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

Autenticación en APIs REST (Basic, Token, JWT) + cURL + Consumo desde otra app

Objetivos de aprendizaje

El alumnado será capaz de:

1. Explicar la diferencia entre **autenticación** y **autorización** en el contexto de una API.
2. Implementar un **login** de API usando **Basic Auth** para emitir un token.
3. Proteger recursos usando **Bearer Token (JWT)**.
4. Verificar y depurar peticiones HTTP usando **cURL** (cabeceras, métodos, códigos).
5. Consumir la API desde una **aplicación cliente** distinta (PHP cURL), adjuntando Authorization: Bearer ....

1) Autenticación en APIs REST (Basic, Token, JWT) + cURL + Consumo desde otra app

1.1 Autenticación vs autorización

- **Autenticación (AuthN):** demostrar quién eres (credenciales, token, certificado).
- **Autorización (AuthZ):** determinar qué puedes hacer (roles, permisos, scopes).

Ejemplo en restaurante:

- AuthN: “soy el cliente-app que tiene acceso”.
- AuthZ: “puedo ver pedidos” pero no “borrarlos” (si no tengo permiso).

1.2 Por qué una API REST suele ser stateless (sin sesión)

En REST, cada request debe contener la información necesaria para ser procesada. Esto favorece escalabilidad y simplicidad (sin "estado" en el servidor). En tu material, la seguridad se integra como paso específico en la versión MVC ("configuración de la seguridad").

1.3 Esquemas de autenticación en APIs

A) Basic Auth (HTTP)

- El cliente envía usuario:password en la cabecera Authorization: Basic ....
- Ventaja didáctica: simple de probar con cURL.
- Riesgo: si no hay HTTPS, la credencial viaja expuesta (por eso, en producción debe ir con TLS).

Uso en este curso (didáctico): Basic Auth para "pedir token" (login).

B) Token simple (Bearer)

- El servidor devuelve un token aleatorio.
- El cliente lo reenvía en Authorization: Bearer <token>.
- Requiere almacenar tokens válidos en BD (revocación, caducidad, etc.). ApiController.php ya ejemplifica token simple y la idea de validarlo contra BD.

C) JWT (JSON Web Token)

- Token firmado, con "claims" internos (p.ej. caducidad).
- Se envía igual: Authorization: Bearer <jwt>.
- No exige guardar cada token en BD para validar (se valida firma + exp). ApiController.php ya implementa generación y validación básica de JWT y señala composer + librería.

1.4 JWT: estructura conceptual

Un JWT típico contiene (conceptualmente):

- **Header:** algoritmo (p.ej. HS256).

- **Payload:** datos (claims) como iat, exp, y datos de usuario.
- **Signature:** firma con clave secreta (HMAC) o clave privada (RSA/ECDSA).

En tu ejemplo se usa:

- Clave: my_secret_key
- iat, exp (+1 hora)
- data con identificadores/valores (incluye passwordToken como “dato genérico”).

Importante didáctico: el JWT **no cifra**, solo firma. Por tanto, no se deben meter secretos en el payload.

1.5 cURL como herramienta profesional de pruebas

cURL permite:

- Probar métodos (-X GET/POST/...)
- Enviar credenciales Basic (-u user:pass)
- Enviar cabeceras (-H "Authorization: Bearer ..." / -H "Content-Type: application/json")
- Enviar body JSON (-d '{...}')
- Ver respuesta y depurar.

cURL es clave para:

- Distinguir errores de **cliente** (400/401/403/404/405) vs **servidor** (500).
- Comprobar si el token se está enviando bien.

Desde la línea de comandos lo podemos probar:

```
curl -i -u profesor:1234 "http://localhost/dwes/mvcapi/apicar/cars"

curl -i -u profesor:1234 "http://localhost/dwes/mvcapi/apicar/car/3"

curl -i -u profesor:1234 ^
  -X POST http://localhost/dwes/mvcapi/apicar/cars ^
  -H "Content-Type: application/json" ^
```

```
-d "{
    \"brand\": \"Seat\",
    \"model\": \"Ibiza\",
    \"color\": \"Blanco\",
    \"owner\": \"Laura\"
  }"
```

Diseñando App que consume API REST

- Consumir mvcapi desde otro proyecto -> mvccurl:
    - GET /apicar/cars (listado)
    - GET /apicar/car/{id} (detalle)
- Hacerlo con:
    - curl en consola (2–3 comandos)
    - PHP cURL (cliente en controlador + vistas)

1. Desde consola:
    - curl -i -u user:pass URL
2. construir mvccurl:
    - config con API_BASE_URL, API_USER, API_PASS
    - Cars::index() + vista tabla
    - Cars::show($id) + vista ficha
3. depuración + entregable
    - capturas de pantalla: consola curl + listado + ficha

Entregable: **2 URLs del cliente** funcionando + captura de curl -i.

requireBasicAuth() en mvcapi

requireBasicAuth() es un **guard (un filtro)**. Si mañana queremo meter roles/scopes, lo normal es:

- requireAuth() (AuthN)
- requireRole('admin') (AuthZ)

El naming “requireX” se lee como “corta la ejecución si no cumples”.

