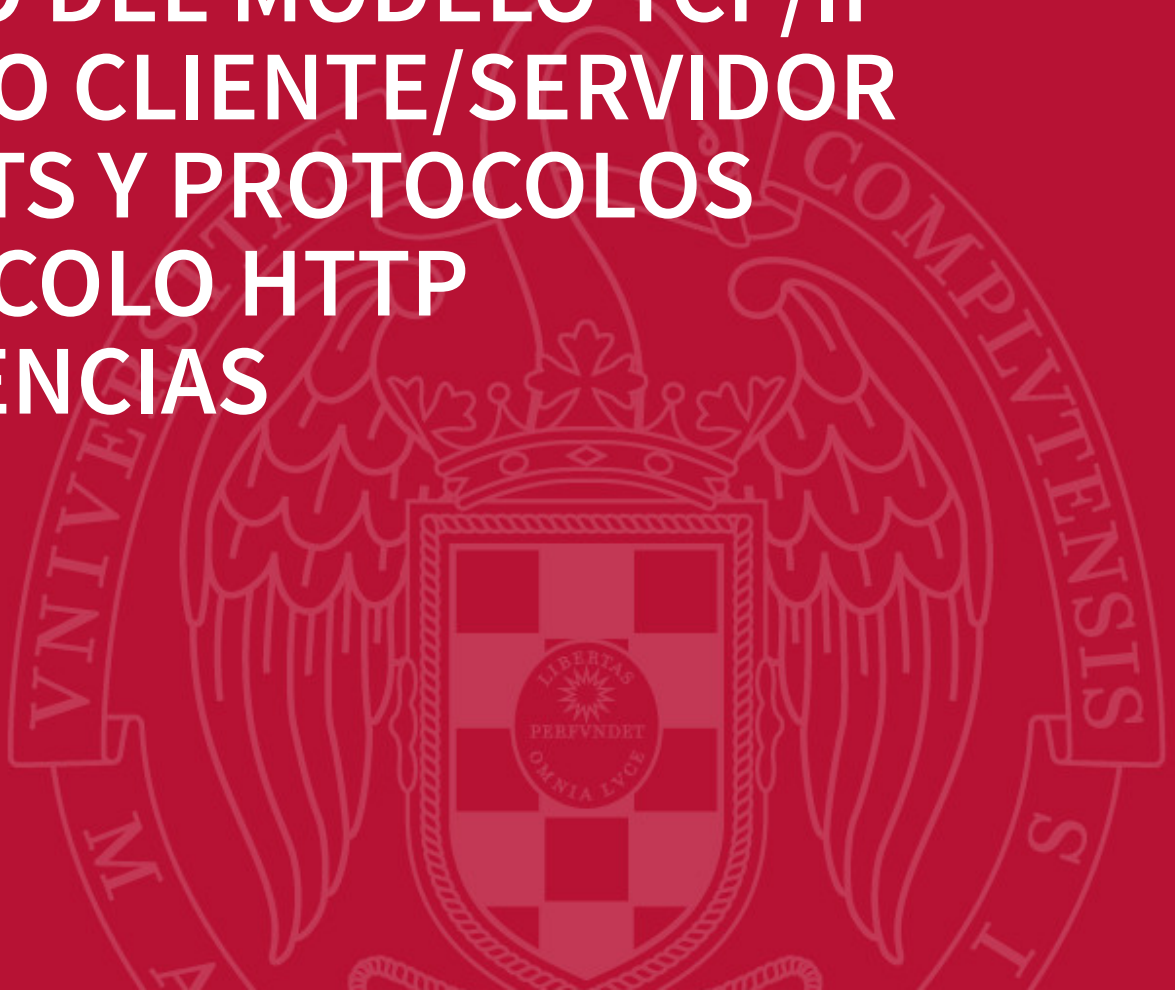
TEMA 1

INTRODUCCIÓN A LAS TECNOLOGÍAS WEB: EL PROTOCOLO HTTP

APLICACIONES WEB - GIS - CURSO 2019/20

Marina de la Cruz [marina.cruz@.ucm.es]
Dpto de Sistemas Informáticos y Computación
Facultad de Informática
Universidad Complutense de Madrid

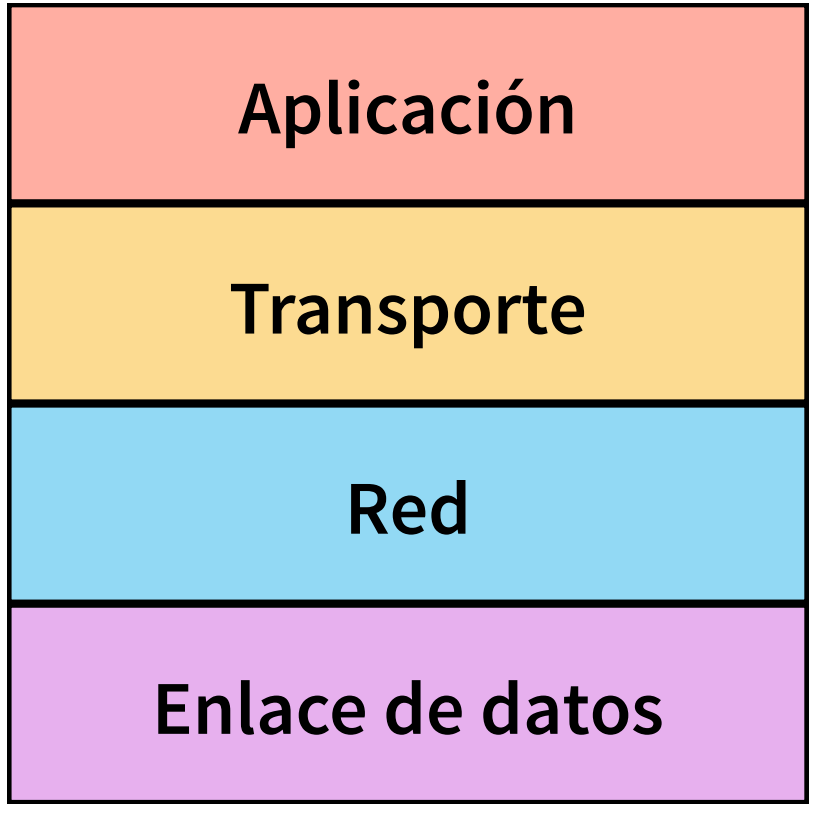
1. REPASO DEL MODELO TCP/IP
2. MODELO CLIENTE/SERVIDOR
3. SOCKETS Y PROTOCOLOS
4. PROTOCOLO HTTP
5. REFERENCIAS



