

Peticiones a servidores

FutureE

Repasemos el concepto de servidor

Un servidor es un equipo informático muy importante que se encarga de almacenar, suministrar y distribuir información para que usuarios puedan acceder a ella, las peticiones al servidor hacen uso del modelo cliente-servidor.

Un servidor es una computadora muy poderosa que se encarga de atender solicitudes de clientes, es decir otros computadores, que deseen consumir su información.



Comencemos por definir que es cliente

Un cliente hace referencia a aquel que realiza la petición de la información que se almacena en el servidor.

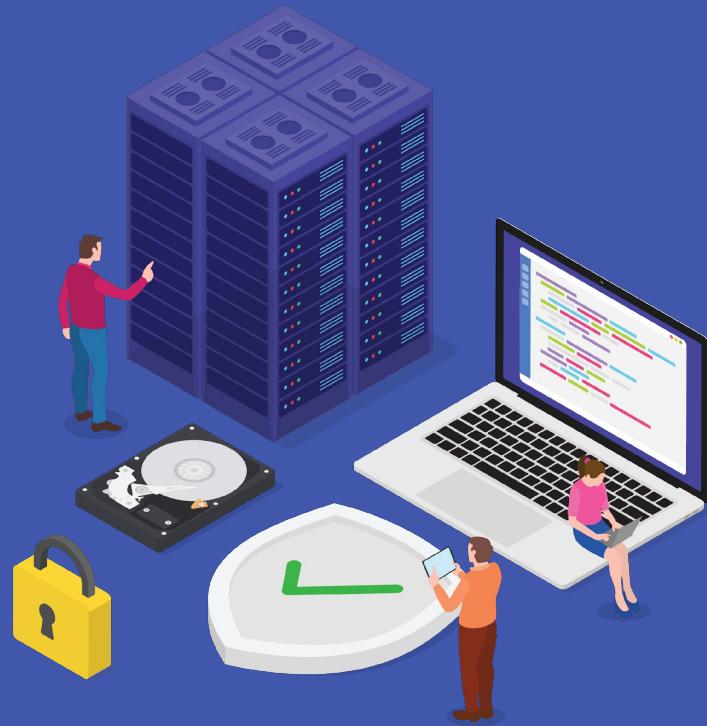
Algunos ejemplos de cliente puede ser cualquier dispositivo que te permita acceder a realizar consultas en la red como tu computadora, celular, tableta, impresora, escáner, entre otros, son equipos cliente que se conectan a servidores de internet para consumir los servicios de información que estos servidores proporcionan.



¿Y cómo funciona el modelo cliente servidor?

Clientes y servidores se comunican intercambiando mensajes. Los mensajes que envía el cliente, se llaman peticiones, y los mensajes enviados por el servidor se llaman respuestas.

El servidor recibe la petición que hizo el cliente, el servidor realiza el servicio y devuelve un resultado en forma de respuesta al cliente. Un servidor puede tratar múltiples peticiones de múltiples clientes al mismo tiempo.





