

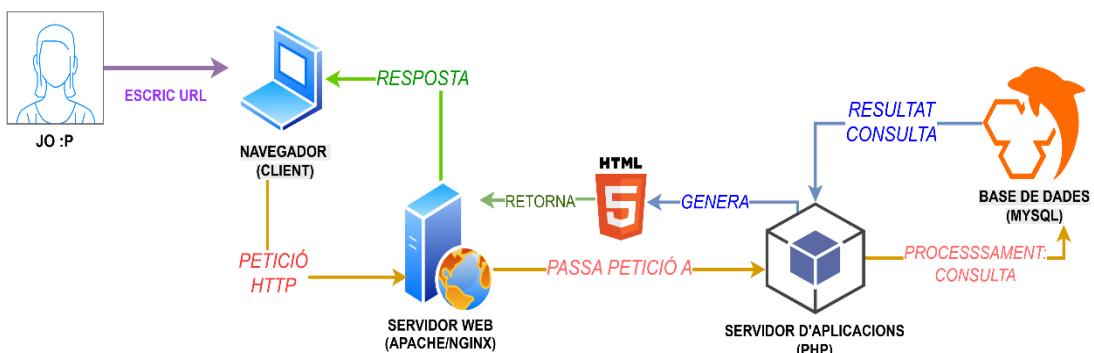
		MÒDUL 0613 – Desenvolupament web en entorn servidor			
		Introducció a l'Arquitectura Client–Servidor			
XAVIER	ETP Xavier	DATA	15/09/2025	GRUP	2n
		NOM	Sana Rut Sabir París		

## Exercici 1 — Dibuixa l'arquitectura web

Objectiu: entendre els components bàsics d'una aplicació web.

Enunciat

- Dibuixa un esquema simple que mostri el camí que segueix una petició web.
- Inclou, com a mínim:
  - Navegador (client)
  - Servidor Web (Apache/Nginx)
  - Servidor d'aplicacions (PHP)
  - Base de dades (MySQL)
  - Resposta HTML al navegador
- Utilitza fletxes per indicar el flux (petició → processament → resposta).



Espero que l'exercici sigui entenedor. He afegit icones i fet ús de colors per a que sigui més visual. M'he basat en l'esquema proposat als apunts i en aquesta part:



### Flux pas a pas d'una petició

- Usuari escriu una URL i prem Enter.
- El client (navegador) engega una petició HTTP(S).
- El servidor web rep la petició i la passa a l'aplicació (PHP/Laravel).
- L'aplicació processa la petició (lògica + consultes a la BBDD).
- L'aplicació genera una resposta (HTML/JSON) i la retorna.
- El navegador rep la resposta i renderitza la pàgina; carrega recursos addicionals.

## Exercici 2 — Explora una petició HTTP amb DevTools

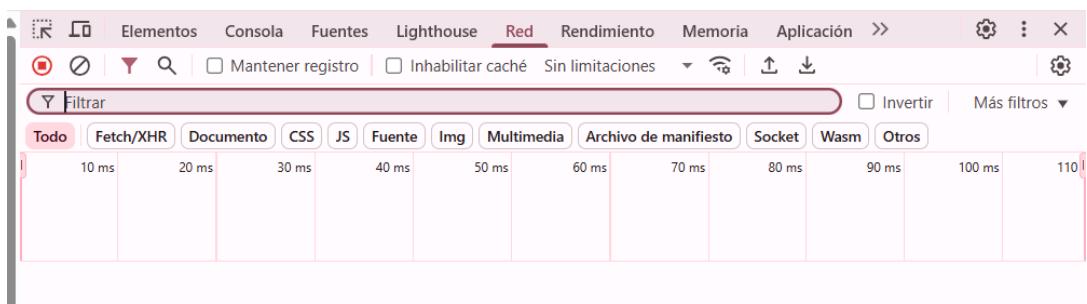
Objectiu: veure a la pràctica què passa quan un navegador fa una petició a un servidor.