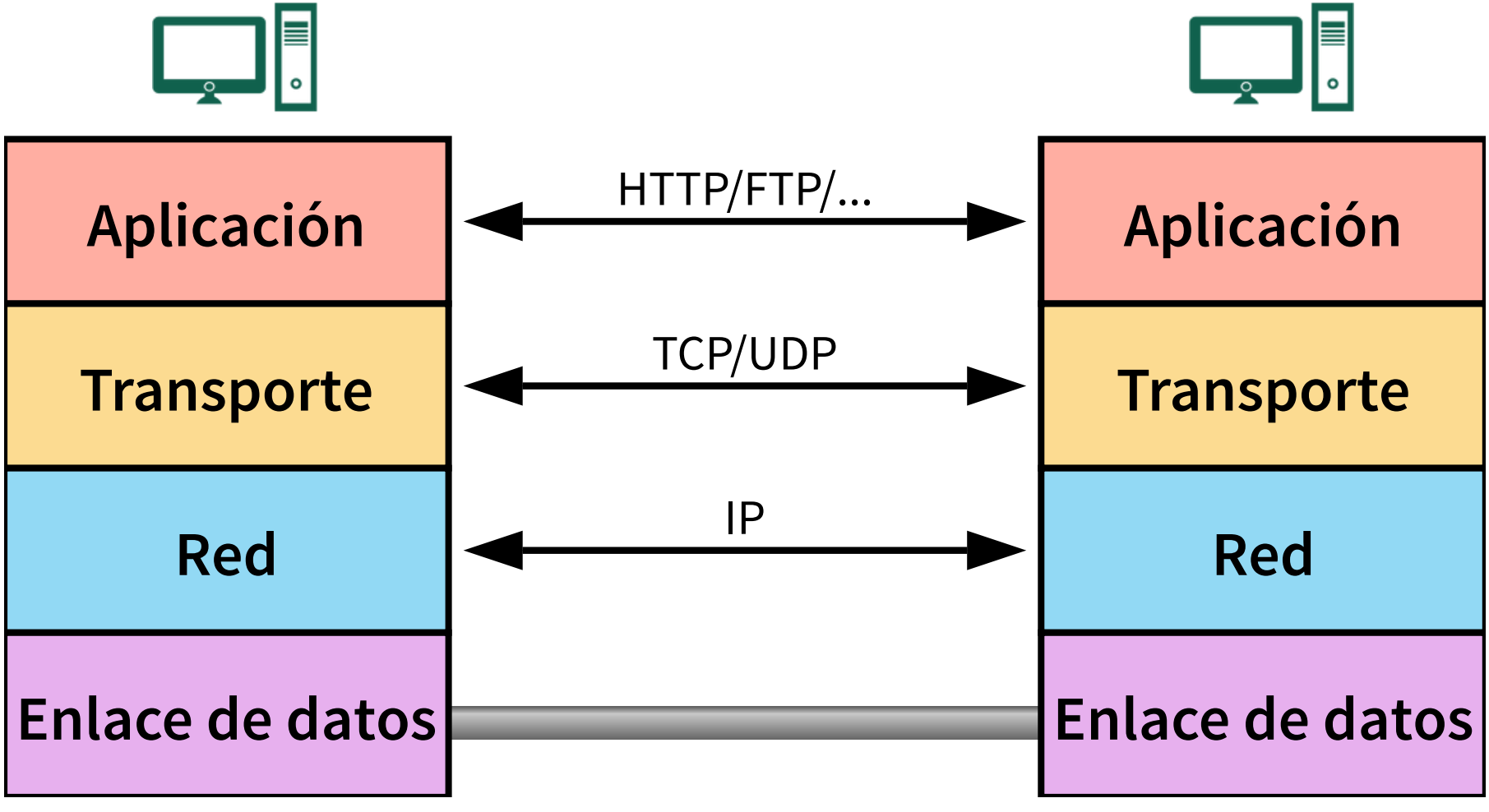
MODELO TCP/IP

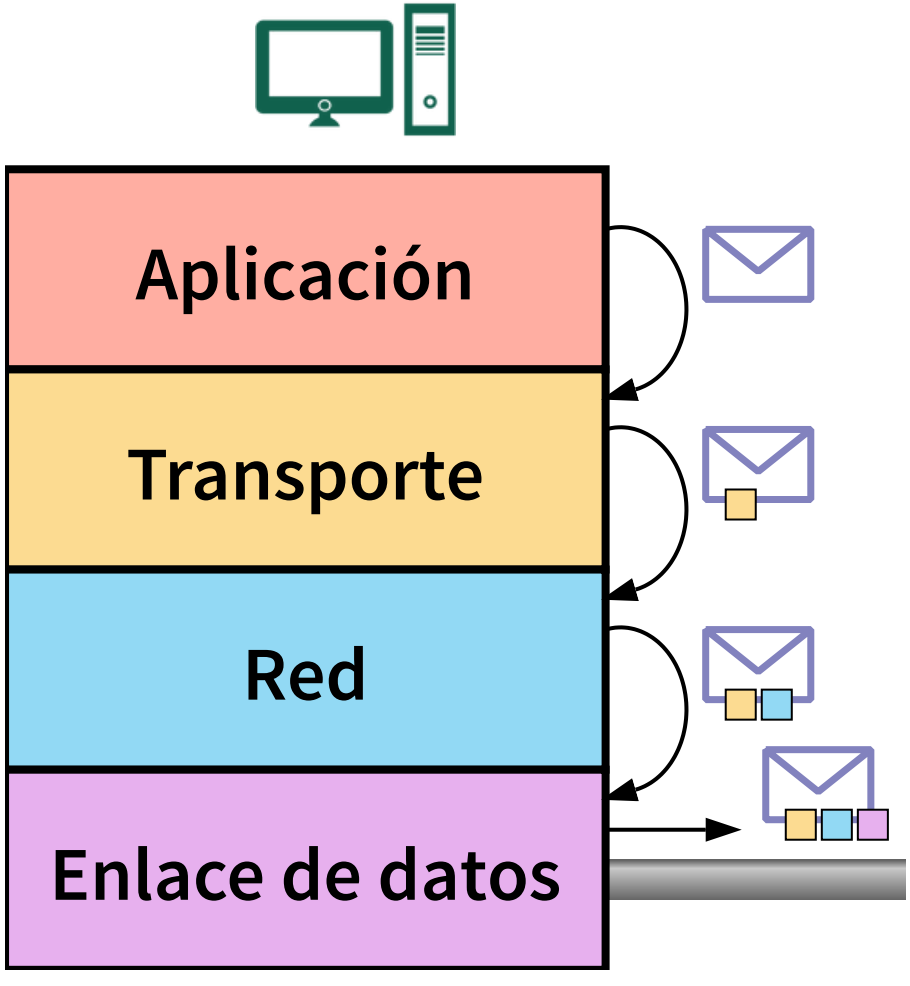
Basado en cuatro capas

Cada capa representa un nivel de abstracción



Cada capa se comunica con su homóloga en otro nodo distinto, utilizando un *protocolo* como lenguaje.





Sin embargo, la comunicación entre capas homólogas no es directa.

