

	NOMBRE:	FECHA:
	APELLIDOS:	
	MÓDULO DWES MULTI3 - UD6 Servicios Web: API REST	Enero 2026

¿Cómo funciona una llamada cURL en PHP?

- 1 Inicializar la conexión (curl_init)
- 2 Configurar la petición (opciones y cabeceras)
- 3 Ejecutar la petición (curl_exec)
- 4 Obtener información adicional (código HTTP, errores)
- 5 Cerrar la conexión (curl_close)
- 6 Procesar la respuesta (JSON → array PHP)

Resumen visual del flujo completo

Relación directa con Cars.php

¿Cómo funciona una llamada cURL en PHP?

(Modelo petición–respuesta explicado paso a paso)

Una llamada cURL en PHP **simula un cliente HTTP** (como un navegador, Postman o [curl.exe](#)) que se conecta a un servidor y recibe una respuesta.

El flujo **siempre es el mismo**:

1 Inicializar la conexión (curl_init)

```
$ch = curl_init($url);
```

Qué significa

Creamos un “cliente HTTP” apuntando a una URL concreta (por ejemplo, la API REST).

Analogía

Abrir el teléfono y marcar el número del servidor.

	NOMBRE:	FECHA:
	APELLIDOS:	
	MÓDULO DWES MULTI3 - UD6 Servicios Web: API REST	Enero 2026

2 Configurar la petición (opciones y cabeceras)

```
curl_setopt_array($ch, [
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
        'Accept: application/json',
        'Authorization: Basic ...'
    ],
]);
```

Qué se define aquí

- **Qué método usamos** (GET, POST, PUT...)
- **Qué cabeceras enviamos** (Authorization, Content-Type, Accept...)
- **Cómo queremos recibir la respuesta** (como string, no por pantalla)

Analogía

Decir quién eres, qué idioma hablas y qué información esperas.

3 Ejecutar la petición (curl_exec)

```
$response = curl_exec($ch);
```

Qué ocurre aquí

- Se envía la petición HTTP al servidor
- El servidor procesa la petición

	NOMBRE:	FECHA:
	APELLIDOS:	
	MÓDULO DWES MULTI3 - UD6 Servicios Web: API REST	Enero 2026

- El servidor devuelve una respuesta HTTP (código + body)

Resultado

- \$response contiene el **body de la respuesta** (normalmente JSON)

Analogía

Hablar con el servidor y escuchar su respuesta.

4 Obtener información adicional (código HTTP, errores)

```
$code = curl_getinfo($ch, CURLINFO_HTTP_CODE);
$error = curl_error($ch);
```

Qué obtenemos

- Código HTTP: 200, 401, 404, 500, etc.
- Errores de red o de cURL (si los hay)

Por qué es importante

Permite distinguir:

- Error del cliente
- Error del servidor
- Problema de red

5 Cerrar la conexión (curl_close)

```
curl_close($ch);
```

	NOMBRE:	FECHA:
	APELLIDOS:	
	MÓDULO DWES MULTI3 - UD6 Servicios Web: API REST	Enero 2026

Qué hace

Libera recursos del sistema.

Regla importante

Toda llamada `curl_init()` debe terminar con `curl_close()`.

6 Procesar la respuesta (JSON → array PHP)

```
$data = json_decode($response, true);
```

Qué ocurre

- El JSON devuelto por la API se convierte en un array asociativo PHP
- Ese array se pasa a la vista para “pintar” los datos

Analogía

Traducir el idioma del servidor a PHP.

Resumen visual del flujo completo

[PHP (mvccurl)]

|

| 1. curl_init(URL)

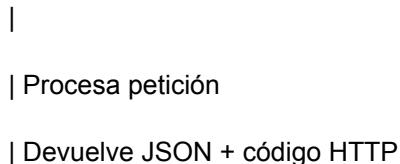
| 2. Configurar cabeceras y opciones

| 3. curl_exec()

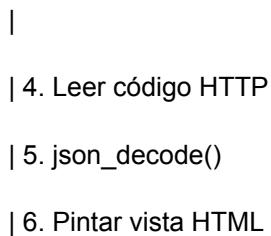
↓

	NOMBRE:	FECHA:
	APELLIDOS:	
	MÓDULO DWES MULTI3 - UD6 Servicios Web: API REST	Enero 2026

[API REST (mvcpapi)]



[PHP (mvccurl)]



Relación directa con Cars.php

En el controlador **Cars.php**:

- **apiGet()** gestiona toda la llamada cURL
- **index()** pide *lista de coches* a la API
- **show(\$id)** pide *un coche concreto* a la API
- La vista **no sabe nada de cURL ni de la API**, solo recibe datos

Esto es MVC bien aplicado:

- El controlador es el cliente HTTP
- La API es un servidor externo
- La vista solo presenta información

	NOMBRE:	FECHA:
	APELLIDOS:	
	MÓDULO DWES MULTI3 - UD6 Servicios Web: API REST	Enero 2026

“cURL convierte nuestra aplicación PHP en un cliente que habla HTTP con otra aplicación.”