

Reporte Técnico de Actividades Práctico-Experimentales Nro. 005

1. Datos de Identificación del Estudiante y la Práctica

Nombre del estudiante(s)	Alexis Vinicio Roman Avila
Asignatura	Desarrollo Basado en Plataformas
Ciclo	5 A
Unidad	2
Resultado de aprendizaje de la unidad	
Práctica Nro.	005
Título de la Práctica	Implementar una página web responsiva y semántica que realice peticiones simuladas a un servidor (real o mock).
Nombre del Docente	Edison Leonardo Coronel Romero
Fecha	Viernes 21 de noviembre
Horario	07h30 – 10h30
Lugar	Laboratorio Computación aplicada Laboratorio Desarrollo de Software Laboratorio de redes y Sistemas Operativos Laboratorio Virtual EVA Aula
Tiempo planificado en el Sílabo	3 horas

2. Objetivo(s) de la Práctica

- Comprender y aplicar los conceptos de HTTP/HTTPS, métodos REST, códigos de estado y CORS desde el entorno del frontend.
- Implementar una página web responsiva y semántica que realice peticiones simuladas a un servidor (real o mock).
- Registrar, analizar y documentar los resultados obtenidos mediante herramientas de desarrollo.
- Fortalecer la capacidad de contrastar la teoría con la ejecución real dentro del proyecto integrador.

3. Materiales, Reactivos, Equipos y Herramientas

- CVS Code o editor de código equivalente.
- Archivo HTML/CSS/JS base del estudiante.
- Servicio de prueba:
- Documentación cargada previamente en NotebookLM (HTTP/REST/CORS/W3C).

4. Procedimiento / Metodología Ejecutada

- Aplicación rápida de métodos HTTP: GET, POST, PUT, DELETE.
- Explicación del objetivo de la práctica.

5. Resultados

Página Web:

Se creo una página web sencilla con un formulario para el envío de datos hacia el servidor MOCK.

El MOCK utilizado fue [JSONPlaceholder](#).

APE 005 - CRUD

Datos de Entrada

Respuesta del Servidor:

Codigo:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CRUD APE005</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>

  <h1>APE 005 - CRUD</h1>

  <div class="control-panel">
    <h3>Datos de Entrada</h3>
    <input type="number" id="inputId" placeholder="ID (Solo para Update/Delete)"
  />
    <input type="text" id="inputTitle" placeholder="Título del post" />
    <textarea id="inputBody" rows="3" placeholder="Contenido del
post"></textarea>
```

```
<div class="btn-group">
  <button class="btn-get" onclick="realizarPeticion('GET')">GET
(Listar)</button>
  <button class="btn-post" onclick="realizarPeticion('POST')">POST
(Crear)</button>
  <button class="btn-put" onclick="realizarPeticion('PUT')">PUT
(Actualizar)</button>
  <button class="btn-delete" onclick="realizarPeticion('DELETE')">DELETE
(Borrar)</button>
</div>
</div>

<h3>Respuesta del Servidor:</h3>
<div id="resultados"></div>

<script src="logic.js"></script>
</body>
</html>
```

Estilos:

```
body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  max-width: 800px;
  margin: 0 auto;
  padding: 20px;
  background-color: #f4f4f9;
}

h2 {
  color: #333;
}

/* Estilos del formulario */
.control-panel {
  background: white;
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
  margin-bottom: 20px;
}

input,
textarea {
  width: 100%;
  padding: 10px;
  margin: 5px 0;
  border: 1px solid #ddd;
  border-radius: 4px;
  box-sizing: border-box;
}

.btn-group {
  display: flex;
  gap: 10px;
  margin-top: 10px;
  flex-wrap: wrap;
}

button {
  padding: 10px 20px;
  cursor: pointer;
  border: none;
}
```

```
border-radius: 4px;
color: white;
font-weight: bold;
transition: opacity 0.3s;
}

button:hover {
  opacity: 0.9;
}

/* Colores de botones */
.btn-get {
  background-color: #3498db;
}

.btn-post {
  background-color: #2ecc71;
}

.btn-put {
  background-color: #f39c12;
}

.btn-delete {
  background-color: #e74c3c;
}

.btn-clear {
  background-color: #95a5a6;
}

/* Área de resultados */
#resultados {
  background: white;
  padding: 20px;
  border-radius: 8px;
  border: 1px solid #ddd;
  min-height: 100px;
  white-space: pre-wrap;
  font-family: monospace;
  color: #333;
  overflow-x: auto;
}

