

Protocolo HTTP y lenguaje HTML

DWESE

IES Doñana

Ricardo Pérez López

Protocolo HTTP y lenguaje HTML

1. Arquitectura cliente/servidor
2. HTML 5 básico I (recordatorio de primero)
3. Protocolo HTTP
 - a. URI
 - b. Peticiones y respuestas
 - c. Verbos: GET, POST
 - d. Códigos de estado
 - e. Experimentos
 - i. telnet
 - ii. netcat
 - iii. Google Chrome Dev Tools
 - f. Cookies
4. HTML 5 básico II: Formularios



1. Arquitectura cliente/servidor

1. Arquitectura cliente/servidor

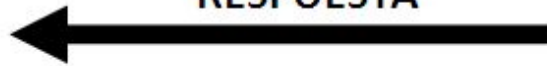


Cliente
(nuestro ordenador en casa)

PETICIÓN

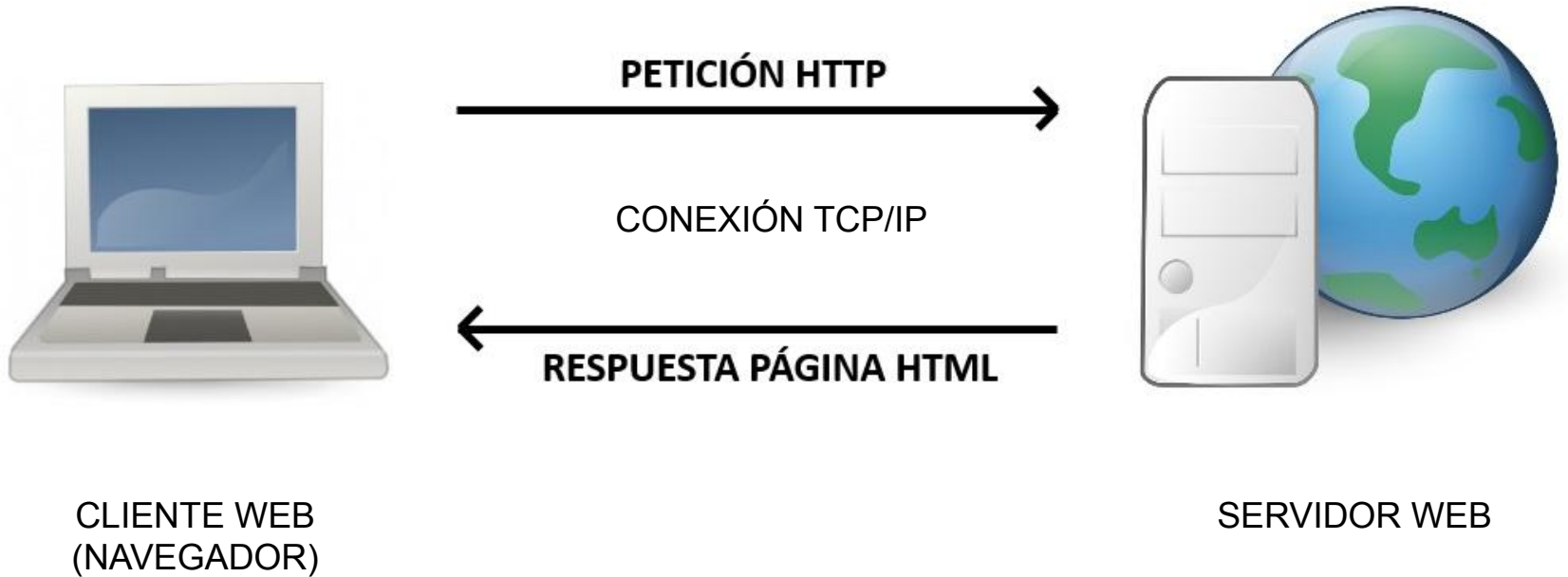


RESPUESTA

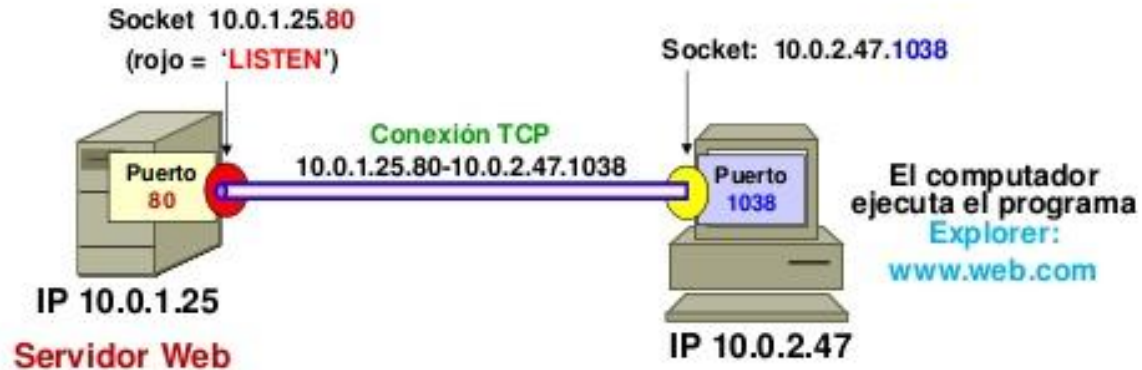


Servidor
(generalmente un gran ordenador remoto)