Cada capa solicita a su capa inferior que se envíe el mensaje al destino.

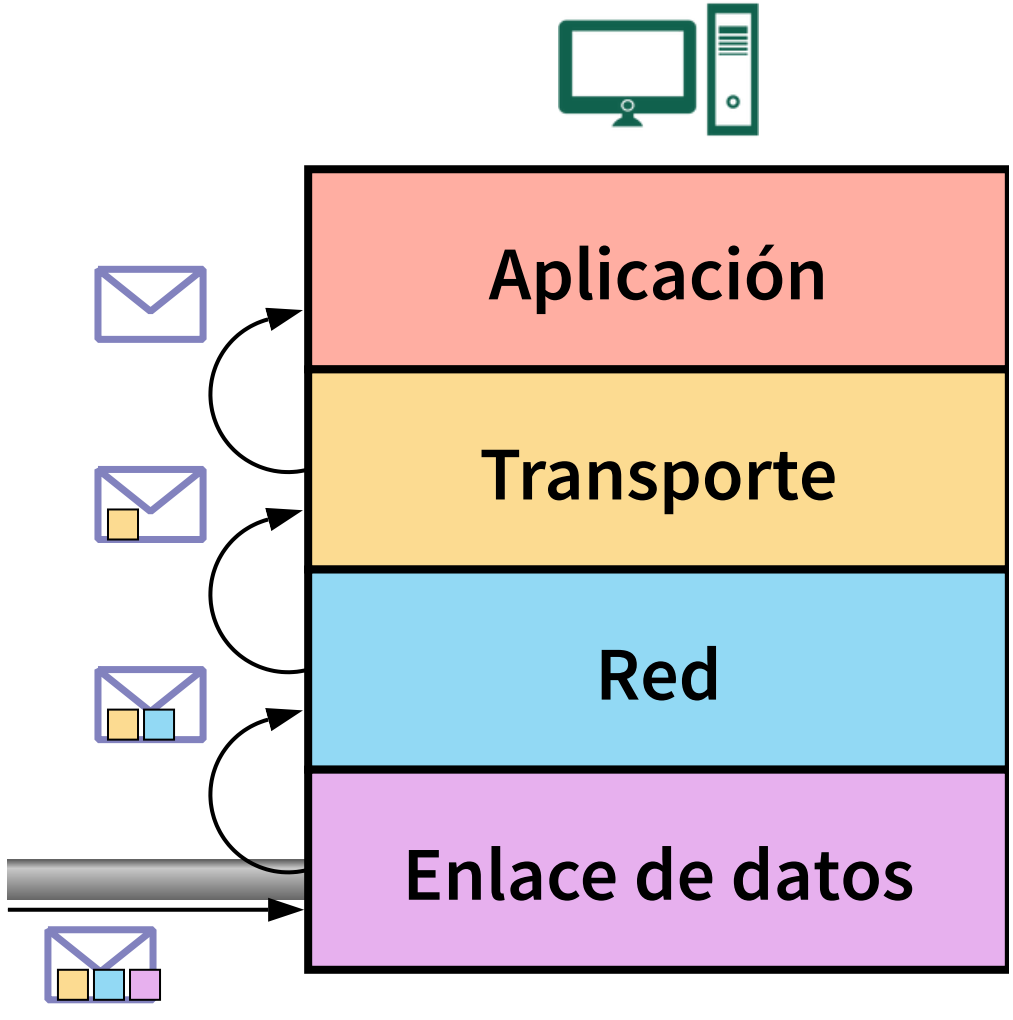
Cada capa extiende el mensaje a enviar con información propia que le permite realizar su cometido

Finalmente, la capa de enlace es la que realiza el envío físico

El nodo receptor recibe el mensaje a través de la capa de enlace.

