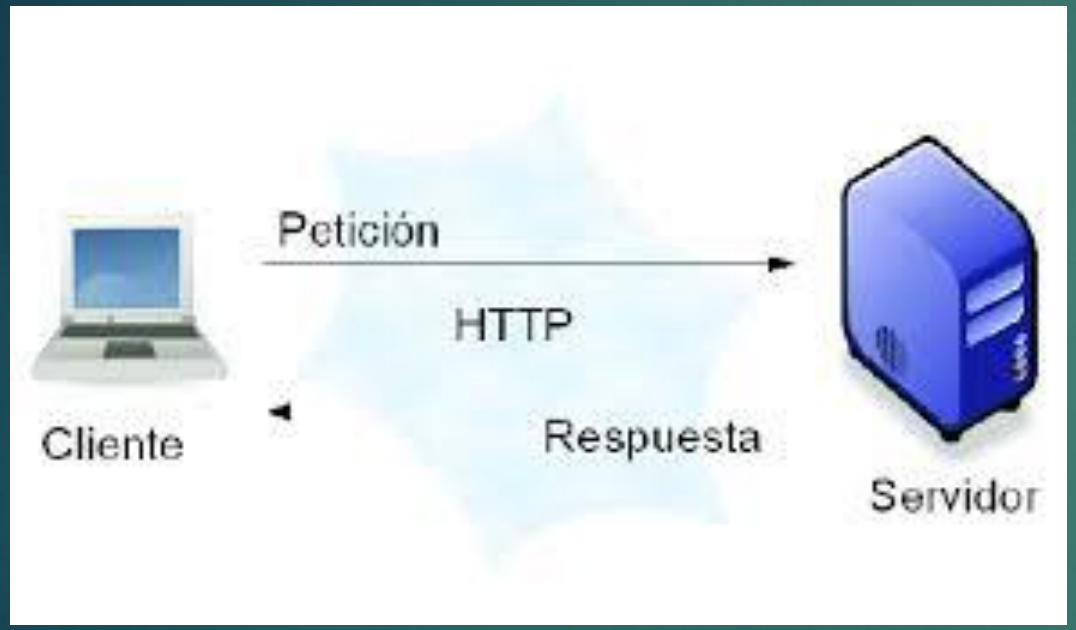
1.1APLICACIONES WEB

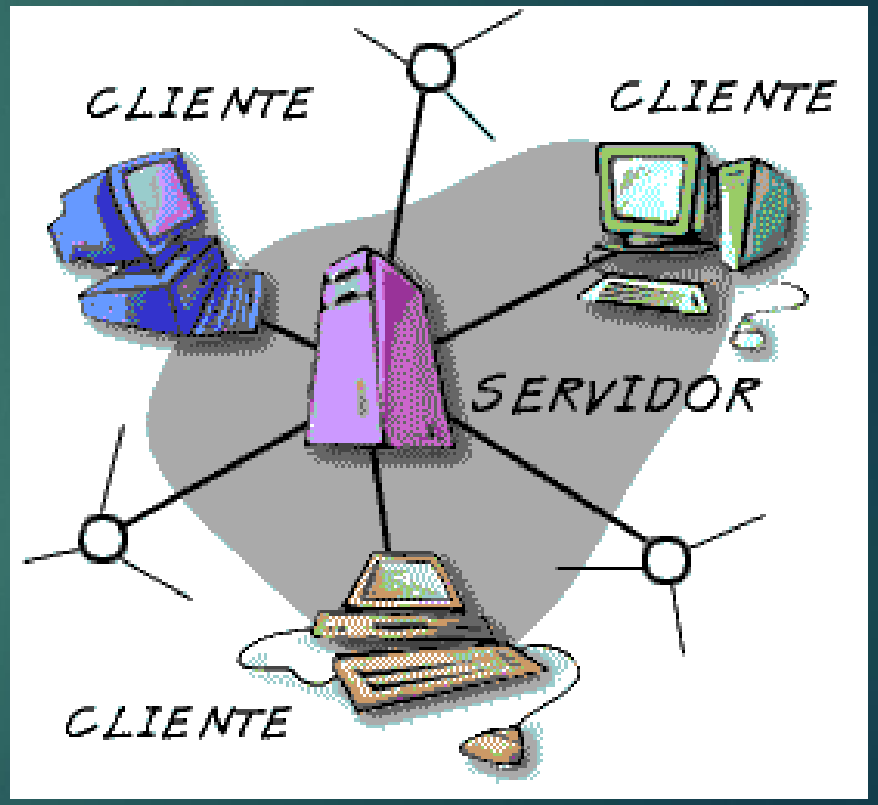
1.1.1 Modelo Cliente-Servidor

- En sus inicios y en algunos casos de uso actual, los sistemas web son archivos que los usuarios descargan con sus navegadores desde ordenadores remotos. Cuando un usuario decide acceder a un sitio web, le comunica al navegador la dirección del sitio y el programa descarga los archivos, procesa su contenido y lo muestra en pantalla.



- Debido a que los sitios webs son de acceso público e Internet es una red global, estos archivos deben estar siempre disponibles.
- Por este motivo, los sitios web no se almacenan en ordenadores personales, sino en ordenadores especializados diseñados para despachar estos archivos a los usuarios que los solicitan.