1. Arquitectura cliente/servidor




1. Arquitectura cliente/servidor



2. HTML 5 básico I

2. HTML 5 básico I

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="estilo.css" />
    <script src="jquery.js"></script>
  </head>
  <body>
    ...
  </body>
</html>
```



3. Protocollo HTTP

3. Protocolo HTTP

- URI
- Peticiones (*requests*) y respuestas (*responses*)
- Verbos: GET, POST
- Códigos de estado
- Recursos
- Experimentos
 - telnet
 - netcat
 - Google Chrome Dev Tools




URI

Dos clases de URI:

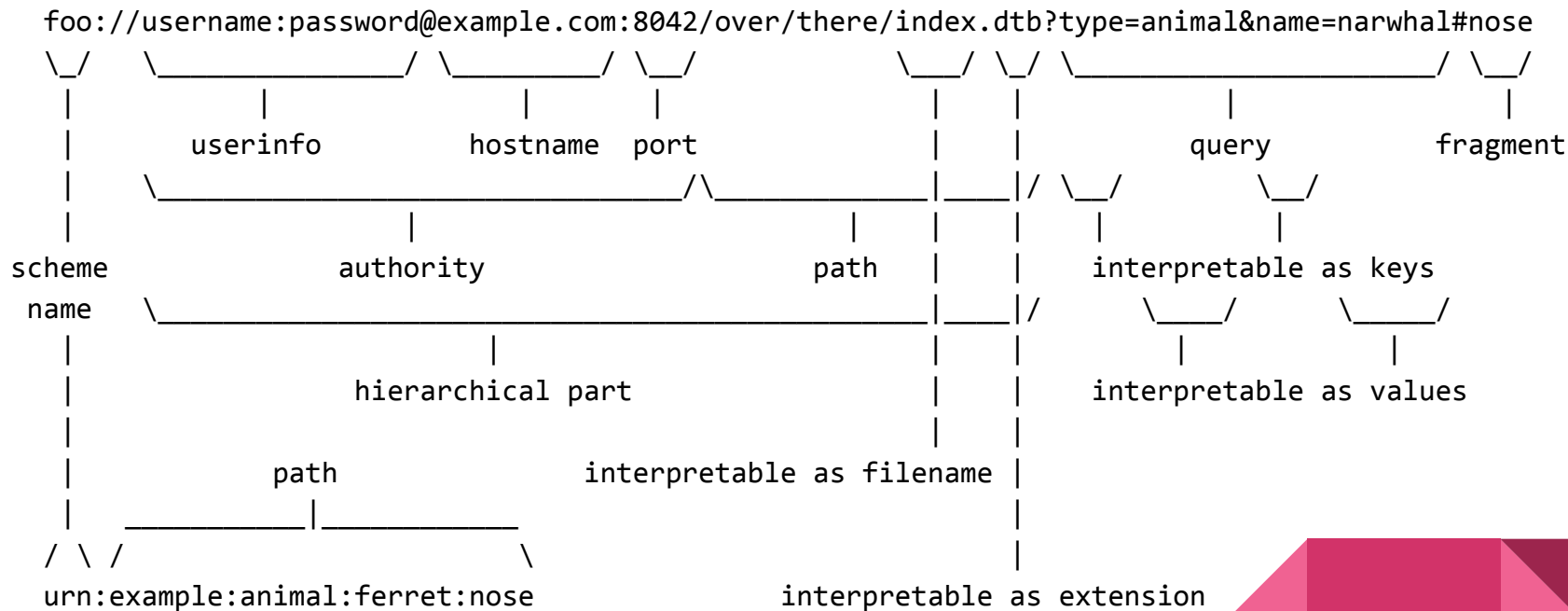
- URL: Identifica un recurso indicando dónde está y cómo alcanzarlo
- URN: Identifica un recurso mediante su nombre

Sintaxis genérica

