

# Capa de aplicación HTTP

# HTTP

Es un protocolo de transferencia de objetos (archivos).  
¿Qué tipo de archivos? Archivos HTML, scripts (javascript), archivos de estilo, imágenes, etc.

Sigue el esquema **cliente-servidor**: define la estructura de los mensajes que se intercambian entre ambos.

En particular, en las siguientes diapositivas consideraremos HTTP 1.0 y 1.1.

# Estructura de requerimiento

<Método> <Recurso> <HTTP vers.>

<Cabeceras>

<Cuerpo del mensaje>

El mensaje completo tiene una estructura basada en texto (ASCII).

# Estructura de requerimiento

**<Método> <Recurso> <HTTP vers.>**

<Cabeceras>

<Cuerpo del mensaje>

## Línea de requerimiento:

Es la primera línea de un requerimiento HTTP y debe estar presente siempre.

# Estructura de requerimiento

**<Método> <Recurso> <HTTP vers.>**

<Cabeceras>

<Cuerpo del mensaje>

**Línea de requerimiento:**

**<Método>**: verbo GET, POST, HEAD, etc

**<Recurso>**: nombre del archivo que se solicita

**<HTTP vers.>**: Indica la versión de HTTP 1.0 ó 1.1

# Estructura de respuesta

<HTTP vers.> <Código> <Frase>

<Cabeceras>

<Cuerpo del mensaje>

El mensaje completo tiene una estructura basada en texto (ASCII).

# Estructura de respuesta

<HTTP vers.> <Código> <Frase>

<Cabeceras>

<Cuerpo del mensaje>

## Línea de respuesta:

Es la primera línea de una respuesta HTTP y debe estar presente siempre.

# Estructura de respuesta

**<HTTP vers.> <Código> <Frase>**

<Cabeceras>

<Cuerpo del mensaje>

## Línea de respuesta:

**<HTTP vers.>**: Indica la versión de HTTP 1.0 ó 1.1

**<Código>**: indica un código de respuesta (1xx, 2xx, 3xx, 4xx, 5xx).

**<Frase>** : texto legible para los humanos sobre el código de respuesta.



# Cabeceras

<Línea de requerimiento> ó <Línea de respuesta>

**<Cabeceras>**

<Cuerpo del mensaje>

## Cabeceras:

Parámetros adicionales para un mensaje HTTP. Un mensaje puede llevar la cantidad de cabeceras que sean necesarias.

# Cabeceras

**User-Agent:** Permite especificar en un requerimiento que software, versión y/o sistema operativo usa el cliente.

HEAD / HTTP/1.1

Host: www.redes.unlp.edu.ar

**User-Agent: curl/7.74.0**

GET / HTTP/1.1

Host: www.redes.unlp.edu.ar

**User-Agent: Mozilla/5.0 (X11; Linux x86\_64;  
rv:91.0) Gecko/20100101 Firefox/91.0**

# Cabeceras

**Host:** indica el nombre del servidor y el número de puerto (por defecto puerto 80) al que se envía el requerimiento.

HEAD / HTTP/1.1

**Host: www.redes.unlp.edu.ar(:80)**

User-Agent: curl/7.74.0

GET / HTTP/1.1

**Host: www.redes.unlp.edu.ar(:80)**

User-Agent: Mozilla/5.0 (X11; Linux x86\_64;  
rv:91.0) Gecko/20100101 Firefox/91.0