- El ordenador que almacena los archivos y datos de un sitio web se llama *servidor* y el ordenador que accede a esta información se llama *cliente*.

- Los servidores son muy similares a los ordenadores personales, con la diferencia de que están continuamente conectados a la red y ejecutando programas que les permiten responder a las solicitudes de los usuarios, sin importar cuándo se reciben o de donde proceden.

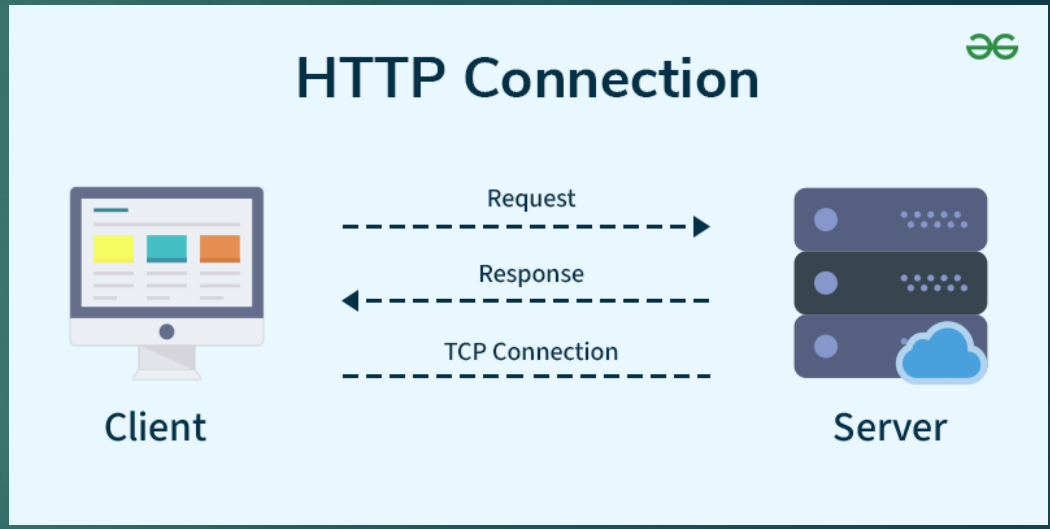


- Los programas más populares para servidores son Apache, para sistemas Linux, e IIS (Internet Information Server), creado por Microsoft para sistemas Windows.
- Entre otras funciones, estos programas son responsables de establecer la conexión entre el cliente y el servidor, controlar el acceso de los usuarios, administrar los archivos, y despachar los documentos y recursos requeridos por los clientes.