Enunciat

1. Obre el teu navegador (Chrome, Firefox, Edge...).
2. Ves a la pàgina <https://www.wikipedia.org>.



3. Obre les DevTools (F12 o clic dret → Inspeccionar) i ves a la pestanya Network.



4. Recarrega la pàgina (F5).

A screenshot of the Network tab in Chrome DevTools after a page reload. The timeline shows a series of requests. The first request, 'www.wikipedia.org', now has a status of 200. The table below shows the details for each request:

Nombre	Estado	Tipo	Iniciador	Tamaño	Tiempo
www.wikipedia.org	304	document	Otros	0,7 kB	47 ms
Wikipedia-logo-v2@1.5x.png	200	png	(índice):30	(caché de m...)	0 ms
index-d786613379.js	200	script	(índice):874	(caché de m...)	0 ms
gt-ie9-38c8b5f74ajs	200	script	(índice):875	(caché de m...)	0 ms
sprite-e49fbf32.svg	200	svg+xml	(índice):617	(caché de m...)	0 ms
Wikinews-logo_sister@2x.png	200	png	(índice):875	(caché de m...)	0 ms
wikipedia.ico	200	vnd.microsof...	Otros	(caché de dis...)	42 ms

5. Respon:

- o Quin és el primer recurs que es demana?

El primer recurs que es demana és [www.wikipedia.org](http://www.wikipedia.org), que apareix en primera línia. La llista de peticions a DevTools és en ordre cronològic. Per tant, la primera és la que jo específicó. S'especifica que el tipus es “document”, així que aquest es l’HTML, a partir del que es descarreguen els altres recursos.

Nombre	Estado	Tipo	Iniciador	Tamaño	Tiempo
 www.wikipedia.org	304 Not Modified	document	Otros	0,7 kB 91,9 kB	55 ms 54 ms

- o Quin és el mètode HTTP utilitzat (GET o POST)?

Abans de respondre a això, he recollit informació sobre els dos mètodes, per tal de poder-ho esbrinar, ja que no veig que aparegui implícitament al resultat si és un o l’altre (no a primera vista).

Mètode GET:

`GET`

El método `GET` solicita una representación de un recurso específico. Las peticiones que usan el método `GET` sólo deben recuperar datos.

Mètode POST:

`POST`

El método `POST` se utiliza para enviar una entidad a un recurso en específico, causando a menudo un cambio en el estado o efectos secundarios en el servidor.

<https://developer.mozilla.org/es/docs/Web/HTTP/Reference/Meth ods>

Adjunto una taula per acabar de diferenciar-ho:

MÉTODO	CONCEPTO	OBSERVACIONES
GET	GET lleva los datos de forma "visible" al cliente (navegador web). El medio de envío es la URL. Los datos los puede ver cualquiera.	Los datos son visibles por la URL, por ejemplo: www.aprenderaprogramar.com/action.php?nombre=pedro&apellidos1=gomez
POST	POST consiste en datos "ocultos" (porque el cliente no los ve) enviados por un formulario cuyo método de envío es post. Es adecuado para formularios. Los datos no son visibles.	La ventaja de usar POST es que estos datos no son visibles al usuario de la web. En el caso de usar get, el propio usuario podría modificar la URL escribiendo diferentes parámetros a los reales en su navegador, dando lugar a que la información tratada no sea la prevista.

[https://www.aprenderaprogramar.com/index.php?option=com\\_content&view=article&id=527:get-y-post-html-method-formas-de-envio-de-datos-en-formulario-diferencias-y-ventajas-ejemplos-cu00721b&catid=69&Itemid=192](https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=527:get-y-post-html-method-formas-de-envio-de-datos-en-formulario-diferencias-y-ventajas-ejemplos-cu00721b&catid=69&Itemid=192)

El mètode doncs ha de ser GET, ja que només sol·licito la pàgina.

Nombre	Encabezados	Vista previa	Respuesta	Iniciador	Tiempos	Cookies
www.wikipedia.org	▼ General					
Wikipedia-logo-v2@1.5x.png	URL De Solicitud	https://www.wikipedia.org/				
/portal/wikipedia.org/asset...	Método De La Solicitud	GET				
index-d786613379.js	Código De Estado	● 304 Not Modified				
/portal/wikipedia.org/asset...	Dirección Remota	185.15.58.224:443				
	Política De Referencia	strict-origin-when-cross-origin				

- o Quin codi d'estat retorna el servidor (200, 301, 404...)?