`nombre_de_esquema : parte_jerárquica [? consulta] [# fragmento]`



URI - Ejemplos



URI - Ejemplos

scheme	name	userinfo	hostname	query
_ _	_ _	_ _	_ _	_ _
/ \	/ \	/ \	/ \	/ \
mailto:username@example.com?subject=Topic				

Peticiones y respuestas

- HTTP requests
- HTTP responses

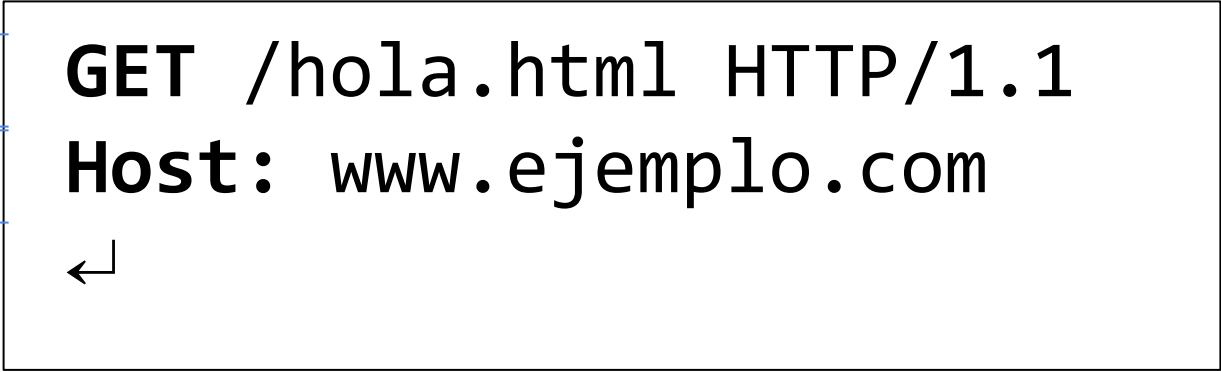


HTTP request

`http://www.ejemplo.com/hola.html`

línea de request

cabeceras

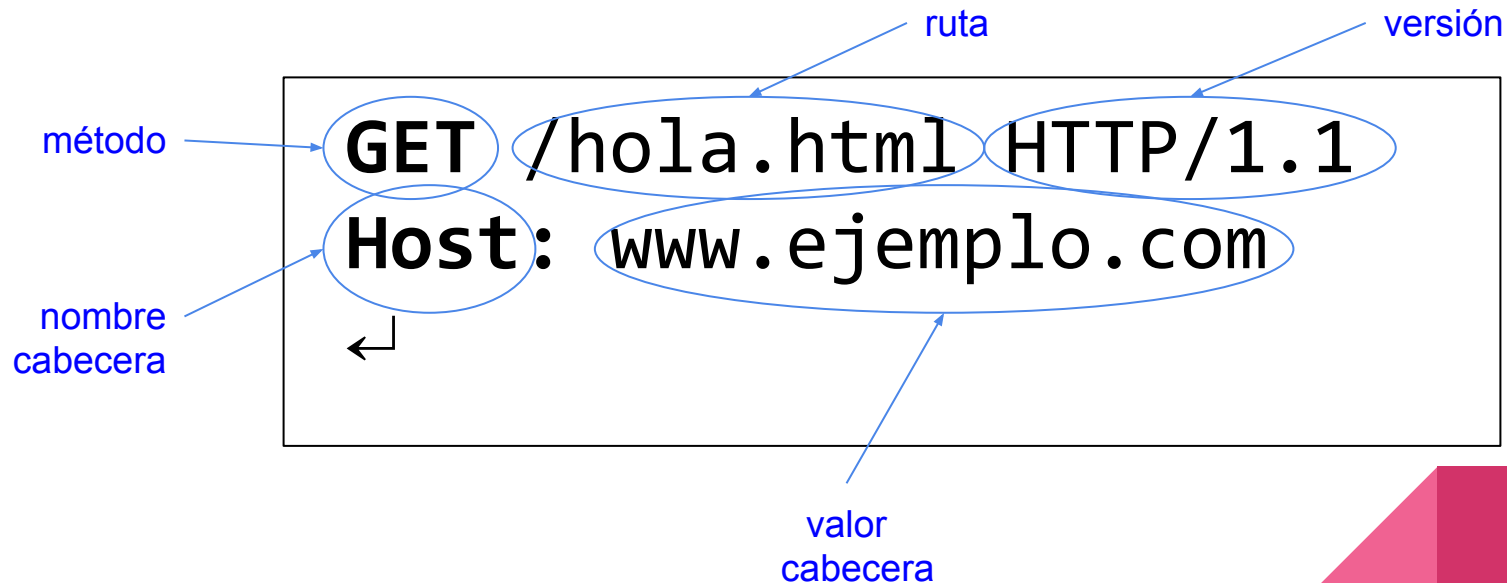


The diagram shows an HTTP request structure within a rectangular box. On the left side of the box, there are two blue brackets. The top bracket is labeled 'línea de request' and the bottom bracket is labeled 'cabeceras'. The text inside the box is as follows:

```
GET /hola.html HTTP/1.1  
Host: www.ejemplo.com  
←
```

HTTP request

http://www.ejemplo.com/hola.html



Métodos HTTP

- **GET**
- **POST**
- **PUT**
- **PATCH**
- **DELETE**
- **HEAD**
- **OPTIONS**



GET vs. POST

GET	POST
Parámetros en la URL	Parámetros en el cuerpo
Usado para recuperar documentos	Usado para actualizar datos
Longitud máxima en la URL	Sin longitud máxima
Se puede cachear	No se puede cachear
No debería cambiar nada en el servidor	Puede cambiar cosas en el servidor



HTTP response

HTTP/1.1 200 OK

Date: Tue mar 2012 04:33:33 GMT

Server: Apache /2.2.3

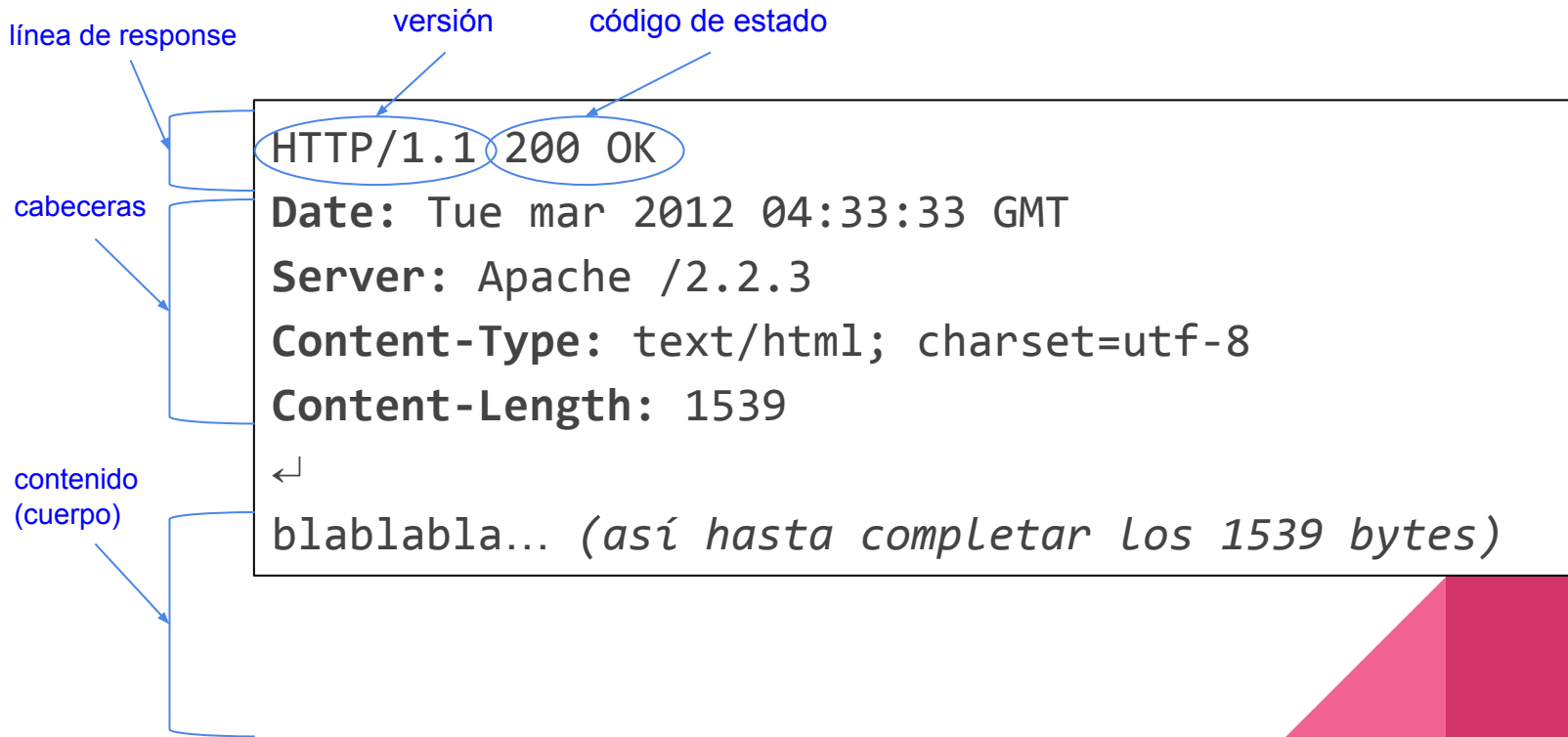
Content-Type: text/html; charset=utf-8

Content-Length: 1539

↵

blablabla... *(así hasta completar los 1539 bytes)*

HTTP response



Otras cabeceras HTTP

https://en.wikipedia.org/wiki/List_of_HTTP_header_fields



Envío de parámetros mediante POST

línea de request

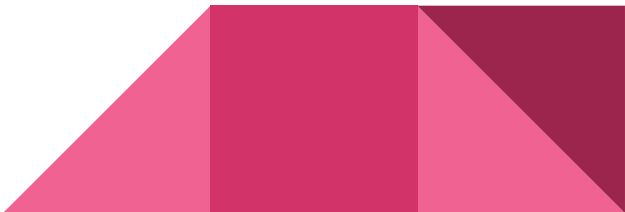
cabeceras