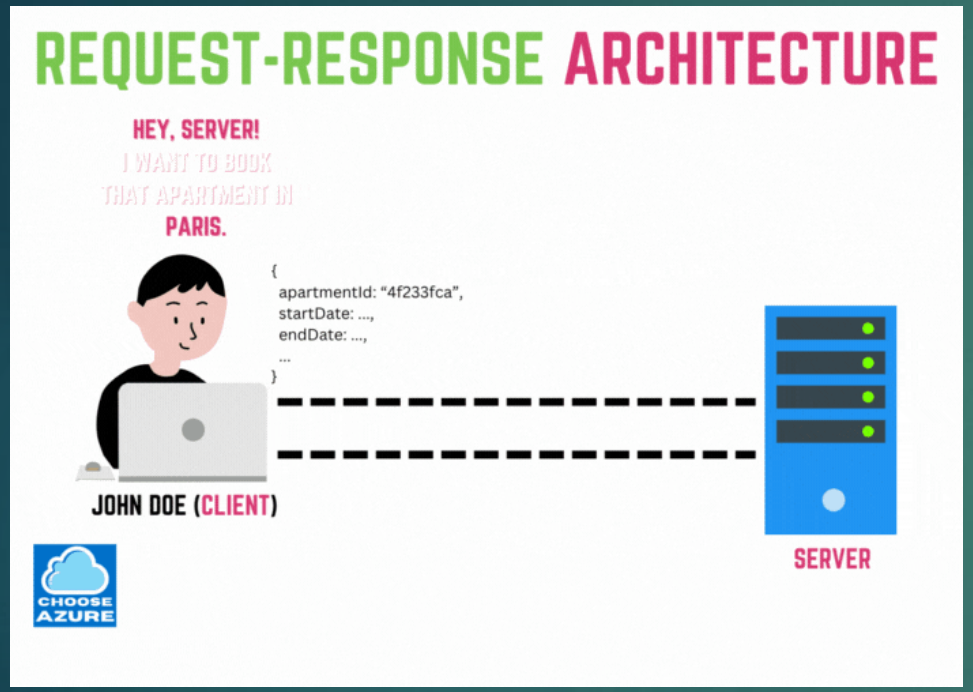
REQUEST

Una solicitud o *request* efectuada en una comunicación que utiliza el protocolo HTTP cuenta con una serie de partes, donde cada una de ellas cumple con una funcionalidad diferente respecto a la transmisión del mensaje que se desea dar a conocer desde el cliente hacia el servidor:

- Método
- URL
- Header o cabecera
- Body o cuerpo



Respuesta, *RESPONSE*

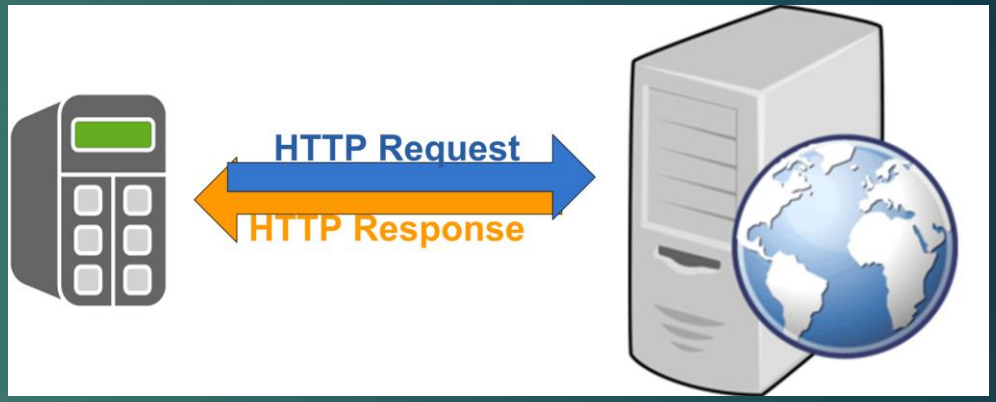


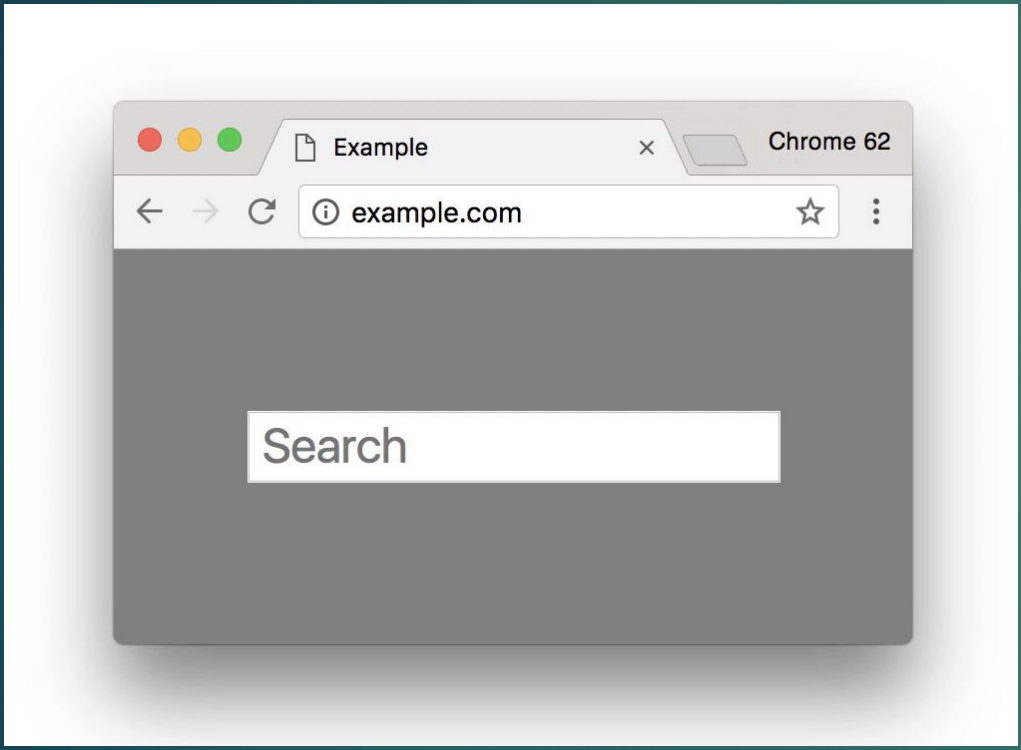
Estos tienen un formato particular que les permiten transportar la información necesaria para atender las solicitudes recibidas.

- Status codes
- Header o cabecera
- Body o cuerpo

1.1.2 Protocolo HTTP y HTTPS

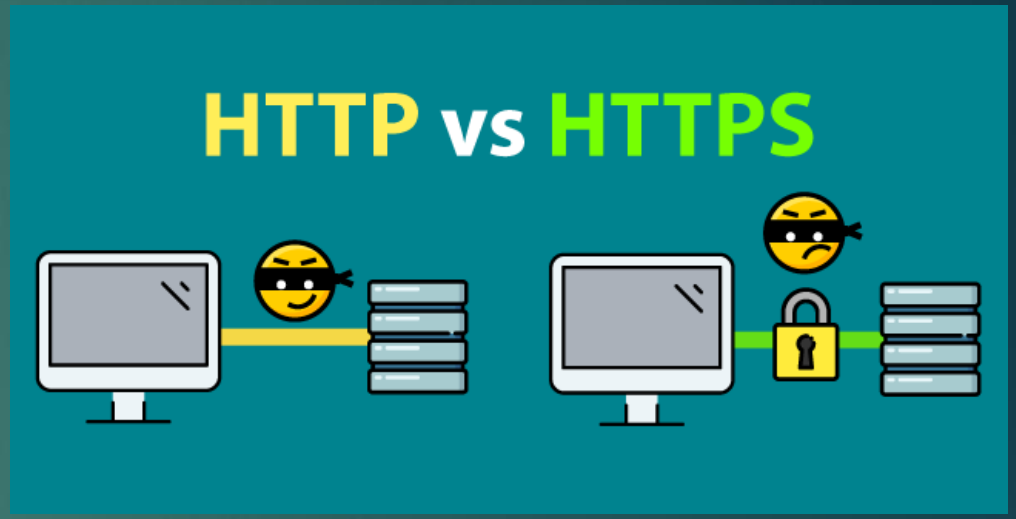
- El Protocolo de Transferencia de Hipertexto (HTTP, HyperText Transfer Protocol), es el Protocolo de la capa de aplicación de la Web.
- HTTP se implementa mediante dos programas: un programa cliente y un programa servidor.
- Ambos programas que se ejecutan en sistemas terminales diferentes, se comunican entre sí intercambiando mensajes HTTP.
- HTTP define la estructura de estos mensajes.