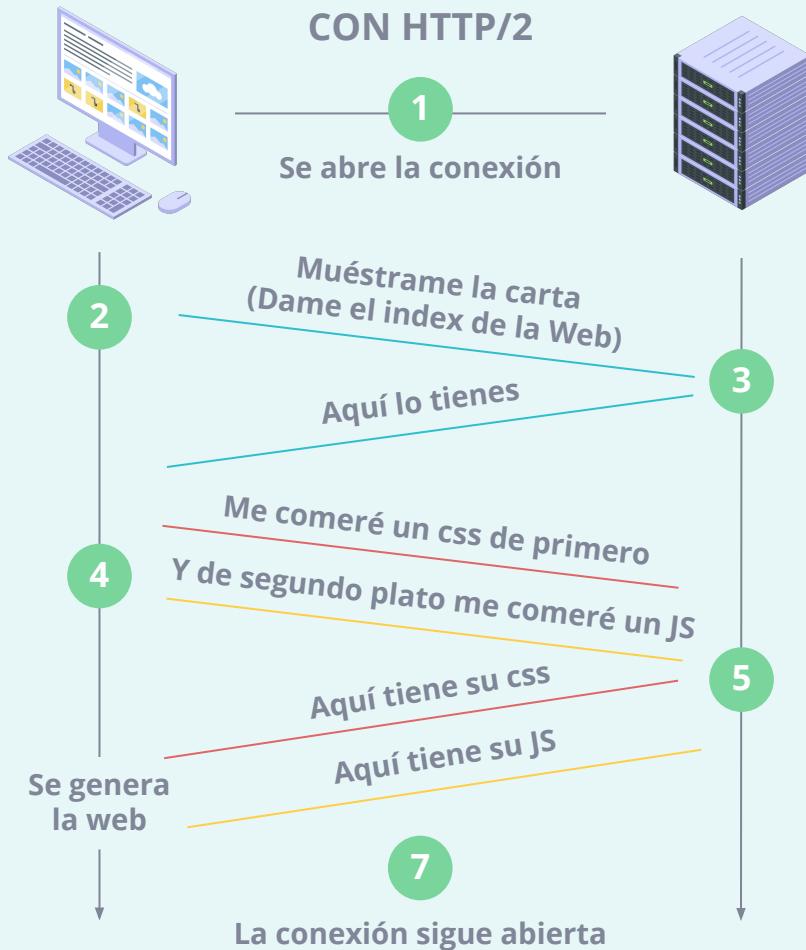
¿Cómo se establece la comunicación?

Como los humanos que establecemos comunicación entre nosotros siguiendo ciertos protocolos que establece nuestra humanidad, la comunicación entre computadoras se hace también por protocolos. Cada protocolo tiene un conjunto de reglas que ayudan a regular la comunicación entre sistemas que transmiten información.

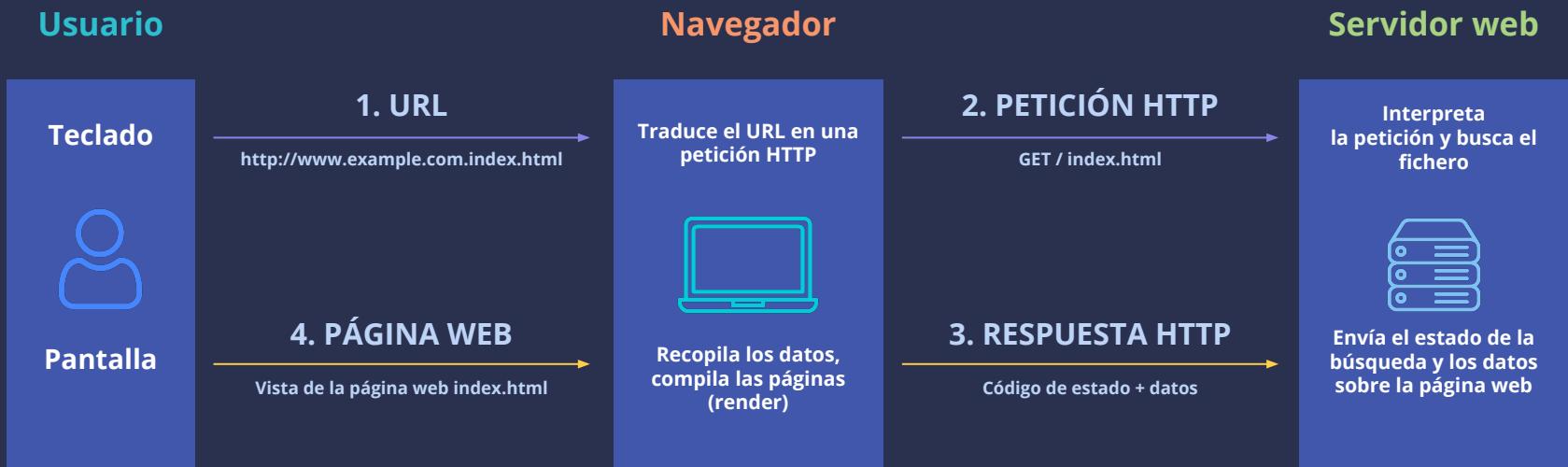
En el caso de equipo informáticos que deseen visualizar contenido desde su navegador, se utiliza un protocolo llamado HTTP.

