

UNIDAD4. ACTIVIDAD2

Francisco Javier Signes Costa 2ºDAW online

CLIENTE

Actividad

En la unidad hemos aprendido distintos aspectos del lenguaje que pueden ser muy interesantes para desarrollar determinadas funcionalidades.

Ahora vamos a crear una aplicación sencilla que necesitará juntar los conocimientos de Desarrollo web en entorno servidor y Desarrollo web en entorno cliente. Recuerda que tenemos que ser creativos, aunque existen unos requisitos mínimos, que son los indicados en la lista de abajo, puedes enriquecerlo con una propuesta más interesante y ambiciosa.

Recuerda que los vídeos de las unidades te pueden servir como ejemplo.

El resultado de la actividad deberás de documentarlo en un archivo de tipo Word que no se extienda más de 5 páginas, explicando paso a paso cómo has creado la aplicación y justificando por qué de las decisiones importantes. Puedes incluir capturas de imagen con fragmentos del código.

El documento y el código deberás de enviárselo a tu profesor a través de la plataforma educativa o haciendo uso de plataformas como WeTransfer.com

Los requisitos mínimos que tendrá que tener el ejercicio son los siguientes:

- Crea una página web que tenga un formulario con el que poder enviar información a un servidor localhost
- A la información enviada al localhost se le añadirá algún tipo de token y se devolverá una respuesta con la misma información y con el token.
- La información recibida se guardará en el navegador utilizando alguna de las opciones que hemos visto
- La página tendrá una tabla que mostrará todos los datos guardados en el sistema de almacenamiento.

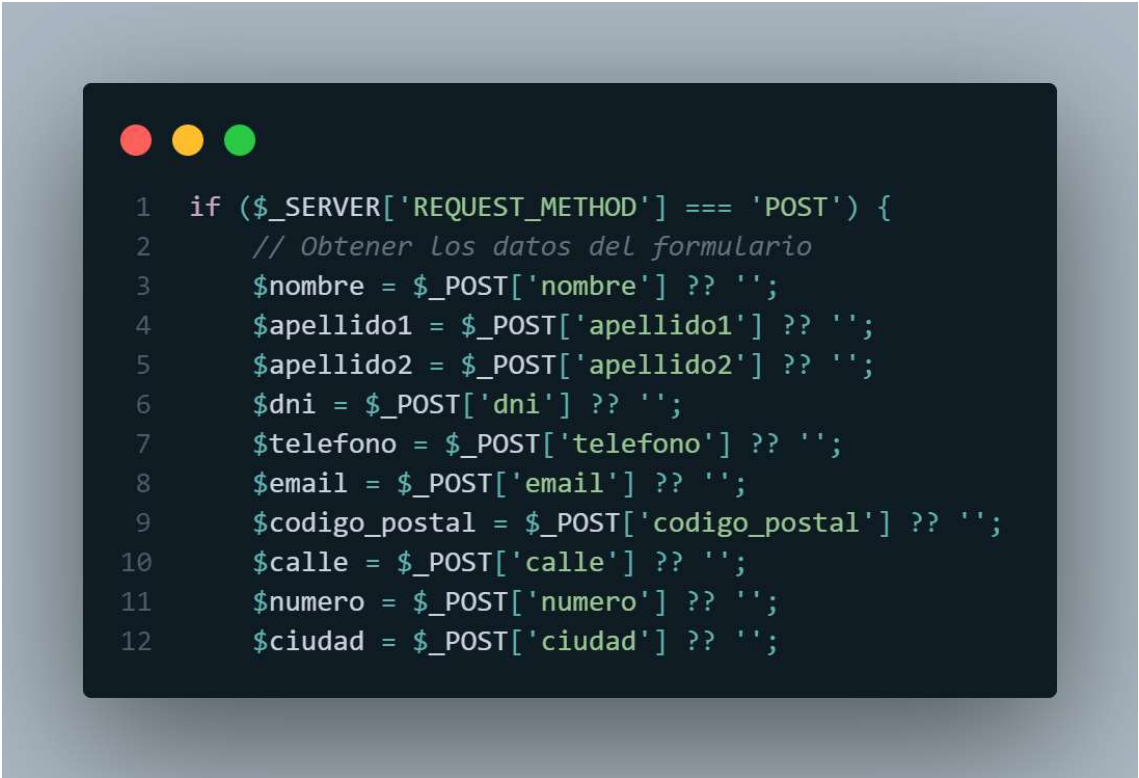
Index.html

Este archivo es el que contiene el formulario de entrada y la tabla donde se reflejan los datos introducidos. No tiene nada de especial. Le he dado unos estilos con Bootstrap para que cuadre un poco y me he centrado más en entender la parte de lógica.