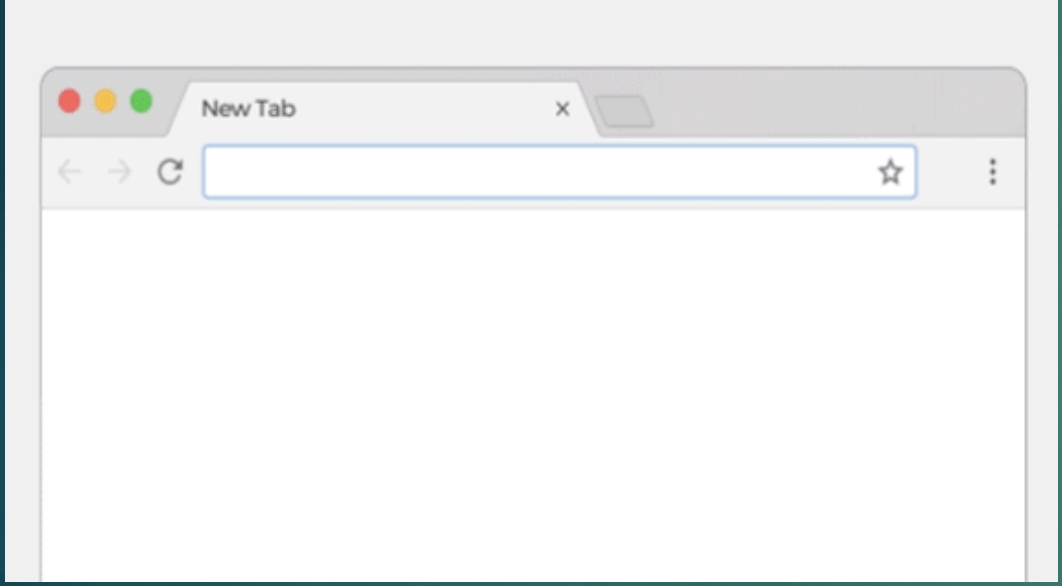




- Se trata del protocolo encargado de hacer posible el envío y recepción de información desde un servidor, hasta nuestros equipos de cómputo mediante el navegador de páginas web.