# Cabeceras

**Host:** indica el nombre del servidor y el número de puerto (por defecto puerto 80) al que se envía el requerimiento.

HEAD / HTTP/1.0

**Host:** www.redes.unlp.edu.ar(:80)

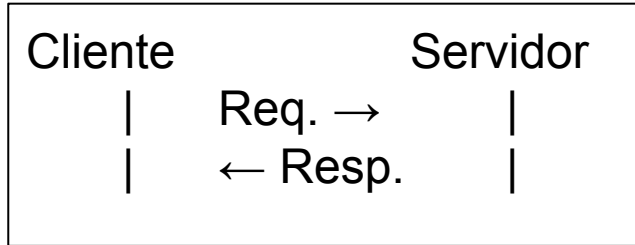
User-Agent: curl/7.74.0

**Obs.:** En HTTP 1.0 la cabecera Host es opcional, pero en 1.1 es obligatoria!

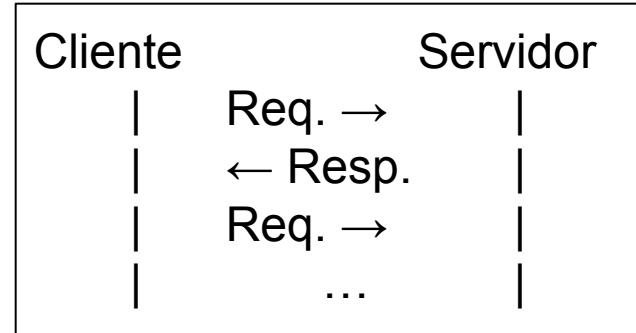
# Cabeceras

**Connection:** indica si la conexión se mantiene abierta o si se cierra al obtener la respuesta.

Connection: **close**



Connection: **keep-alive**



Al mantener la conexión abierta es posible realizar más de un intercambio (para obtener otros recursos).

# Cabeceras

**Connection:** indica si la conexión se mantiene abierta o si se cierra al obtener la respuesta.

## Requerimiento

HEAD / HTTP/1.0

Host: www.redes.unlp.edu.ar

User-Agent: curl/7.74.0

## Respuesta

HTTP/1.1 200 OK

Server: Apache/2.4.56 (Unix)

**Connection: close**

Content-Type: text/html

En HTTP 1.0 las conexiones son “no persistentes por defecto”.

**Obs.:** El servidor responde utilizando HTTP 1.1 que es retrocompatible con HTTP 1.0.

# Cabeceras

**Connection:** indica si la conexión se mantiene abierta o si se cierra al obtener la respuesta.

## Requerimiento

HEAD / HTTP/1.0

Host: www.redes.unlp.edu.ar

User-Agent: curl/7.74.0

**Connection: keep-alive**

## Respuesta

HTTP/1.1 200 OK

Server: Apache/2.4.56 (Unix)

**Connection: keep-alive**

Content-Type: text/html

Pero si se se indica, es posible utilizar conexiones persistentes, siempre y cuando el servidor lo soporte.

# Cabeceras

**Connection:** indica si la conexión se mantiene abierta o si se cierra al obtener la respuesta.

## Requerimiento

HEAD / HTTP/1.0

Host: www.redes.unlp.edu.ar

User-Agent: curl/7.74.0

**Connection: keep-alive**

## Respuesta

HTTP/1.0 200 OK

Server: Apache/2.4.56 (Unix)

Content-Type: text/html

En HTTP 1.0 el servidor podría no reconocer la cabecera, por lo que no la incluye en la respuesta.



# Cabeceras

**Connection:** indica si la conexión se mantiene abierta o si se cierra al obtener la respuesta.

## Requerimiento

HEAD / HTTP/1.1

Host: www.redes.unlp.edu.ar

User-Agent: curl/7.74.0

## Respuesta

HTTP/1.1 200 OK

Server: Apache/2.4.56 (Unix)

**Connection: keep-alive**

Content-Type: text/html

En HTTP 1.1 por defecto las conexiones son persistentes.

# Cabeceras

Otras cabeceras: If-modified-since, Etag, Authorization, Content-Length, Transfer-Encoding...

HTTP Headers: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>

# Cuerpo del mensaje

<Línea de requerimiento> ó <Línea de respuesta>

<Cabeceras>

**<Cuerpo del mensaje>**

**Cuerpo del mensaje:**

Contiene información útil (archivos, datos). ¿Siempre lleva carga útil?

# GET vs POST

En los requerimientos:

- GET: el cuerpo del mensaje no lleva información.
- POST: el cuerpo del mensaje lleva información.

# GET vs POST

En los requerimientos:

- GET: el cuerpo del mensaje no lleva información.
- POST: el cuerpo del mensaje lleva información.

¿Es posible enviar información con el método GET? ¿Se envía en el cuerpo?

# Uso de diferentes clientes

¿Cambia el formato de los mensajes de requerimiento según el cliente que se utilice (curl, navegadores web)?

# Uso de diferentes clientes

¿Cambia el formato de los mensajes de requerimiento según el cliente que se utilice (curl, navegadores web)?

- No, los clientes son herramientas que nos permiten generar requerimientos HTTP respetando los estándares del protocolo.
- Lo que cambia son los mecanismos que puede proveer cada cliente. Por ejemplo: web caché privada, redirecciones, requerimientos adicionales cuando se necesita más de un recurso.