Generar el “cliente mvccurl”

Toma como base el usuario/pass en mvcapi, ahora el siguiente paso es **mvccurl**:

A) mvccurl/config.php (cliente)

- RUTA_URL apuntando a mvccurl
- API_BASE_URL apuntando a mvcapi
- API_BASIC_USER / API_BASIC_PASS **(en mvcapi y mvccurl)**

B) Controlador Cars (cliente)

- index() → llama a mvcapi/apicar/cars con Basic Auth
- show($id) → llama a mvcapi/apicar/car/$id con Basic Auth

C) Vistas

- vistas/cars/index.php tabla con enlaces a /cars/show/{id}
- vistas/cars/show.php ficha JSON o campos bonitos

A continuación tienes lo necesario para mvccurl:

1. mvccurl/app/config/config.php (cliente)
2. mvccurl/app/controladores/Cars.php (cliente HTTP con PHP cURL + Basic Auth)
3. mvccurl/app/vistas/cars/index.php y mvccurl/app/vistas/cars/show.php (con tu patrón RUTA_APP/inc/header|footer)
4. comandos curl para la demo (contra mvcapi)

1) mvccurl/app/config/config.php (cliente)

```
<?php
/**********************
 * CONFIG CLIENTE (mvccurl)
 **********************/

// Ruta de la aplicación
define('RUTA_APP', (dirname(__DIR__)));

// URL base de ESTE proyecto (cliente)
define('RUTA_URL', 'http://localhost/dwes/mvccurl');

define('NOMBRESITIO', 'MVC Cliente (cURL) - Consumidor de API');
// ---- API SERVER (mvcapi) ----
define('API_BASE_URL', 'http://localhost/dwes/mvcapi');

// Basic Auth (didáctico)
define('API_BASIC_USER', 'profesor');
define('API_BASIC_PASS', '1234');

// ---- Endpoints del recurso Cars ----
define('API_CARS_LIST', '/apicar/cars');     // GET listado
define('API_CAR_ITEM',  '/apicar/car/%d');   // GET item por id
```

```
// timeouts recomendados (para aula)
define('API_CONNECT_TIMEOUT', 5);
define('API_TIMEOUT', 10);
```

2) mvccurl/app/controladores/Cars.php

Encaja con el Core: URL /cars/index → controlador Cars, método index.

```
<?php
namespace Cls\Mvc2app;
use Cls\Mvc2app\Controlador;
class Cars extends Controlador
{
    private string $apiBase;
    public function __construct()
    {
        $this->apiBase = rtrim(API_BASE_URL, '/');
    }
    private function apiUrl(string $path): string
    {
        return $this->apiBase . '/' . ltrim($path, '/');
    }
    private function apiGet(string $path): array
    {
```

```
if (!function_exists('curl_init')) {
    return [
        'ok' => false,
        'code' => 0,
        'data' => ['error' => 'Extensión cURL de PHP no habilitada (curl_init no existe).'],
        'raw' => ''
    ];
}

$ch = curl_init($this->apiUrl($path));

curl_setopt_array($ch, [
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
        'Accept: application/json'
    ],

    // Basic Auth
    CURLOPT_USERPWD => API_BASIC_USER . ':' . API_BASIC_PASS,

    // timeouts
    CURLOPT_CONNECTTIMEOUT => API_CONNECT_TIMEOUT,
    CURLOPT_TIMEOUT => API_TIMEOUT,
]);
```

```
$raw  = curl_exec($ch);
$err  = curl_error($ch);
$code = (int) curl_getinfo($ch, CURLINFO_HTTP_CODE);
curl_close($ch);

if ($raw === false) {
    return [
        'ok' => false,
        'code' => 0,
        'data' => ['error' => 'cURL error', 'detail' => $err],
        'raw' => ''
    ];
}

$decoded = json_decode($raw, true);
if (!is_array($decoded)) {
    $decoded = ['raw' => $raw];
}

return [
    'ok' => ($code >= 200 && $code < 300),
    'code' => $code,
    'data' => $decoded,
```

```
        'raw' => $raw
    ];
}

// GET http://localhost/dwes/mvccurl/cars/index
public function index(): void
{
    $resp = $this->apiGet(API_CARS_LIST);

    $datos = [
        'titulo' => 'mvccurl: listado de coches (consumiendo mvcapi con Basic Auth)',
        'http'   => $resp['code'],
        'cars'   => $resp['ok'] ? $resp['data'] : [],
        'error'  => $resp['ok'] ? null : $resp['data'],
    ];

    $this->vista('cars/index', $datos);
}

// GET http://localhost/dwes/mvccurl/cars/show/3
public function show(int $id): void
{
    $path = sprintf(API_CAR_ITEM, $id);
    $resp = $this->apiGet($path);
```

```
        $datos = [
            'titulo' => "mvccurl: ficha coche #$id (consumiendo mvcapi con Basic Auth)",
            'http'   => $resp['code'],
            'car'    => $resp['ok'] ? $resp['data'] : null,
            'error'  => $resp['ok'] ? null : $resp['data'],
        ];

        $this->vista('cars/show', $datos);
    }
}
```