¿Qué es el protocolo HTTP?

Por sus siglas en inglés *Hypertext Transfer Protocol* o en español Protocolo de Transferencia de Hipertexto, permite solicitar los recursos a un servidor, a través de este protocolo podemos llamar archivos almacenados en el servidor.

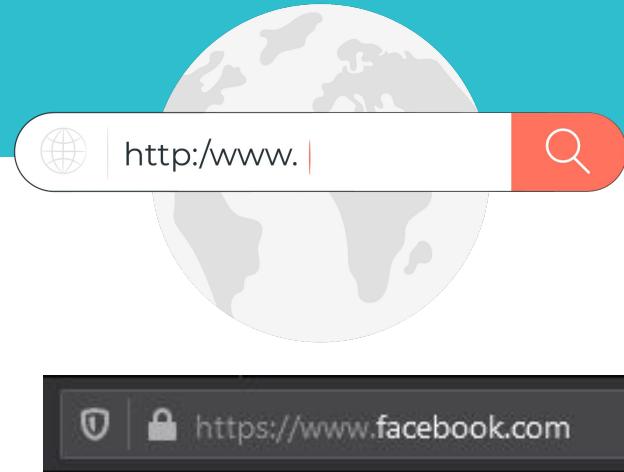


El proceso de comunicación según HTTP



¿Cómo identificar el protocolo?

Usualmente cuando en nuestro navegador (Firefox, Chrome, etc) escribimos la dirección de un sitio se complementa añadiendo el protocolo del sitio, dependiendo de la configuración del servidor, hay algunos servidores que por mejores prácticas y seguridad lo cambian por HTTPS. Por ejemplo, si intentas escribir <http://www.facebook.com> se cambiará por <http://www.facebook.com>

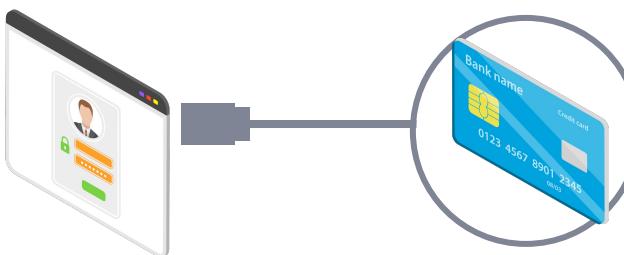


HTTPS es una evolución del protocolo HTTP, con la gran diferencia que es seguro, (la S viene de Secure/Seguro). Esto significa que la información transmitida será cifrada y que solo el cliente y el servidor podrán intercambiar y entender la información. Por lo que si hubiera una computadora espiando la información que se intercambia no podría entenderla.

A continuación ve el video HTTP.vs.HTTPS (<https://www.youtube.com/watch?v=xgxdAAMYY2k>) en donde verás cómo es posible ver la información intercambiada entre un servidor y un cliente cuando la comunicación es insegura.

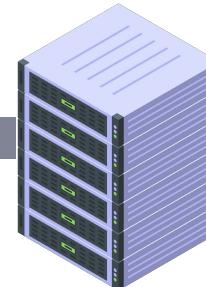
HTTP vs HTTPS

Usuario

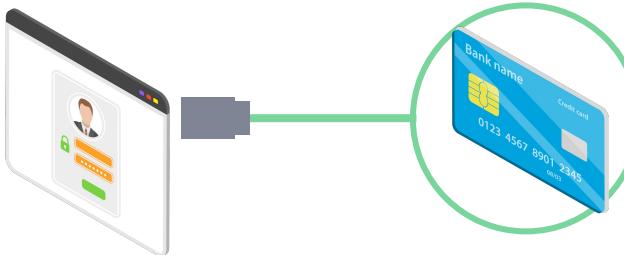


Conexión
Insegura

HTTP Normal (80)



Usuario



Conexión
Encriptada

HTTPS Seguro (443)



Certificado SSL

Características de HTTP para el control de elementos

Caché: Determina el tiempo que queremos almacenar los documentos descargados del servidor en el navegador web, como son imágenes, archivos HTML, Javascript y CSS. De esta manera cuando el usuario refresca la página web este no deberá esperar a que se descarguen nuevamente los documentos ya que se encuentran almacenados temporalmente en el navegador. Esto brindará una mejor experiencia al usuario al hacer una pagina mas rápida.