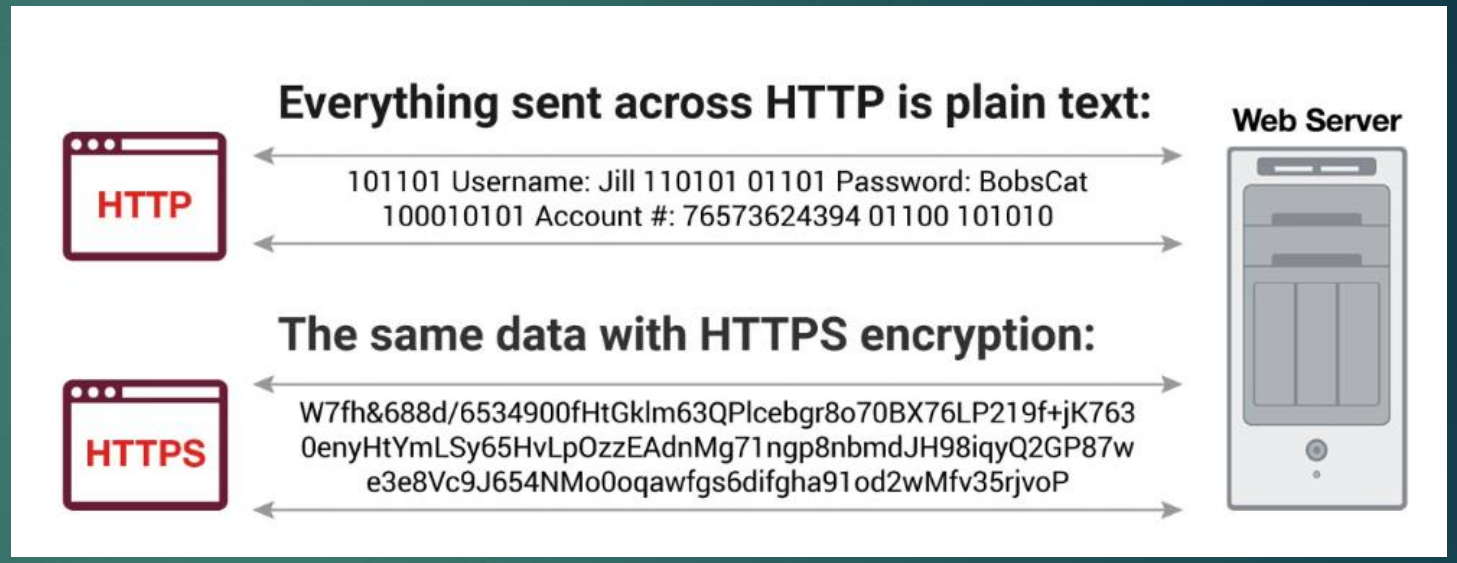
- HTTP es un protocolo involucrado en la comunicación entre los clientes y servidores web, a través de un sistema de peticiones y respuestas.
- Sin embargo, esta transacción de información corre el riesgo de ser interceptada y vista por terceros desde que sale del servidor y entra a los dispositivos, por este motivo, se hace obligatorio el uso del protocolo HTTPS.
- De esta manera, aunque la información sea interceptada, está protegida porque al estar encriptada se trata de un conjunto de caracteres incomprensibles y algoritmos indescifrables.





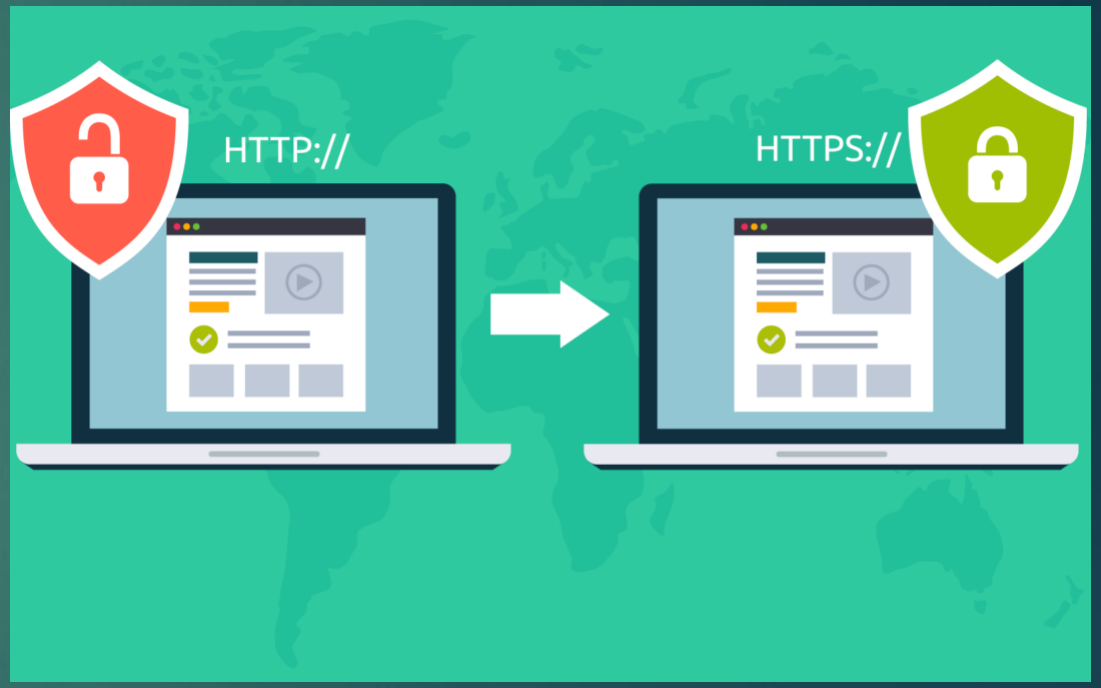
- HTTPS s la versión segura de HTTP, con la diferencia fundamental de que los datos de la transacción entre el cliente y el servidor están cifrados.
- El funcionamiento del protocolo HTTPS no difiere demasiado del HTTP, incluso, su diferencia fundamental es que el primero promueve una conexión segura a través de los certificados SSL o certificados HTTPS. Es decir, cuando entramos en una página web, esta le envía a nuestro navegador un comprobante de autenticidad validando que, en efecto, no habrá suplantación de identidad.