Nombre	Estado
www.wikipedia.org	304 Not Modified

Codi d'estat “304 Not Modified” . Segons el que he trobat a internet, el motiu és que està guardat al caché, i al fer el refresh apareix aquest codi:

# 304 Not Modified

El código HTTP de redirección **304 Not Modified** en el response de la petición indica que no hay necesidad de retransmitir los recursos solicitados. Es una redirección implícita a un elemento/recurso de caché. Esto sucede cuando el método de la solicitud es seguro (safe), como en las peticiones con métodos GET o HEAD (inglés), o cuando el request (petición) está condicionada y usa la cabecera If-None-Match (inglés) o un If-Modified-Since (inglés). El response **200 OK** habría incluido los encabezados Cache-Control, Content-Location, Date (inglés), ETag, Expires, y Vary.

<https://developer.mozilla.org/es/docs/Web/HTTP/Reference/Status/304>

No obstant, els següents codis dels altres elements son el codi d'estat "200 OK".

	index-d786613379.js /portal/wikipedia.org/assets/js		200 OK
	gt-ie9-38c8b5f74a.js /portal/wikipedia.org/assets/js		200 OK
	sprite-e49fbf32.svg /portal/wikipedia.org/assets/img		200 OK

## 200 OK

El código de respuesta de estado satisfactorio HTTP **200 OK** indica que la solicitud ha tenido éxito. Una respuesta 200 es almacenable de forma predeterminada.

El significado de un éxito depende del método de solicitud HTTP:

- GET: El recurso ha sido recuperado y se transmite el mensaje al body.
- HEAD (inglés): Los encabezados de entidad están en el body del mensaje.
- POST: El recurso que describe el resultado de la acción se transmite en el body del mensaje.
- TRACE: El body del mensaje contiene el mensaje de solicitud tal como lo recibió el servidor.

El resultado exitoso de un método PUT o uno DELETE no es a menudo un **200 OK** sino un **204** (inglés) No Content (o un **201 Created** cuando el recurso es subido por primera vez).

<https://developer.mozilla.org/es/docs/Web/HTTP/Reference/Status/200>

6. Fes una captura de pantalla i marca-hi aquests elements.

## Exercici 3 — Diferència entre fitxer HTML i fitxer PHP

Objectiu: entendre diferència entre contingut estàtic i contingut generat al servidor.

Enunciat

1. Crea dos fitxers dins la carpeta del servidor local (www amb Laragon o htdocs amb XAMPP).

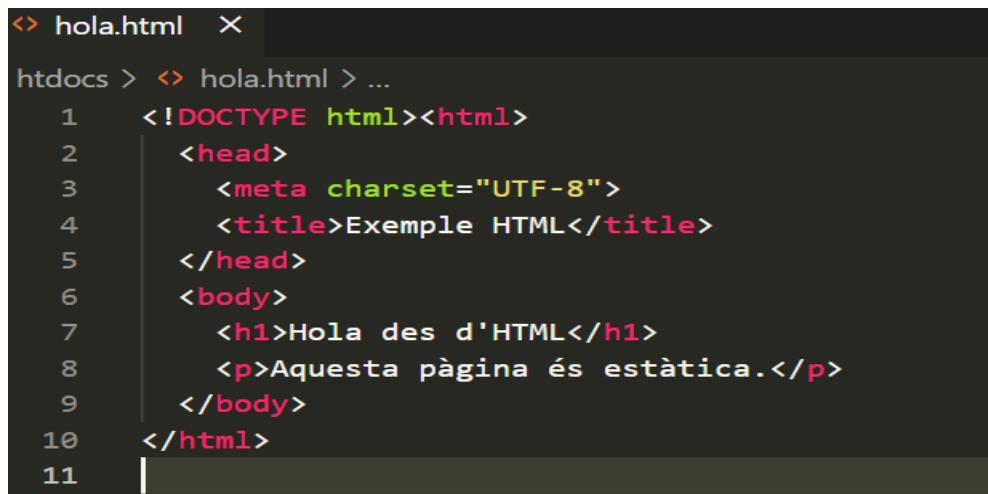
Fitxer hola.html:

```
<!DOCTYPE html><html>
<head>
<meta charset="UTF-8">
<title>Exemple HTML</title>
</head>
<body>
<h1>Hola des d'HTML</h1>
<p>Aquesta pàgina és estàtica.</p>
</body>
</html>
```

Fitxer hola.php: <?php echo "<h1>Hola des de  
PHP</h1>"; echo "<p>Aquesta pàgina és generada pel  
servidor.</p>";  
?>