Características de HTTP para el control de elementos

Autenticación: Existen páginas que pueden estar protegidas de manera que solo usuarios autorizados pueden acceder.

Imaginemos que en una aplicación de un banco tenemos dos tipos de usuario; cajero y gerente, un cajero solo tiene acceso a consultar la información de clientes, mientras que el gerente puede consultar y modificar información de los clientes y cajeros.

Características de HTTP para el control de elementos

Proxies y tunneling: Facilitan el acceso al contenido, un proxy se puede almacenar en tu equipo o en cualquier otro lugar entre tu equipo y el servidor.



Características de HTTP para el control de elementos

Flexibilidad del requisito de origen

Para prevenir invasiones de la privacidad de los usuarios, los navegadores Web, solamente permiten a páginas del mismo origen, compartir la información o datos. Esto es una complicación para el servidor, así que mediante cabeceras HTTP, se puede flexibilizar o relajar esta división entre cliente y servidor.

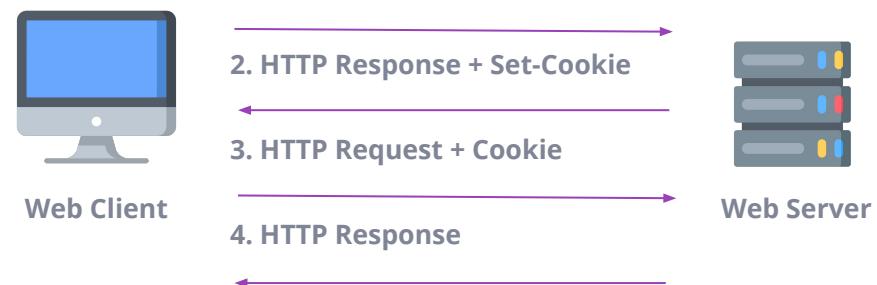
Características de HTTP para el control de elementos

Sesiones

Http cookies es un pequeño fragmento de datos, por ejemplo al acceder a un sitio los cookies recabar y almacenar datos mientras el usuario se encuentra en el sitio que un servidor envía a tu navegador web (Google Chrome, Firefox, Microsoft Edge).

El navegador puede almacenarlo y enviarlo de vuelta al mismo servidor. Normalmente, se usa para saber si dos solicitudes proceden del mismo navegador.

El uso de HTTP cookies permite relacionar peticiones con el estado del servidor.



Métodos HTTP para hacer peticiones al servidor

El protocolo HTTP posee varios métodos de solicitud que te serán muy útiles para realizar peticiones al servidor.

HTTP define un conjunto de métodos para indicar la acción que se desea realizar. En esta lección veremos los 3 métodos más utilizados. Cada uno posee una semántica diferente pero con características similares.



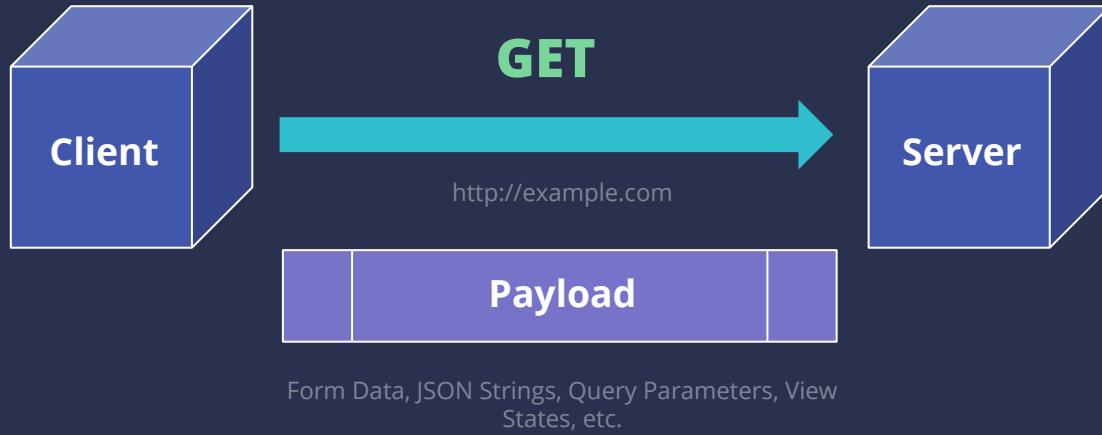


El método GET solicita una representación de un recurso específico.