Cada capa interpreta la información adicional introducida por su capa homóloga durante el envío, y acaba remitiendo el mensaje original (sin esta información adicional) a la capa superior.

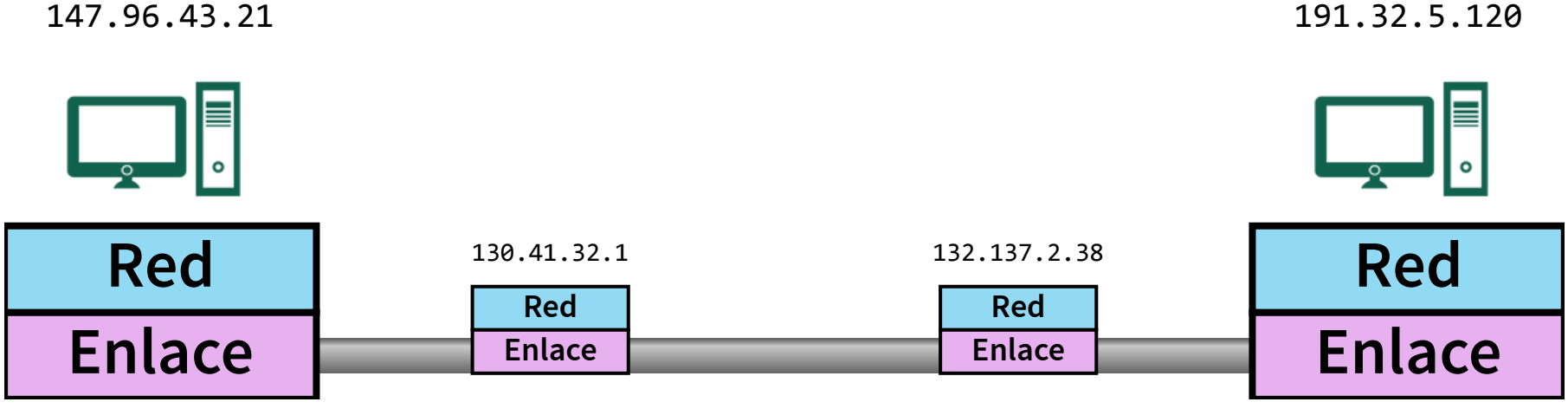
Finalmente, la capa de aplicación recibe el mensaje tal y como lo envió la capa de aplicación en el origen.



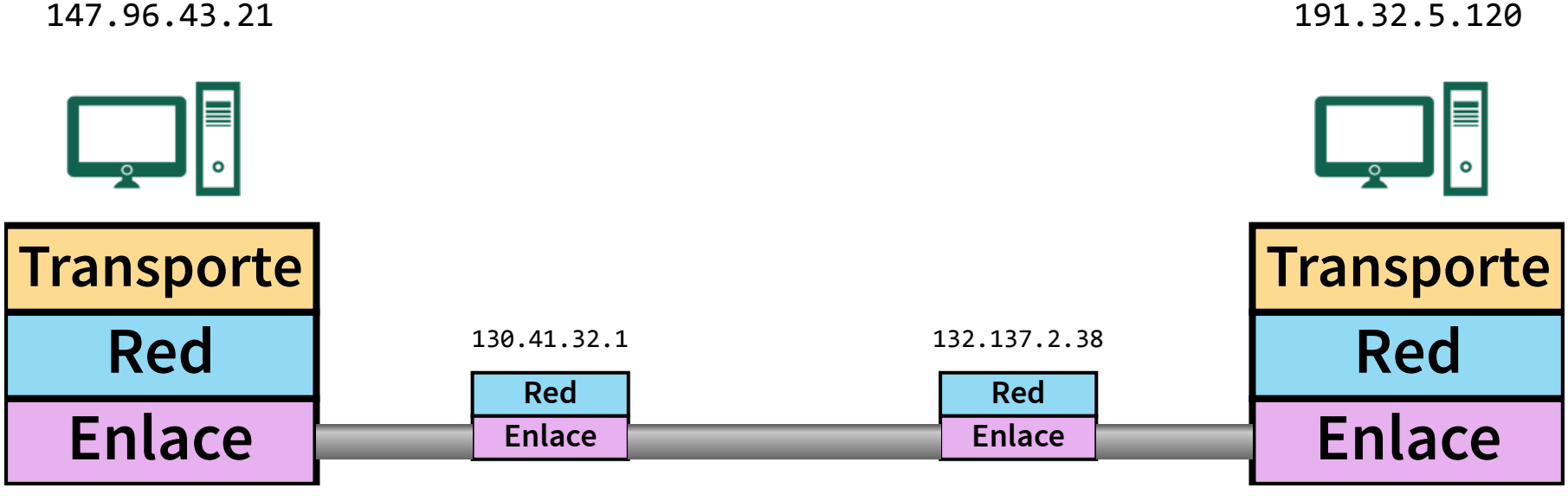
La capa de *enlace de datos* proporciona una vía de transmisión de paquetes de datos entre dos nodos enlazados directamente

La capa de *red* gestiona el envío de mensajes a través de múltiples nodos intermedios.
Cada nodo está identificado mediante una *dirección IP*

Las capas de red de los nodos intervinientes *enrutan* los paquetes desde el origen hasta el destino



La capa de *transporte* se encarga de la transmisión sin errores desde origen a destino.
Segmentación en paquetes, recepción en orden correcto...



