

Práctica 8. Introducción a JSON y AJAX

Objetivo: Manipular elementos de DOM HTML y alimentarlos de datos contenidos en un archivo JSON utilizando AJAX.

Prerrequisitos: Tener la práctica 6 completa

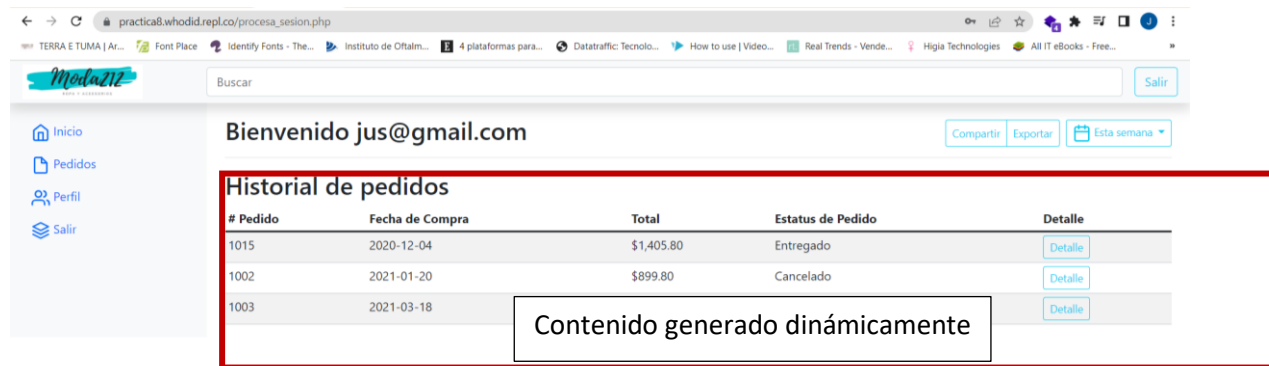
Inicialmente se tendrá un formulario de acceso, en el cual se solicita correo y contraseña

Formulario de acceso con el título "Soy Cliente". Debajo del título, un texto indica: "Campos con * son requeridos". A continuación, se presenta un mensaje: "Si ya cuentas con los beneficios de la Tienda Virtual. Inicia sesión con tu correo electrónico y contraseña." El formulario contiene dos campos de entrada: "Correo electrónico *" y "Contraseña *". Debajo de estos campos hay un botón azul con el texto "Entrar".

Al capturar un correo y una contraseña, se va validar que el usuario esté disponible en los datos almacenados en formato JSON, si el usuario no existe en el archivo JSON o no captura datos se mostrará dinámicamente el mensaje "Correo o contraseña incorrectos, intente de nuevo" como se muestra en la siguiente imagen:

Formulario de acceso con el título "Soy Cliente". Debajo del título, un texto indica: "Campos con * son requeridos". A continuación, se presenta un mensaje: "Si ya cuentas con los beneficios de la Tienda Virtual. Inicia sesión con tu correo electrónico y contraseña." El formulario contiene dos campos de entrada: "Correo electrónico *" y "Contraseña *". Debajo de estos campos hay un botón azul con el texto "Entrar". Debajo del botón, se muestra un mensaje de error en rojo: "Correo o contraseña incorrectos, intente de nuevo". Una flecha roja grande apunta hacia este mensaje de error.

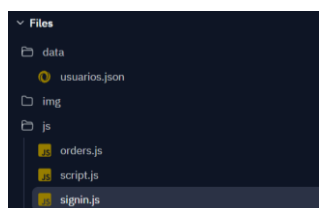
Si el usuario existe, permitirá el acceso a la página `procesa_sesion.php` y mostrará los pedidos asociados al usuario consultados del archivo en formato JSON que se generarán dinámicamente con javascript.



Parte 1. Abre tu proyecto

- Abre tu proyecto en Replit (debe ser del tipo PHP Web Server)
- Ejecuta tu proyecto y valida que funcione tu página login.html
- Dentro de la carpeta llamada js y crea los archivos llamados signin.js y orders.js
- Crea una carpeta llamada data y carga el archivo usuarios.json adjunta a esta práctica.