Las peticiones que hacen uso de este método solo hacen solicitud de un recurso.

Por ejemplo cuando realizas una búsqueda en tu navegador, tu navegador se conecta con el servidor y hace envío de la petición GET.

La respuesta que el servidor enviará será la búsqueda que realizaste.



El método POST se utiliza para enviar paquetes de mayor tamaño, como imágenes o datos, los métodos POST suelen emplearse en formularios.

Por ejemplo cuando ingresas tus datos para registrarte a un sitio donde te pide que tengas una cuenta creada, tus datos se envían con el método POST y se almacenan.

HEAD

El método **HEAD** pide una respuesta idéntica a la de una petición GET, pero sin el cuerpo de la respuesta. **HEAD** se utiliza para solicitar que el servidor solo envíe el encabezado de la respuesta, sin el archivo. Esta alternativa es conveniente cuando se han de transferir archivos muy voluminosos, ya que, con esta petición, el cliente conoce primero el tamaño del archivo para luego poder decidir si acepta recibirla o no.

Por ejemplo cuando ingresas a un sitio de descarga de libros, en el sitio solo me muestra el enlace:

[downloads/libro.pdf](#)

Cuando ingresas al sitio el documento no se descarga a menos que se seleccione.

Si deseas ver un vídeo
te recomendamos ver:

¿Qué es y para qué sirve el protocolo HTTP?

<https://www.youtube.com/watch?v=VR0ImVct7YA>

¿Qué tipo de información se intercambia por HTTP/HTTPS?

Además de intercambiar documentos como **HTML, CSS y JavaScript**, es posible consumir otros archivos como los de tipo **JSON o XML**.

Bastante utilizados y estandarizados para consumir información de base de datos.

JSON en los servidores

Las respuestas que nos envía un servidor web comúnmente son objetos [JSON](#) o [XML](#) estos se utilizan como estándares, en la actualidad son más usados los objetos [JSON](#), este tipo de objetos contiene información que nos envía el servidor para ser mostrada en la interfaz del usuario o para determinar el flujo de nuestro código.

¿Qué es un objeto?

Un objeto en programación es un conjunto de información. Por ejemplo: el siguiente cuadro contiene información relacionada a una persona. Esto lo podemos traducir en un objeto tipo JSON.

Nombre: Ana Montes
Edad: 20 años
Nacionalidad: Mexicana
Sexo: Femenino

← Objeto persona

Objeto JSON:
{
 “nombre”: “Ana Montes”,
 “edad”: 20,
 “nacionalidad”: “Mexicana”,
 “Sexo”: “F”
}

¿Qué son los objetos JSON?

Es probable que el objeto **JSON** te pudiera haber recordado a los arreglos en JavaScript, y en parte es por que se inspiran en ello, **JSON significa: JavaScript Object Notation (JSON)**, es decir que utiliza la notación de objetos de JavaScript.

Este es un formato ligero de intercambio de datos, que resulta sencillo de leer y escribir y simple de interpretar y generar para las máquinas.

Pese a su nombre, no solo es parte de JavaScript, en realidad es un estándar basado en texto plano para el intercambio de datos, por lo que se usa en muchos sistemas que requieren mostrar o enviar información para ser interpretada por otros sistemas.