3) Vistas del cliente (mvccurl) con tu patrón header/footer

Crea carpeta: mvccurl/app/vistas/cars/

3.1 mvccurl/app/vistas/cars/index.php

```
<?php require_once RUTA_APP.'/vistas/inc/header.php'; ?>

<h1><?php echo $datos['titulo'] ?? 'Listado'; ?></h1>

<p><strong>Servidor API:</strong> <code><?php echo htmlspecialchars(API_BASE_URL); ?></code></p>

<?php if (!empty($datos['error'])): ?>
  <pre>
Error HTTP <?php echo (int)$datos['http']; ?>:
<?php echo htmlspecialchars(json_encode($datos['error'], JSON_UNESCAPED_UNICODE|JSON_PRETTY_PRINT)); ?>
  </pre>
```

```
<?php endif; ?>


<table border="1" cellpadding="6" cellspacing="0">
  <thead>
  <tr>
    <th>ID</th><th>Brand</th><th>Model</th><th>Color</th><th>Owner</th><th>Acción</th>
  </tr>
  </thead>
  <tbody>
  <?php foreach (($datos['cars'] ?? []) as $c): ?>
    <tr>
      <td><?php echo htmlspecialchars($c['id'] ?? ''); ?></td>
      <td><?php echo htmlspecialchars($c['brand'] ?? ''); ?></td>
      <td><?php echo htmlspecialchars($c['model'] ?? ''); ?></td>
      <td><?php echo htmlspecialchars($c['color'] ?? ''); ?></td>
      <td><?php echo htmlspecialchars($c['owner'] ?? ''); ?></td>
      <td>
        <a href="<?php echo rtrim(RUTA_URL,'/'); ?>/cars/show/<?php echo (int)($c['id'] ?? 0); ?>">Ver ficha</a>
      </td>
    </tr>
  <?php endforeach; ?>
  </tbody>
</table>
<?php require_once RUTA_APP.'/vistas/inc/footer.php'; ?>
```

3.2 mvccurl/app/vistas/cars/show.php

```
<?php require_once RUTA_APP.'/vistas/inc/header.php'; ?>

<h1><?php echo $datos['titulo'] ?? 'Ficha'; ?></h1>

<?php if (!empty($datos['error'])): ?>

  <pre>

Error HTTP <?php echo (int)$datos['http']; ?>:

<?php echo htmlspecialchars(json_encode($datos['error'], JSON_UNESCAPED_UNICODE|JSON_PRETTY_PRINT)); ?>

  </pre>

  <p><a href="<?php echo rtrim(RUTA_URL,'/'); ?>/cars/index">Volver al listado</a></p>

  <?php require_once RUTA_APP.'/vistas/inc/footer.php'; ?>

  <?php return; ?>

<?php endif; ?>

<?php $car = $datos['car'] ?? []; ?>

<ul>

  <li><strong>ID:</strong> <?php echo htmlspecialchars($car['id'] ?? ''); ?></li>

  <li><strong>Brand:</strong> <?php echo htmlspecialchars($car['brand'] ?? ''); ?></li>

  <li><strong>Model:</strong> <?php echo htmlspecialchars($car['model'] ?? ''); ?></li>

  <li><strong>Color:</strong> <?php echo htmlspecialchars($car['color'] ?? ''); ?></li>

  <li><strong>Owner:</strong> <?php echo htmlspecialchars($car['owner'] ?? ''); ?></li>

</ul>


<h3>JSON recibido</h3>

<pre><?php echo htmlspecialchars(json_encode($car, JSON_UNESCAPED_UNICODE|JSON_PRETTY_PRINT)); ?></pre>
```

```
<p><a href="<?php echo rtrim(RUTA_URL,'/'); ?>/cars/index">Volver al listado</a></p>
```

```
<?php require_once RUTA_APP.'/vistas/inc/footer.php'; ?>
```

4) URLs para probar (mvccurl)

- Listado (cliente):
  - http://localhost/dwes/mvccurl/cars/index
- Ficha:
  - http://localhost/dwes/mvccurl/cars/show/3

5) Demo con cURL en consola (contra mvcapi)

curl -i -u profesor:1234 "http://localhost/dwes/mvcapi/apicar/cars"

curl -i -u profesor:1234 "http://localhost/dwes/mvcapi/apicar/car/3"

***Si el servidor exige Basic Auth, sin -u debería devolver 401.**

Nota rápida (si falla "curl_init")

En el servidor web y PHP, la extensión **cURL de PHP** debe estar habilitada. Si falla:

- function_exists('curl_init') da false
- Solución: activar extension=curl en php.ini y reiniciar Apache.