CAPA DE TRANSPORTE

Protocolos:

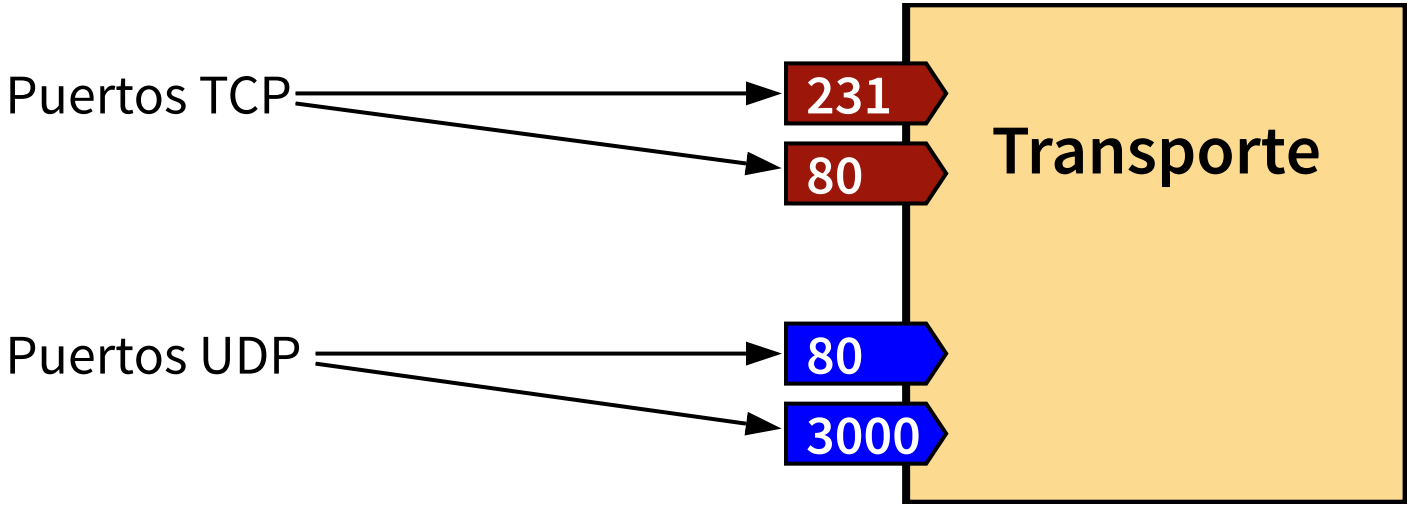
- **TCP**: orientado a conexión, bidireccional, fiable.
- **UDP**: no conexión, no comprobación de paquetes duplicados, extraviados, etc.

La capa de transporte permite varios canales de comunicación entre nodos.

Cada canal está identificado por un *número de puerto*.

Existen *puertos TCP* y *puertos UDP*.

EJEMPLO



PUERTOS TCP/UDP MÁS COMUNES

- **Puerto 20**: FTP (transmisión de archivos).
- **Puerto 22**: SSH (transmisión de archivos segura).
- **Puerto 25**: SMTP (envío de correo).
- **Puerto 80**: HTTP (web).
- **Puerto 110**: POP3 (recepción correo).
- **Puerto 143**: IMAP (recepción correo).
- **Puerto 443**: HTTPS (web segura).

Ver: List of TCP and UDP port numbers (Wikipedia)

1. REPASO DEL MODELO TCP/IP
2. MODELO CLIENTE/SERVIDOR
3. SOCKETS Y PROTOCOLOS
4. PROTOCOLO HTTP
5. REFERENCIAS

MODELO CLIENTE/SERVIDOR

Supongamos que dos nodos quieren conectarse para intercambiar información

¿Cómo y cuándo se establece esta conexión?

En el modelo *cliente/servidor* cada nodo adquiere un rol:

- El *servidor* permanece conectado a la espera de información que llegue por un determinado puerto. Se dice que el servidor *escucha* en dicho puerto.
- El *cliente* inicia la comunicación enviando información al servidor.