```
POST /shop/28/Shop.aspx HTTP/1.1
Host: mdsec.net
Content-Type: application/x-www-form-urlencoded
Content-Length: 20
↵
quantity=1&price=449
```

contenido
(cuerpo)

Códigos de estado

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

- **1**** : Información
 - **2**** : Éxito
 - **3**** : Redirección
 - **4**** : Error en el cliente
 - **5**** : Error en el servidor
- 

Códigos de estado

- 200 OK
- 301 Moved Permanently
- 302 Found
- 304 Not modified
- 404 Not found
- 500 Server error



Ejemplo: redirecciones

- Código de estado: 302 Found
- Cabecera: Location

```
HTTP/1.1 302 Found
```

```
Location: http://www.ejemplo.com/nuevositio.html
```

```
↵
```

Cookies



Cookies



Cookies

1. El servidor responde al cliente con esta cabecera:

Set-Cookie: usuario=manolo123

2. El cliente (navegador), a partir de ahora, enviará al servidor esta cabecera:

Cookie: usuario=manolo123



Experimentos

- telnet
- netcat
- Google Chrome Dev Tools



telnet

```
$ telnet www.ejemplo.com 80
```

```
GET /hola HTTP/1.0
```

```
Host: www.ejemplo.com
```

Respuesta del servidor:

```
HTTP/1.0 200 OK
```

```
Content-Type: text/html; charset=utf8
```

```
...
```



netcat (1/2)

```
$ nc -l -p 8000
```

En el navegador ponemos, por ejemplo: **http://localhost:8000/pepe**

En el terminal veremos:

```
GET /pepe HTTP/1.1
Host: localhost:8000
Connection: keep-alive
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_1) AppleWebKit/537.1 (KHTML, like Gecko)
Chrome/21.0.1180.89 Safari/537.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate,sdch
Accept-Language: es-ES,es;q=0.8
Accept-Charset: UTF-8,*;q=0.5
```



netcat (2/2)

El navegador se queda a la espera de más información.

Si ahora en el terminal escribimos:

Hola

y pulsamos ^D, se cerrará la conexión y en el navegador aparecerá:

Hola



4. HTML 5 básico II: Formularios

4. HTML 5 básico II: Formularios

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="estilo.css" />
    <script src="jquery.js"></script>
  </head>
  <body>
    <form action="destino.html" method="post">
      <label for="telefono">Teléfono:</label>
      <input type="text" name="telefono" id="telefono" />
      <input type="submit" value="Enviar" />
    </form>
  </body>
</html>
```



Ejercicio

- Formularios en HTML5. ¿Qué hay de nuevo?
 - Tipos semánticos de entrada para la etiqueta `<input>`:
 - `email, url, date, time, datetime, month, week, number, range, tel, search, color`
 - Nuevos atributos:
 - `autofocus, min, max, pattern, placeholder, required, step`
 - Nuevas etiquetas:
 - `<datalist>, <output>`