Debes tener la siguiente estructura similar a



Parte 2. Modificación de estructura del formulario

- Abre el archivo login.html y ubica el código que estructura el formulario con los campos de usuario y contraseña.
- Después del botón agrega este elemento `<p></p>`, ya que permitirá mostrar el mensaje de error cuando el usuario no esté autorizado (Ver imagen de abajo)
- Al botón cambia el type por button en vez de submit y agrega el id="iniciar" (Ver imagen de abajo)

```
119
120 <button type="button" class="form-control btn btn-primary px-3" id="iniciar">Iniciar
    Sesión</button>
121 </div>
122 </div>
123 <p><span id="error"></span></p>
124
125 </form>
126
```

- d) Al elemento <form> agrega el id="formulario"

```
33 <div class="text-center">Soy Cliente:</div>
34 <p>Campos con <span class="text-danger">*</span> son requeridos </p>
35 <p class="justify-content-center">Si ya cuentas con los beneficios de la Tienda Virtual. Inicia sesión con tu correo electrónico y contraseña.</p>
36 </div>
37 <form action="procesa_sesion.php" method="post" id="formulario">
38 <div class="row mb-3">
39 <div class="form-floating col-md-12">
40 <input type="email" class="form-control" id="floatingInput" placeholder="Correo electrónico" required name="email">
41 <label for="floatingInput" class="m-3">Correo electrónico <span class="text-danger">*</span></label>
42 </div>
43 </div>
```

- e) Agrega el script <script type="text/javascript" src="js/signin.js"></script> antes del element </body>. Este script permitirá procesar los datos capturados por el usuario y permitir o negar el acceso.

```
201 </footer>
202 </div>
203 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
    alpha1/dist/js/bootstrap.bundle.min.js" integrity="sha384-
    w76AqPfdkMBDXo30jS1Sgez6pr3x5MlQ1ZAGC+nuZB+EYdgRZglwhtBTKf7CXvN" crossorigin="anonymous">
</script>
204
205 <script type="text/javascript" src="js/signin.js"></script>
206 </body>
207 </html>
```

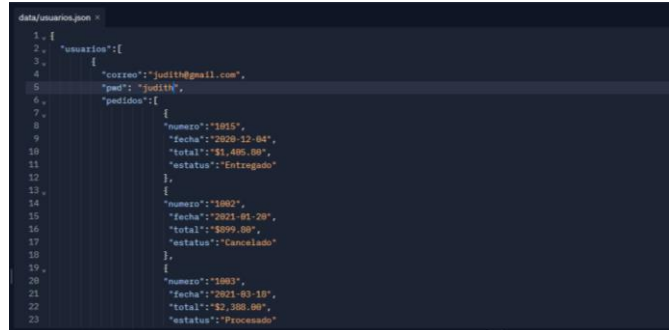
- f) Verifica que los elementos input de email y password tengan el atributo id con los valores floatingInput y floatingPassword, respectivamente (recuerda estos id los contenía la estructura del formulario que se copió de los ejemplos de Bootstrap 5). Haremos referencia a estos id más adelante.

```
39 <div class="form-floating col-md-12">
40 <input type="email" class="form-control" id="floatingInput" placeholder="Correo electrónico" required name="email">
41 <label for="floatingInput" class="m-3">Correo electrónico <span class="text-danger">*</span></label>
42 </div>
43 </div>
44 <div class="row mb-3">
45 <div class="form-floating col-md-12">
46 <input type="password" class="form-control" id="floatingPassword" placeholder="Contraseña" required>
47 <label for="floatingPassword" class="m-3">Contraseña <span class="text-danger">*</span></label>
48 </div>
49 </div>
50
```

- g) Verifica que se muestre correctamente el formulario, ahora procederemos a darle funcionalidad.

