: text/html
```


4. PATCH

PATCH se utiliza para realizar una actualización parcial a un recurso existente. A diferencia de PUT, no reemplaza todo el recurso, sino que modifica solo los campos especificados.

Características:
- No es necesariamente idempotente.
- Es útil para cambios pequeños o específicos.

Ejemplo de uso en IoT: cambiar solo el nombre de un sensor sin afectar otras configuraciones.
Solicitud: PATCH /api/sensores/1
Cuerpo: {"nombre": "Sensor Interior"}

Respuesta HTTP

200	Respuesta exitosa con body
204	Responde exitosa sin body

Características

Petición con cuerpo	Si
Respuesta con cuerpo	Si
Seguro	No
Idempotente	No
Cacheable	No
Permitido en HTML forms	No

5. DELETE

El método DELETE se utiliza para eliminar un recurso específico en el servidor. Es directo y debe manejarse con cuidado, ya que puede borrar información crítica.

Características:
- Es idempotente: eliminar el mismo recurso varias veces no cambia el resultado.
- El recurso se especifica mediante su URI.

Ejemplo de uso en IoT: eliminar un dispositivo IoT obsoleto del sistema.

Solicitud: DELETE /api/dispositivos/1

ADEMÁS DE ESTOS MÉTODOS COMUNES, EXISTEN OTROS MÉTODOS HTTP, COMO:

HEAD: Similar al método GET, pero solo devuelve los encabezados HTTP, no el cuerpo de la respuesta.

OPTIONS: Devuelve los métodos HTTP admitidos por el servidor.

CONNECT: Establece un túnel TCP/IP con el servidor.

TRACE: Realiza un análisis de seguimiento de la solicitud.

PATCH: Actualiza parcialmente un recurso existente en un servidor.

Es importante mencionar que la seguridad es fundamental al utilizar estos métodos.

Debe considerarse la autenticación, la autorización y la validación de entradas para evitar ataques malintencionados.

Los métodos HTTP son fundamentales para la comunicación entre el cliente y el servidor en la web.

Cada método tiene una función específica y se utiliza para realizar operaciones diferentes en un servidor.

Estos métodos HTTP son fundamentales para la interacción entre clientes y servidores en aplicaciones web y sistemas IoT. Comprender su funcionamiento y diferencias es esencial para diseñar arquitecturas eficientes y seguras.