- El certificado recibido por el navegador contiene la clave necesaria para generar la conexión segura con el servidor y también para descifrar los datos luego de ser cifrados para su transferencia.



HTTP vs HTTPS: Principales diferencias

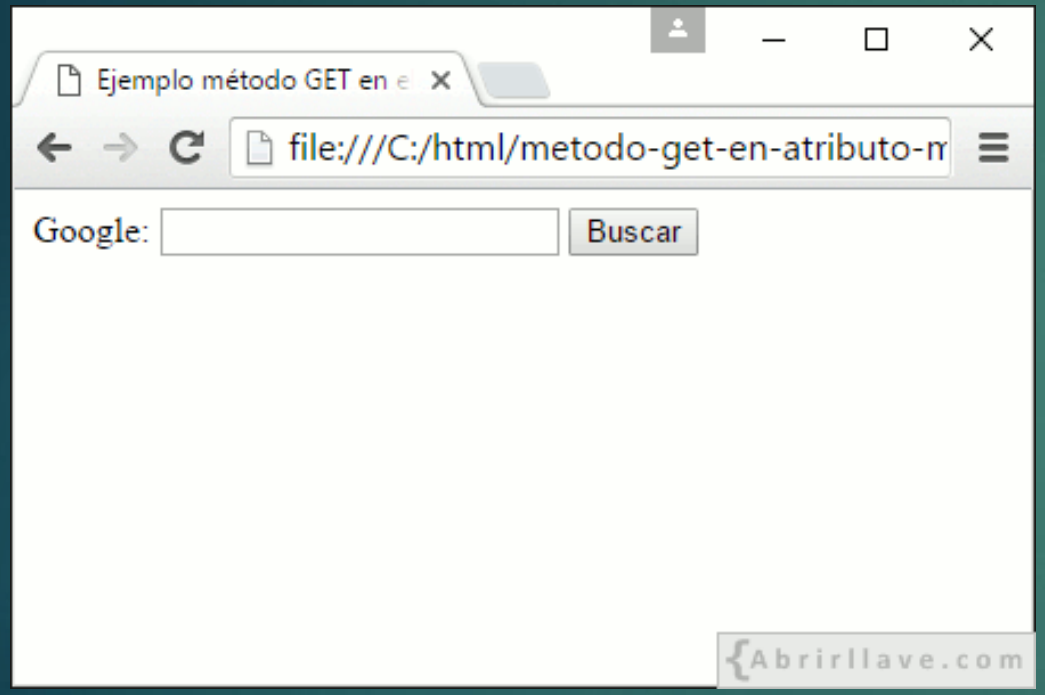
- **Seguridad del sitio.** La principal diferencia entre el protocolo HTTP y HTTPS se encuentra en los niveles de seguridad que ofrecen.
- **Validación del dominio.** Esto es posible gracias a los certificados SSL.
- **Transferencia de datos.** La transferencia de datos es el área de acción esencial del protocolo HTTP, sin embargo, su proceso no contempla hacerlo con seguridad. En su lugar, HTTPS trabaja con certificados SSL a fin de establecer un canal de comunicación segura, además del cifrado de los datos.