Ejemplo creado por nosotros:

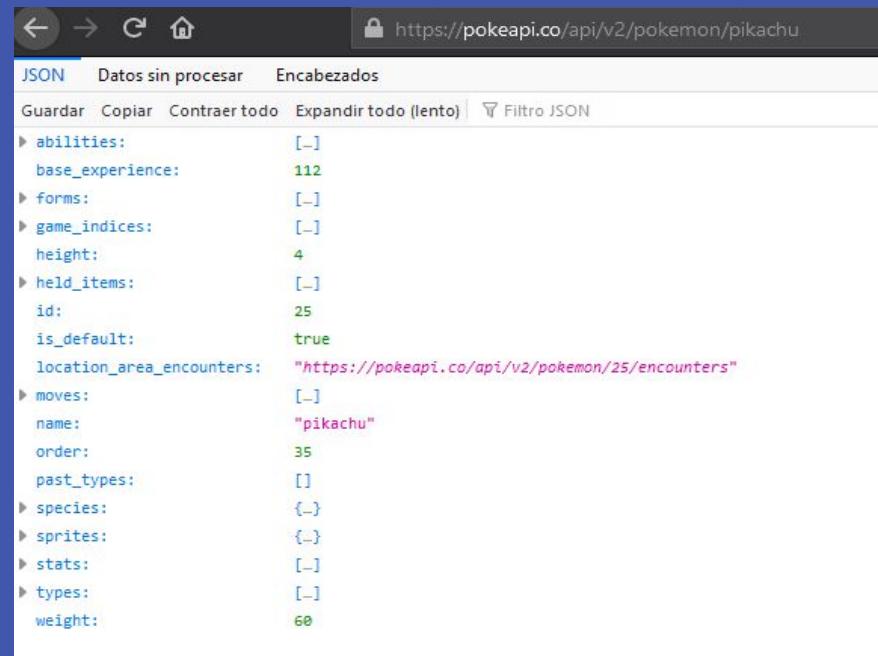
Puedes crear un archivo que tenga la información mostrada en el lado derecho, lo puedes guardar como datos.json

```
[  
  {"NombreEscuadron": "Archi escuadron",  
   "Ciudad": "Villa oliva",  
   "BaseSecreta": "Punta torre",  
   "miembros": [  
     {  
       "nombre": "Hombre molecula",  
       "edad": 29,  
       "poderes": [  
         "Resistencia a la radiación",  
         "Volviéndose diminuto",  
         "Explosión de radiación"  
       ]  
     },  
     {  
       "nombre": "Madame Uppercut",  
       "edad": 39,  
       "poderes": [  
         "Telequinesis",  
         "Resistencia al daño",  
         "Reflejos sobrehumanos"  
       ]  
     },  
     {  
       "nombre": "Llama eterna",  
       "edad": 1000000,  
       "poderes": [  
         "Inmortalidad",  
         "Teletransportación",  
         "Viajes interdimensionales"  
       ]  
     }  
   ]
```

Ejemplo real:

Para ver un ejemplo real de cómo luce un objeto JSON puedes probar visitando: <https://pokeapi.co/api/v2/pokemon/pikachu> (se recomienda que sea en Firefox para que lo acomodo de manera amigable).

Atentos, que esta API la volveremos a utilizar para en las siguientes lecciones. Puedes ver su documentación en <https://pokeapi.co/docs/v2>



The screenshot shows a browser window displaying the JSON response for Pikachu from the PokeAPI. The URL in the address bar is <https://pokeapi.co/api/v2/pokemon/pikachu>. The page has tabs for 'JSON', 'Datos sin procesar' (Raw data), and 'Encabezados' (Headers). Below the tabs are buttons for 'Guardar', 'Copiar', 'Contraer todo', 'Expandir todo (lento)', and a 'Filtro JSON'. The main content area shows the JSON structure with collapsible sections for abilities, base_experience, forms, game_indices, height, held_items, id, is_default, location_area_encounters, moves, name, order, past_types, species, sprites, stats, types, and weight. The value for base_experience is 112, and the value for weight is 60.

```
abilities: []
base_experience: 112
forms: []
game_indices: []
height: 4
held_items: []
id: 25
is_default: true
location_area_encounters: "https://pokeapi.co/api/v2/pokemon/25/encounters"
moves: []
name: "pikachu"
order: 35
past_types: []
species: {}
sprites: {}
stats: []
types: []
weight: 60
```

Ahora ya sabes que existen diversos tipos de peticiones a un servidor, que el protocolo HTTP se compone por métodos que nos ayudan a hacer estas peticiones y que las respuestas que nos envían los servidores normalmente vendrán en objetos JSON.

Estudia bien estos conceptos, puesto que los utilizaremos en el siguiente material.