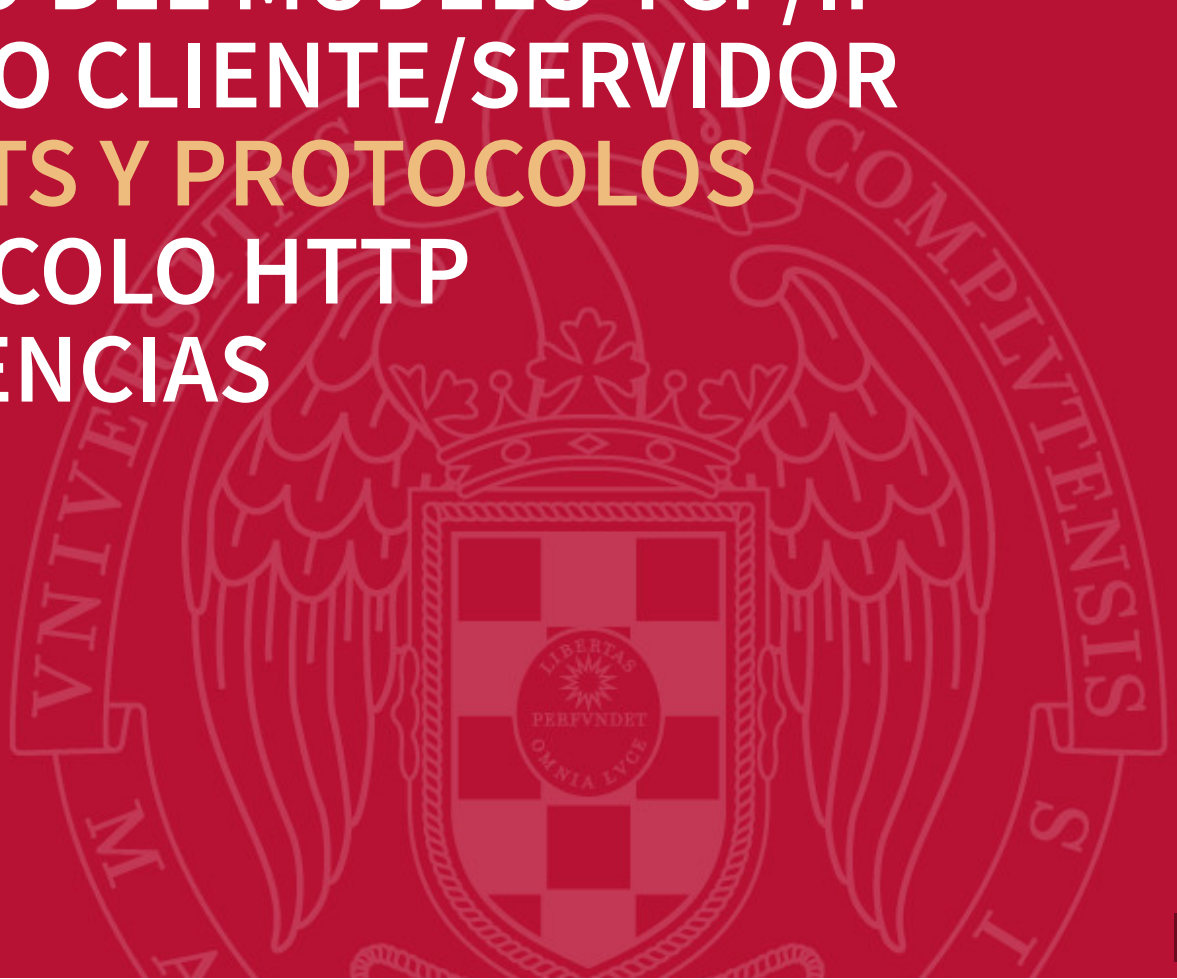
SERVIDOR

- Pasivo: espera a que un cliente envíe una petición
- Responde (sirve) las peticiones de varios clientes
- A cada tipo de petición se le llama *servicio*

CLIENTE

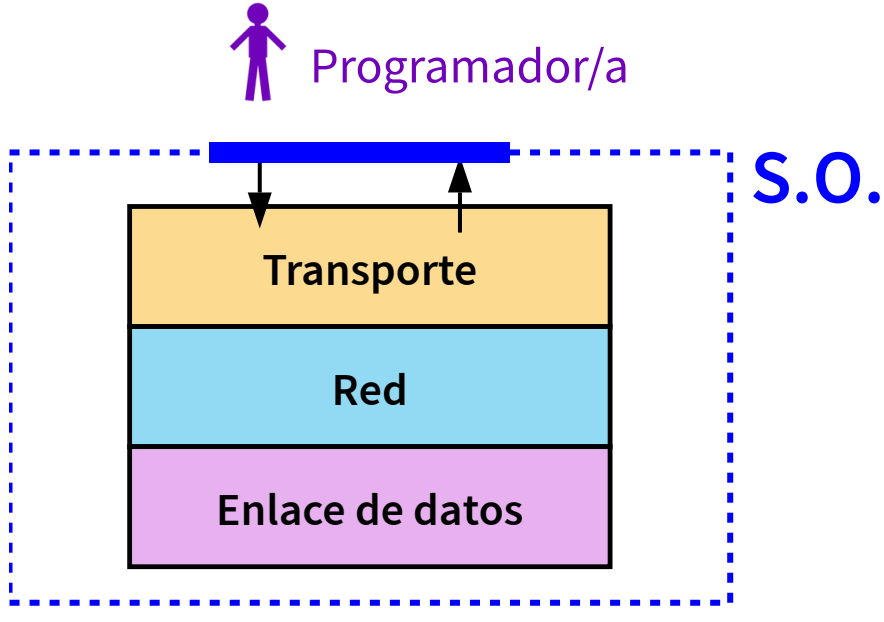
- Activo: toma la iniciativa en la comunicación con el servidor
- Ha de conocer la dirección IP del servidor, y el puerto al que enviar la información (petición)

1. REPASO DEL MODELO TCP/IP
2. MODELO CLIENTE/SERVIDOR
3. SOCKETS Y PROTOCOLOS
4. PROTOCOLO HTTP
5. REFERENCIAS



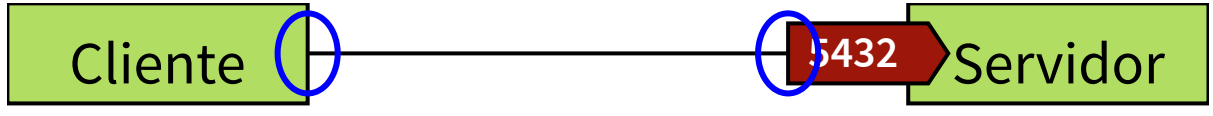
SOCKETS

El sistema operativo proporciona un mecanismo de acceso a la capa de transporte.



Este mecanismo se basa en el uso de *sockets*.