No te pongo pantallazo porque lo tienes en el repositorio de GitHub que te indico al final del documento.

Guardar.php

Este archivo va a procesar los datos procedentes del formulario. Te comento brevemente las partes más importantes.



```
1  if ($_SERVER['REQUEST_METHOD'] === 'POST') {  
2      // Obtener los datos del formulario  
3      $nombre = $_POST['nombre'] ?? '';  
4      $apellido1 = $_POST['apellido1'] ?? '';  
5      $apellido2 = $_POST['apellido2'] ?? '';  
6      $dni = $_POST['dni'] ?? '';  
7      $telefono = $_POST['telefono'] ?? '';  
8      $email = $_POST['email'] ?? '';  
9      $codigo_postal = $_POST['codigo_postal'] ?? '';  
10     $calle = $_POST['calle'] ?? '';  
11     $numero = $_POST['numero'] ?? '';  
12     $ciudad = $_POST['ciudad'] ?? '';
```

Parte del código con la que verificamos que la solicitud al servidor fue enviada mediante el método POST y, si es el caso, recuperamos los datos enviados a través de nuestro formulario index.html.

El operador ?? verifica si el campo existe. Si no existe lo dejamos vacío “. Así evitamos los errores por dejar campos vacíos.

```
1  if (!file_exists($xmlFile)) {  
2      // Crear un XML base con la etiqueta <datos>  
3      $xml = new SimpleXMLElement('<datos></datos>');  
4  } else {  
5  
6      $xml = simplexml_load_file($xmlFile);  
7  }
```

Fragmento donde creamos el archivo data.xml y metemos los datos.

Básicamente aquí lo que hacemos primero es verificar si la variable \$xmlFile contiene el archivo data.xml. Esta variable la hemos declarado justo al empezar el php.

Si no existe lo creamos; si existe lo cargamos. Para eso usamos un condicional.

Con la clase SimpleXMLElement manejamos estructuras XML y con simplexml_load_file cargamos los datos.

```
1  
2  $registro = $xml->addChild('registro');  
3  $registro->addChild('nombre', htmlspecialchars($nombre));  
4  $registro->addChild('apellido1', htmlspecialchars($apellido1));  
5  $registro->addChild('apellido2', htmlspecialchars($apellido2));  
6  $registro->addChild('dni', htmlspecialchars($dni));  
7  $registro->addChild('telefono', htmlspecialchars($telefono));  
8  $registro->addChild('email', htmlspecialchars($email));  
9  $registro->addChild('codigo_postal', htmlspecialchars($codigo_postal));  
10 $registro->addChild('calle', htmlspecialchars($calle));  
11 $registro->addChild('numero', htmlspecialchars($numero));  
12 $registro->addChild('ciudad', htmlspecialchars($ciudad));
```

Añadimos los registros a un objeto SimpleXMLElement que representa un archivo XML. Con addChild vamos creando los distintos nodos que actuarán como contenedores.

Usamos htmlspecialchars para convertir caracteres especiales en entidades HTML. Evitamos problemas al guardar datos y también es útil para evitar ataques XSS.