**Començem! :D**

Este equipo > Disco local (C:) >xampp2 > htdocs >



```
hola.html  x
htdocs > hola.html > ...
1  <!DOCTYPE html><html>
2  <head>
3  <meta charset="UTF-8">
4  <title>Exemple HTML</title>
5  </head>
6  <body>
7  <h1>Hola des d'HTML</h1>
8  <p>Aquesta pàgina és estàtica.</p>
9  </body>
10 </html>
11 |
```

```
hola.html    hola.php X
htdocs > hola.php
1 <?php
2 echo "<h1>Hola des de PHP</h1>";
3 echo "<p>Aquesta pàgina és generada pel servidor.</p>";
4 ?>
5
```

2. Obre el navegador i prova:

- o <http://localhost/hola.html>



- o <http://localhost/hola.php>



3. Respon:

- o Quina diferència hi ha entre els dos fitxers?

**La diferència (segons Google) d' HTML i PHP, i també als apunts que ens has facilitat:**

diferencia entre html y php

Todo Imágenes Vídeos Vídeos cortos Noticias Web Libros Más ▾ Herramientas

◆ Vista creada con IA

La diferencia principal es que **HTML es un lenguaje de marcado estático para estructurar el contenido de una página web**, mientras que **PHP es un lenguaje de scripting del lado del servidor que genera contenido dinámico**, como HTML, a partir de su propio código. HTML define la estructura de la página y se puede abrir directamente en un navegador, mientras que PHP requiere un servidor web para ejecutarlo, procesar formularios, interactuar con bases de datos y generar el HTML que finalmente se envía al navegador del usuario.



**ETP Xavier**

### Pas 3 – El servidor web rep la petició

- El servidor web (Apache/Nginx) rep la petició, escolta el port (80/443). Decisions que pren:
  - Si la ruta és d'un fitxer estàtic (css, jpg) serveix directament el fitxer.
  - Si és dinàmic (ex. .php o ruta d'un framework) el web server delega l'execució a l'app server (PHP-FPM via FastCGI, mod\_php, etc.).
    - Quan el servidor Web detecta que la ruta és un fitxer .php, crida el motor PHP (mod\_php) i executa el codi dins d'aquest fitxer.
    - El codi PHP pot:
      - Llegir dades de la petició (\$\_GET, \$\_POST, \$\_COOKIE...),
      - Processar informació (validar, calcular, consultar BBDD)
      - Finalment generar HTML que es retorna al navegador.

**El fitxer HTML és estàtic. El servidor web l'envia al navegador sense cap tipus de modificació.**

**El fitxer PHP és dinàmic. El servidor primer executa el codi PHP i genera un document HTML, que és el que finalment rep el navegador.**

- o Quin fitxer ha executat el servidor abans de retornar el resultat?

**El fitxer PHP, ja que el servidor web detecta l'extensió .php i passa el fitxer al motor PHP, que l'executa i genera la resposta HTML.**

*He fet una comprovació amb Ctrl. + U:*

A screenshot of a browser's developer tools, specifically the "View Source" tab for the URL "view-source:localhost/hola.php". The title bar shows the URL. The main area displays the raw source code of the PHP file:

```
juste de linea □
1 <h1>Hola des de PHP</h1><p>Aquesta pàgina és generada pel servidor.</p>
2
```

Si ens fixem en el codi, aquí s'omet el “echo”. Per tant, això vol dir que s'ha processat amb PHP, i la resposta és la que podem veure nosaltres (part del codi s'oculta i només es veu l'HTML generat com a resultat).

- o Quin fitxer s'ha enviat tal qual sense processar?

**HTML. El servidor web el reconeix com un fitxer estàtic i l'envia directament al navegador sense cap processament.**

A screenshot of a browser's developer tools, specifically the "View Source" tab for the URL "view-source:localhost/hola.html". The title bar shows the URL. The main area displays the raw source code of the HTML file:

```
juste de linea □
1 <!DOCTYPE html><html>
2   <head>
3     <meta charset="UTF-8">
4     <title>Exemple HTML</title>
5   </head>
6   <body>
7     <h1>Hola des d'HTML</h1>
8     <p>Aquesta pàgina és estàtica.</p>
9   </body>
10 </html>
11
```

**Aquí si que podem veure tot el codi original, sense filtres!**