Supongamos que un servidor está escuchando en un determinado puerto y que un cliente se conecta al mismo:



Esto crea un canal de comunicación bidireccional entre cliente y servidor.

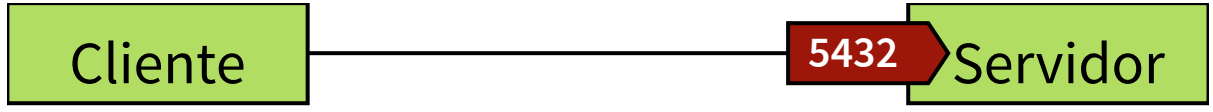
A cada uno de los extremos de este canal de comunicación se le llama *socket*

Cada socket tiene asociados dos flujos:

- **Flujo de entrada** (*input stream*)
  La información que llegue desde el otro extremo se lee a través de este flujo.

- **Flujo de salida** (*output stream*)
  Todo lo que se escriba en este flujo es enviado al otro extremo.

El programador lee y escribe en estos flujos como si de ficheros se tratase



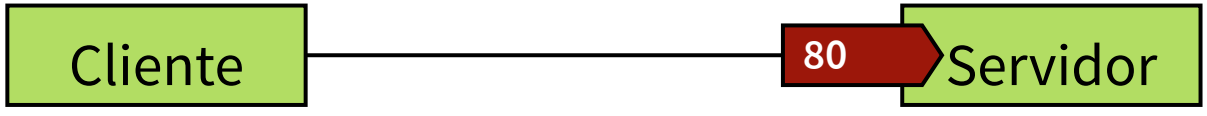
SOCKET DE SERVIDOR

- Recibe las peticiones de a través de su flujo de entrada.
- Envía las respuestas a través del flujo de salida.

SOCKET DE CLIENTE

- Envía las peticiones a través del flujo de salida.
- Recibe las respuestas a través del flujo de entrada.

APLICACIONES WEB

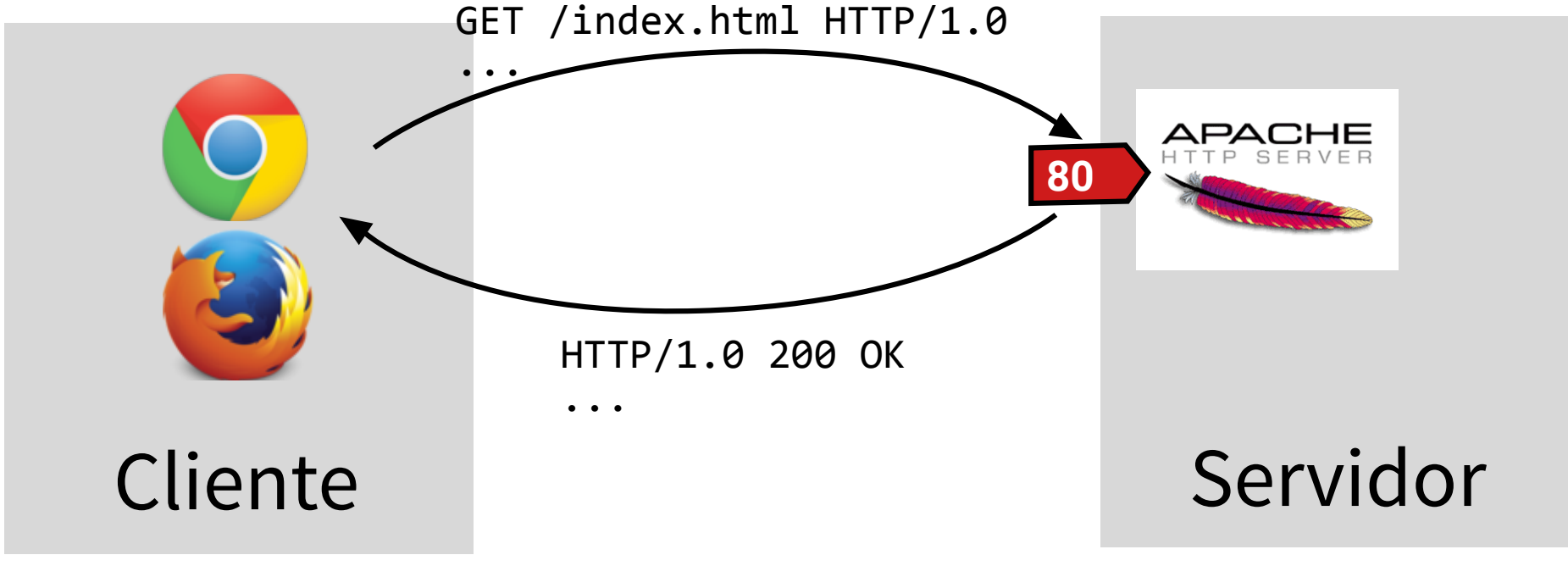


Las aplicaciones web también utilizan el modelo cliente-servidor:

- **Cliente:** Navegador web.
  Firefox, Chrome, Opera, etc.
- **Servidor:** Servidor web.
  Apache HTTP server, Tomcat, Nginx, etc.
- **Protocolo:** HTTP

1. REPASO DEL MODELO TCP/IP
2. MODELO CLIENTE/SERVIDOR
3. SOCKETS Y PROTOCOLOS
4. PROTOCOLO HTTP
5. REFERENCIAS

ESQUEMA GENERAL



ESTRUCTURA DE UNA PETICIÓN HTTP

```
GET /register.html HTTP/1.1
Host: www.foo.com
Accept: text/html
Accept-Language: en-US, es-ES
Connection: keep-alive
If-Modified-Since: Fri, 26 Feb 2016 03:09:05 GMT
                   (línea en blanco)
name=Francisco%20Sanchez&age=12
...
```

Línea de petición

Cabeceras

Cuerpo de la petición (opcional)

LÍNEA DE PETICIÓN

Tiene el siguiente formato:

```
[método] [recurso] [versión_HTTP]
```

- `[método]`: Acción a realizar (GET, POST, HEAD, PUT, ...)
- `[recurso]`: Nombre del recurso (URI) sobre el que se accederá.
- `[versión_HTTP]`: HTTP/1.0 o HTTP/1.1.