Data.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE datos [
3      <!ELEMENT datos (registro)*>
4      <!ELEMENT registro (nombre , apellido1 , apellido2 , dni ,
5          telefono , email , codigo_postal , calle , numero , ciudad)>
6      <!ELEMENT nombre (#PCDATA)>
7      <!ELEMENT apellido1 (#PCDATA)>
8      <!ELEMENT apellido2 (#PCDATA)>
9      <!ELEMENT dni (#PCDATA)>
10     <!ELEMENT telefono (#PCDATA)>
11     <!ELEMENT email (#PCDATA)>
12     <!ELEMENT codigo_postal (#PCDATA)>
13     <!ELEMENT calle (#PCDATA)>
14     <!ELEMENT numero (#PCDATA)>
15     <!ELEMENT ciudad (#PCDATA)>
16 ]>
17 <datos>
18     <registro>
19         <nombre>Ramón</nombre>
20         <apellido1>López</apellido1>
21         <apellido2>Pérez</apellido2>
22         <dni>23899332T</dni>
23         <telefono>34877892</telefono>
24         <email>ramon@gmail.com</email>
25         <codigo_postal>50196</codigo_postal>
26         <calle>Iglesia</calle>
27         <numero>33</numero>
28         <ciudad>Madrid</ciudad>
29     </registro>
30 </datos>
```

Guardamos los datos en un xml. No tiene mayor complejidad esta parte.

Script.js

Aquí es donde, personalmente he tenido más problemas. Me resulta complejo trabajar con promesas, teniendo en cuenta que engloban conceptos como funciones callback que considero ya un JavaScript más avanzado. He echado mano de tu código y de toda ayuda que he podido recabar para hacer que el código funcione. A grandes rasgos puedo entender el funcionamiento, pero para ser honesto contigo, no sería capaz de replicarlo sin ayuda.



```
1 document.addEventListener("DOMContentLoaded", function () {
2
3     // Añadimos un evento listener y empezamos con dos constantes que mediante el método querySelector
4     // nos hacemos con los datos que nos interesa. Es interesante usar indistintamente querySelector
5     // o getElementById dependiendo de la circunstancia.
6
7     const formulario = document.querySelector(".formulario-principal");
8     const tablaBody = document.querySelector("table tbody");
9
10    formulario.addEventListener("submit", function (event) {
11        event.preventDefault(); //Hacemos que el formulario no se envíe hasta que se procese por completo
12
13        const formData = new FormData(formulario);
14    })
```

Con `DOMContentLoaded` lo que hacemos es disparar el evento `Listener` cuando el HTML se ha cargado completamente. La función que recibe se ejecutará cuando el DOM esté listo.

Obtenemos elementos con `querySelector`. En este caso los obtenemos por su clase. Podríamos haber usado `getElementById` también, definiendo una `id` en el HTML previamente.

Capturamos después el envío del formulario. Con `event.preventDefault()` evitamos que el formulario recargue la página.

Creamos un objeto `formData` que recopilará los datos del formulario (todos en conjunto) y los prepara para enviarlos al servidor.

```
1  fetch("guardar.php", {
2      method: "POST",
3      body: formData,
4  })
5      .then((response) => response.text())
6      .then((data) => {
7          console.log(data);
8
9          // Si hay un mensaje de "No hay datos registrados", Lo eliminamos
10
11          const mensajeNoDatos = document.querySelector("#mensaje-vacio");
12          if (mensajeNoDatos) {
13              mensajeNoDatos.remove();
14          }
15
16          // Con los datos del fetch construimos la tabla
17
18          const nuevaFila = document.createElement("tr");
19
20          for (const valor of formData.values()) {
21              const celda = document.createElement("td");
22              celda.textContent = valor;
23              nuevaFila.appendChild(celda);
24          }
25
26          // Agregamos cada una de las filas
27
28          tablaBody.appendChild(nuevaFila);
29
30          // Limpiamos el formulario
31
32          formulario.reset();
33      })
34      .catch((error) => console.error("Error:", error));
35  });
```

Esta parte me cuesta un poco.

Usamos fetch para enviar los datos al servidor. Especificamos el método POST y el body va a recoger los datos que hemos almacenado en el objeto formData anteriormente.

Manejamos la respuesta del servidor con response.text() que convierte la respuesta en texto legible y con console.log(data) imprimimos la respuesta para depuración.

Buscamos el id para #mensaje-vacio para el caso de que haya datos. Lo eliminamos.

Creamos una nueva fila en la tabla con el elemento <tr> y recorremos formData con un for. Por cada elemento se crea un <td>, se asigna el valor del formulario a la tabla y se añade con appendChild la celda a la nueva fila. Básicamente este es el proceso.

Posteriormente añadimos a la tablaBody la nueva fila. Limpiamos el formulario y capturamos errores si los hay.

Mucha tela ¿no?. A mí me lo parece...

Repositorio de GitHub

Te dejo el código en el repositorio para que lo pruebes.

<https://github.com/Frank512-lab/ejerciciosSgundoTrimestreCliente.git>