.log-entry {
  border-bottom: 1px solid #eee;
  padding: 10px 0;
}

.status-ok {
  color: green;
}

.status-error {
  color: red;
}
```

Logica para conectarse con el MOCK:

```
const BASE_URL = 'https://jsonplaceholder.typicode.com/posts';

async function realizarPeticion(metodo) {
  const resultadosDiv = document.getElementById('resultados');
  resultadosDiv.innerHTML = 'Cargando...';
}
```

```
const id = document.getElementById('inputId').value;
const title = document.getElementById('inputTitle').value;
const body = document.getElementById('inputBody').value;

let url = BASE_URL;
let options = {
  method: metodo,
  headers: {
    'Content-type': 'application/json; charset=UTF-8',
  }
};

if (metodo === 'GET') {
  url = id ? `${BASE_URL}/${id}` : `${BASE_URL}?_limit=5`;
}
else if (metodo === 'POST') {
  options.body = JSON.stringify({ title: title, body: body, userId: 1 });
}
else if (metodo === 'PUT') {
  if (!id) { alert("Se requiere un ID para actualizar."); return; }
  url = `${BASE_URL}/${id}`;
  options.body = JSON.stringify({ id: id, title: title, body: body, userId: 1
});
}
else if (metodo === 'DELETE') {
  if (!id) { alert("Se requiere un ID para eliminar."); return; }
  url = `${BASE_URL}/${id}`;
}

const startTime = performance.now();

try {
  const response = await fetch(url, options);

  const endTime = performance.now();
  const duracion = (endTime - startTime).toFixed(2);

  //registro en consola
  console.group(`petición ${metodo}`);
  console.log(`URL solicitada: ${url}`);
  console.log(`metodo usado: ${metodo}`);
  console.log(`tiempo de respuesta: ${duracion} ms`);
  console.log(`codigo de estado: ${response.status}`);
  console.groupEnd();

  let data;
  if (metodo === 'DELETE') {
    data = { mensaje: "Elemento eliminado correctamente (simulado)" };
  } else {
    data = await response.json();
  }

  mostrarEnPantalla(data, response.status);

} catch (error) {
  console.error('Error en la petición:', error);
  resultadosDiv.innerHTML += `<span class="status-error">Error:
${error.message}</span>`;
}

function mostrarEnPantalla(data, status) {
  const resultadosDiv = document.getElementById('resultados');
```

```
const colorClase = (status >= 200 && status < 300) ? 'status-ok' : 'status-error';

resultadosDiv.innerHTML = `
    <strong>Estado:</strong>    <span
class="${colorClase}">${status}</span><br>
    <strong>Datos Recibidos:</strong><br>
    ${JSON.stringify(data, null, 4)}
`;
```

Evidencias de ejecución en consola:

APE 005 - CRUD

Datos de Entrada

1

prueba

diefaalkdsfolkads

GET (Listar) POST (Crear) PUT (Actualizar) DELETE (Borrar)

Respuesta del Servidor:

Estado: 200

Datos Recibidos:

```
{
  "mensaje": "Elemento eliminado correctamente (simulado)"
}
```

petición GET logic.js:44

URL solicitada: https://jsonplaceholder.typicode.com/posts?_limit=5 logic.js:45

metodo usado: GET logic.js:46

tiempo de respuesta: 242.70 ms logic.js:47

codigo de estado: 200 logic.js:48

petición POST logic.js:44

URL solicitada: https://jsonplaceholder.typicode.com/posts logic.js:45

metodo usado: POST logic.js:46

tiempo de respuesta: 268.80 ms logic.js:47

codigo de estado: 201 logic.js:48

petición PUT logic.js:44

URL solicitada: https://jsonplaceholder.typicode.com/posts/1 logic.js:45

metodo usado: PUT logic.js:46

tiempo de respuesta: 270.00 ms logic.js:47

codigo de estado: 200 logic.js:48

petición DELETE logic.js:44

URL solicitada: https://jsonplaceholder.typicode.com/posts/1 logic.js:45

metodo usado: DELETE logic.js:46

tiempo de respuesta: 267.50 ms logic.js:47

codigo de estado: 200 logic.js:48

Pestañas Network:

- Peticion GET

Name	Headers	Payload	Preview	Response
posts?_limit=5	General			
posts	Request URL			https://jsonplaceholder.typicode.com/posts?_limit=5
1	Request Method			GET
1	Status Code			304 Not Modified
	Remote Address			172.67.167.151:443
	Referrer Policy			strict-origin-when-cross-origin
	Response Headers			
	Access-Control-Allow-Credentials			true
	Access-Control-Allow-Origin			null
	Access-Control-Expose-Headers			X-Total-Count
	Age			2698
	Alt-Svc			h3=":443"; ma=86400
	Cache-Control			max-age=43200

- Peticion POST:

Name	×	Headers	Payload	Preview	Response	>>
posts?_limit=5	▼	General				
posts		Request URL		https://jsonplaceholder.typicode.com/posts		
1		Request Method		POST		
1		Status Code		201 Created		
		Remote Address		104.21.59.19:443		
		Referrer Policy		strict-origin-when-cross-origin		
	▼	Response Headers				
		Access-Control-Allow-Credentials		true		
		Access-Control-Allow-Origin		null		
		Access-Control-Expose-Headers		Location		
		Alt-Svc		h3=":443"; ma=86400		
		Cache-Control		no-cache		
		Cf-Cache-Status		DYNAMIC		
4 / 16 requests	1.3 kB / 28.0 kB tr					

Peticion PUT:

Name	×	Headers	Payload	Preview	Response	>>
posts?_limit=5	▼	General				
posts		Request URL		https://jsonplaceholder.typicode.com/posts/1		
1		Request Method		PUT		
1		Status Code		200 OK		
		Remote Address		104.21.59.19:443		
		Referrer Policy		strict-origin-when-cross-origin		
	▼	Response Headers				
		Access-Control-Allow-Credentials		true		
		Access-Control-Allow-Origin		null		
		Alt-Svc		h3=":443"; ma=86400		
		Cache-Control		no-cache		
		Cf-Cache-Status		DYNAMIC		
		Cf-Ray		9a20d2712d5eb602-MIA		
4 / 16 requests	1.3 kB / 28.0 kB tr					

Peticion DELETE:

Name	×	Headers	Preview	Response	Initiator	>>
posts?_limit=5						
posts						
1						
1						

6. Preguntas de Control

- **¿Qué diferencia existe entre un código de estado 200, 201, 400 y 500?**
Estos códigos le dicen al navegador qué pasó con su petición:
 - **200 (OK):** Todo salió bien. El servidor respondió correctamente
 - **201 (Created):** Todo salió bien y además se creó algo nuevo.
 - **400 (Bad Request):** Se enviaron incorrectamente los datos
 - **500 (Internal Server Error):** Error del servidor, algo se rompió en el código del backend.
- **¿Qué función cumple CORS en una aplicación web?**
Es un permiso de seguridad. Le dice al navegador si una página web tiene derecho a pedirle datos a un servidor que está en una dirección diferente. Sin esto, el navegador bloquea la conexión por seguridad.
- **¿Cuál es la diferencia entre request headers y response headers?**
 - **Request Headers:** Lo que se envía hacia el servidor
 - **Response Headers:** Lo que el servidor envía de vuelta
- **¿Por qué es importante documentar los tiempos de respuesta?**
Para saber si tu sistema es rápido o lento y ayuda a detectar fallos de rendimiento.
- **¿Qué riesgos tiene exponer peticiones sin validar en el frontend?**
Para evitar que usuarios maliciosos envíen datos que corrompan el sistema
- **¿Qué consideraciones de seguridad se deben mantener al exponer endpoints entre servicios?**
Si dos servidores se hablan entre sí, se debe usar:
 - **Autenticación interna:** Tokens o llaves (API Keys) para que solo ellos se entiendan.
 - **Red privada:** Que solo se pueda acceder desde direcciones IP conocidas (listas blancas).
 - **Encriptación:** Usar HTTPS para que los datos viajen ocultos.



7. Conclusiones

- Se logro implementar un frontend que realiza peticiones RESTFul y que estas consultan a una API externa para obtener información o escribir datos.
- Se logro comprender la diferencia entre los distintos códigos de respuesta de un servidor web y para que sirve cada uno de ellos.
- Se logro apreciar mediante la visualización en la consola de desarrolladores en el navegador que se envía a través de los headers de una petición HTTP.