Ejemplo:

```
GET /index.html HTTP/1.1
```

ALGUNOS MÉTODOS DISPONIBLES

- `GET`: Obtener un recurso del servidor.
- `HEAD`: Igual que GET, pero el servidor no devolverá el recurso; sólo las cabeceras de la respuesta.
- `POST`: Enviar información al servidor.
- `PUT`: Guardar información en el servidor.
- `DELETE`: Eliminar información del servidor.
- `OPTIONS`: Obtener la lista de métodos soportados por el servidor.

RECURSOS HTTP

```
[método] [recurso] [versión_HTTP]
```

El nombre de recurso tiene la aparencia de un nombre de fichero, pero **no tiene por qué corresponder con un fichero físico almacenado en el servidor**

Un servidor web puede hace corresponder las URIs con llamadas a programas (por ejemplo, servlets), accesos a recursos físicos almacenados con un nombre distinto, etc.

CABECERAS EN UNA PETICIÓN

Especifican información adicional en una petición

```
Host: www.foo.com
Accept: text/html
Accept-Language: en-US, es-ES
Connection: keep-alive
If-Modified-Since: Fri, 26 Feb 2016 03:09:05 GMT
```

ALGUNOS EJEMPLOS DE CABECERAS

- `Host: www.ucm.es`
  Obligatorio en HTTP/1.1. Indica el servidor virtual al que acceder.
- `Accept: text/html, image/gif, */*`
  Tipos de contenido (MIME types) que se espera recibir (Lista de tipos MIME)
- `Accept-Language: en-US`
  Idioma(s) en los que se espera recibir el recurso (Idiomas y Regiones)
- `Accept-Charset: UTF-8`
  `Accept-Encoding: x-gzip`
  Codificación de caracteres.

ALGUNOS EJEMPLOS DE CABECERAS

- `User-Agent: Mozilla/5.0 ...`
  Navegador o cliente que envía la petición.
- `Content-Type: text/html`
  En peticiones POST, indica el tipo de contenido que se adjunta en el cuerpo de la petición.
- `If-Modified-Since: Fri, 26 Feb 2016 03:09:05 GMT`
  Indica al servidor que sólo devuelva el recurso si ha sido modificado después de la fecha dada.

ESTRUCTURA DE UNA RESPUESTA HTTP

```
HTTP/1.1 200 OK
Date: Mon, 15 Aug 2016 14:05:30 GMT
Server: Apache
Set-Cookie: ucmweb=a8bcbkqfml72df71mee86drgj7; domain=ucm.es
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Content-Type: text/html; charset=utf-8
```

(línea en blanco)

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>UCM-Universidad Complutense de Madrid</title>
...
```

Línea de estado

Cabeceras de la respuesta

Cuerpo de la respuesta

LÍNEA DE ESTADO

```
HTTP/1.1 200 OK
```

Indica la versión del protocolo HTTP utilizada, un código de estado y un texto con la descripción dicho código.

CÓDIGOS DE ESTADO

- `200 OK`: Petición recibida y servida con éxito.
- `3xx`: Redirección: el recurso ha cambiado de dirección.
  - `301 Move Permanently`
  - `302 Move Temporarily`
  - `304 Not modified`

En los códigos 301 y 302 se indica en la cabecera de la respuesta la nueva dirección del recurso. El código 304 se devuelve en el caso en el que el cliente haya incluido `If-Modified-Since` y el recurso no haya cambiado desde la fecha indicada.

CÓDIGOS DE ESTADO

- `4xx`: Errores atribuibles al cliente.
  - `400 Bad Request`: Petición incorrecta.
  - `401 Authentication Required`: Se requiere identificación (nombre y contraseña).
  - `403 Forbidden`: Identificación incorrecta.
  - `404 Not Found`: Recurso no encontrado en el servidor.
  - `405 Method not Allowed`: Acción (GET, POST, etc.) no permitida.
- `5xx`: Errores atribuibles al servidor.
  - `500 Internal Server Error`
  - `503 Service Unavailable`

CUERPO DE LA RESPUESTA

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>UCM-Universidad Complutense de Madrid</title>
...
```

Tal y como indica la cabecera `Content-Type: text/html`, el cuerpo de la respuesta contiene un texto en formato HTML que define la estructura de la página web recuperada.

El código HTML puede hacer referencia a otros recursos:

```
...
<link type="text/css" media="screen" rel="stylesheet"
href="/themes/ucm3/css/portada.css?a=1" />
...
<img src="/nodes/web/00media/img/ucm-ws.png"/>
...
```

Cada uno de estos recursos deberá ser obtenido por el navegador Web en sucesivas peticiones HTTP al servidor.

1. REPASO DEL MODELO TCP/IP
2. MODELO CLIENTE/SERVIDOR
3. SOCKETS Y PROTOCOLOS
4. PROTOCOLO HTTP
5. REFERENCIAS

REFERENCIAS

- Wikipedia - *Internet Protocol Suite*
  https://en.wikipedia.org/wiki/Internet_protocol_suite
- *An Introduction to HTTP Basics*
  https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html