1.1.3 Métodos GET, POST, PUT, DELETE

- Los métodos HTTP son instrucciones estandarizadas que se utilizan para solicitar e interactuar con recursos en un servidor web. Cada método ordena al servidor que realice una acción específica, garantizando que la interacción entre el cliente y el servidor siga un protocolo conocido y consistente.

Método GET



Solicita datos de un recurso específico. Es seguro, idempotente y almacenable en caché, lo que lo hace ideal para consultar datos sin causar ningún efecto secundario.

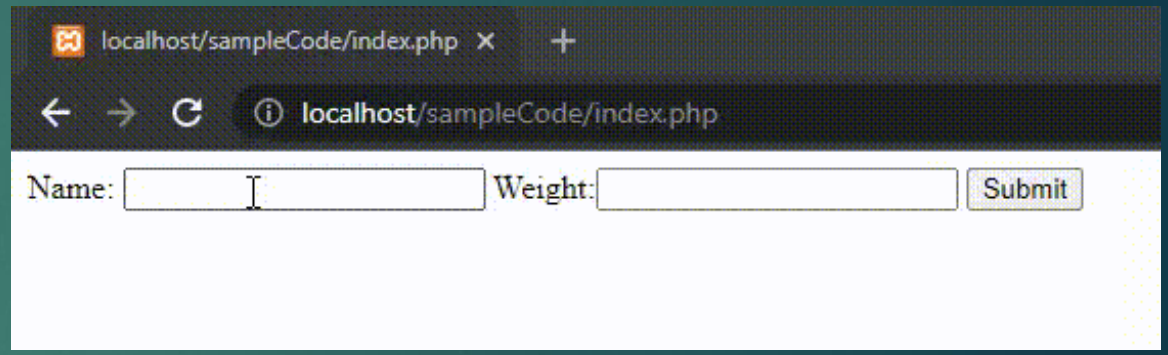
Importancia:

- **Recuperación de datos de solo lectura:** Se utiliza para recuperar datos sin modificarlos.
- **Capacidad de almacenamiento en caché:** Las respuestas a las solicitudes GET se pueden almacenar en caché, lo que mejora el rendimiento.
- **Seguridad e idempotencia:** Múltiples solicitudes idénticas no tienen ningún efecto adicional, lo que garantiza la estabilidad en operaciones repetidas.

Método POST

Envía datos al servidor, lo que a menudo da como resultado la creación de un nuevo recurso. A diferencia de GET, POST no es seguro ni idempotente.

- **Importancia:**
- **Envío de datos:** Se utiliza para enviar datos al servidor, como enviar datos de formularios.
- **Creación de recursos:** Crea nuevos recursos basados en los datos enviados.
- **No idempotencia:** Cada solicitud puede tener resultados diferentes, útiles para operaciones que necesitan registrar el estado.



Método PUT

Actualiza un recurso o lo crea si no existe. Este método es idempotente, lo que significa que múltiples solicitudes idénticas dan como resultado el mismo estado.

Importancia:

- **Actualización de recursos:** Ideal para actualizar recursos existentes.
- **Idempotencia:** Garantiza resultados consistentes con solicitudes repetidas.
- **Reemplazo completo:** A menudo reemplaza todo el recurso con los datos proporcionados.

Método DELETE

El método DELETE elimina un recurso específico del servidor.

Importancia:

- **Eliminación de recursos:** Elimina recursos del servidor.
- **Idempotencia:** Garantiza que el recurso se elimine una vez, independientemente de cuántas veces se realice la solicitud.
- **Gestión del estado:** Ayuda a gestionar el ciclo de vida de los recursos al permitir una eliminación limpia.