Parte 3. Estructura de archivo JSON

- a) Verifica que tengas tu carpeta llamado data, dentro el archivo adjunto a esta práctica llamado usuarios.json



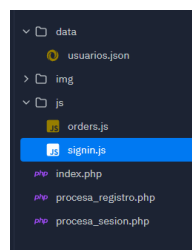
```
1. [
2.   "usuario": [
3.     {
4.       "correo": "judith@gmail.com",
5.       "pass": "judith",
6.       "pedidos": [
7.         {
8.           "numero": "1001",
9.           "fecha": "2020-12-04",
10.          "total": "$1,405.00",
11.          "estatus": "Entregado"
12.        },
13.        {
14.          "numero": "1002",
15.          "fecha": "2021-01-20",
16.          "total": "$999.00",
17.          "estatus": "Cancelado"
18.        },
19.        {
20.          "numero": "1003",
21.          "fecha": "2021-03-10",
22.          "total": "$2,305.00",
23.          "estatus": "Procesado"
24.        }
25.      ]
26.    },
27.    {
28.      "usuario": "Juan",
29.      "correo": "juan@gmail.com",
30.      "pass": "juan",
31.      "pedidos": [
32.        {
33.          "numero": "1004",
34.          "fecha": "2021-04-15",
35.          "total": "$1,200.00",
36.          "estatus": "Entregado"
37.        },
38.        {
39.          "numero": "1005",
40.          "fecha": "2021-05-20",
41.          "total": "$1,500.00",
42.          "estatus": "Procesado"
43.        },
44.        {
45.          "numero": "1006",
46.          "fecha": "2021-06-10",
47.          "total": "$1,800.00",
48.          "estatus": "Entregado"
49.        },
50.        {
51.          "numero": "1007",
52.          "fecha": "2021-07-10",
53.          "total": "$1,200.00",
54.          "estatus": "Procesado"
55.        }
56.      ]
57.    }
58.  ]
```

- b) Da clic sobre el archivo usuarios.json para poder abrirlo y editarlo, el archivo representa usuarios con información de correo, contraseña y pedidos realizados. El archivo contiene dos usuarios, el primer usuario tiene 3 pedidos y el usuario 2 tiene 4 pedidos. Puedes editar el archivo para cambiar el correo y contraseña de los usuarios y así probar tu página con valores nuevos, también puedes cambiar los datos de los pedidos, quitar pedidos o sumarle pedidos a cada usuario.

Para más información de la estructura Json , puedes consultar la documentación:
https://www.w3schools.com/js/js_json_intro.asp

Parte 4. Código Javascript para validar el correo y contraseña

- a) Abre el archivo llamado signin.js dentro de tu carpeta js, aquí se desarrollará todo el código para validar el correo y contraseña capturada por el usuario.



- b) Abre el archivo signin.js y captura el siguiente código, el código tiene comentarios con el objeto de clarificar cada instrucción. Para entender mejor el código puedes consultar la documentación asociada para comprender la petición al servidor asíncrona:

https://www.w3schools.com/xml/xml_http.asp

```
js/login.js
1
2 window.onload=function(){
3
4     //Se obtiene en variables los elementos del DOM del
5     formulario a utilizar
6     var mail = document.getElementById("floatingInput"); //input de email, coincide con el atributo id
7     var pass = document.getElementById("floatingPassword"); //input de contraseña, coincide con el atributo id
8     var error = document.getElementById("error"); // span para mostrar errores
9     var form = document.getElementById("formulario"); // Formulario
10    var btn_iniciar = document.getElementById("iniciar"); // Botón Iniciar Sesión
11    //Regulamos la función procesarFormulario en
12    //el evento click del botón iniciar, al dar click desactivamos la validación
13    //de correo y contraseña capturada contra la definida en el archivo JSON
14    btn_iniciar.addEventListener("click", procesarFormulario);
15
16    //función que abre la conexión con servidor para solicitar el
17    archivo JSON
18    function procesarFormulario() {
19        var xhttp = new XMLHttpRequest(); //Se crea la instancia de la petición
20        xhttp.onreadystatechange = function() {
21            if (this.readyState == 4 && this.status == 200) {
22                leerJSON(this) //función que obtiene y procesa la respuesta del servidor
23            }
24        };
25        xhttp.open("GET", "data/usuarios.json", true) //Se define que recibe JSON que se necesita
26        xhttp.send() //Se envía la petición
27    }
28
29
30
31
32 //Se procesa la respuesta cuando se obtiene de servidor
33 function leerJSON(respuestaJSON)
34 {
35     //Se obtiene el archivo JSON y se convierte en un objeto de Javascript
36     var objJson = JSON.parse(respuestaJSON.responseText);
37     var si;
38     var bandera = false
39     //Se recorre cada usuario para tratamiento
40     for ( i in objJson.usuarios)
41     {
42         //Se valida si el correo y contraseña es igual
43         //a la capturada por el usuario en el formulario
44         if (objJson.usuarios[i].correo == mail.value && objJson.usuarios[i].pwd == pass.value )
45             bandera = true //Si hay coincidencia se cambia el valor de esta bandera
46     }
47     //Se valida el valor de la bandera, si es falso significa que no encontró coincidencias en
48     //el archivo JSON con el usuario y contraseña y muestra el mensaje de error
49     if (bandera==false)
50     {
51         error.style.color="red";
52         error.innerHTML="Correo o contraseña incorrectos, Intente de nuevo"
53     }
54     //Hubo coincidencia con usuario y contraseña
55     //y envía los datos al servidor haciendo el submit del formulario
56     else
57     {
58         form.submit();
59     }
60 }
61
62
63
64 //Se registra el evento focus para que cuando
65 //se tenga el foco en el input de email y contraseña se limpien el tiempo valores
66 //al igual el mensaje de error
67 mail.addEventListener("focus", limpiarMensajeEmail);
68 pass.addEventListener("focus", limpiarMensajePwd);
69 //función para limpiar valores del input email
70 function limpiarMensajeEmail() {
71     error.innerHTML="";
72     mail.value=""
73 }
74
75 //función para limpiar valores del input contraseña
76 function limpiarMensajePwd() {
77     error.innerHTML="";
78     pass.value=""
79 }
80
81
82 }
```

- c) Reserva tu código, aún no está listo tu código para pruebas

Parte 5. Estructura de administrador de pedidos

- a) Abre el archivo procesa_sesion.php
- b) Ubica la línea de código que recibe el email
- ```
<h1 class="h2">Bienvenido <?php echo $_POST["email"]; ?> </h1>
```
- c) Al elemento span agrega el atributo id="correo", quedará
- ```
<h1 class="h2">Bienvenido <span id="correo"><?php echo $_POST["email"]; ?>
</span></h1>.
```
- Esto servirá para obtener el valor de correo logeado y así obtener el detalle de su pedido.

Parte 8. Código Javascript para mostrar los detalles del pedido

- a) Ubica en el código la estructura de `<table></table>` y sustitúyela por el siguiente código:

```
81
82 <h2>Historial de pedidos</h2>
83 <div class="table-responsive">
84   <table class="table table-striped table-sm">
85     <thead>
86       <tr>
87         <th># Pedido</th>
88         <th>Fecha de Compra</th>
89         <th>Total</th>
90         <th>Estado de Pedido</th>
91         <th>Detalle</th>
92       </tr>
93     </thead>
```

- b) Elimina o comenta todo el bloque `<tbody></tbody>`, deja sólo `<tbody id="tabla">` `</tbody>`, el id es importante porque aquí se generará el código de forma dinámica de los pedidos asociado al usuario.

```
81
82 <h2>Historial de pedidos</h2>
83 <div class="table-responsive">
84   <table class="table table-striped table-sm">
85     <thead>
86       <tr>
87         <th># Pedido</th>
88         <th>Fecha de Compra</th>
89         <th>Total</th>
90         <th>Estado de Pedido</th>
91         <th>Detalle</th>
92       </tr>
93     </thead>
94     <tbody id="tabla">
95
96   </tbody>
97 </table>
98 </div>
99 </main>
100 </div>
101 </div>
102
```

- c) Antes de que termine el `</body>` agrega el script `<script type="text/javascript" src="js/orders.js"></script>`, donde se incluirá el procesamiento de los pedidos de forma dinámica

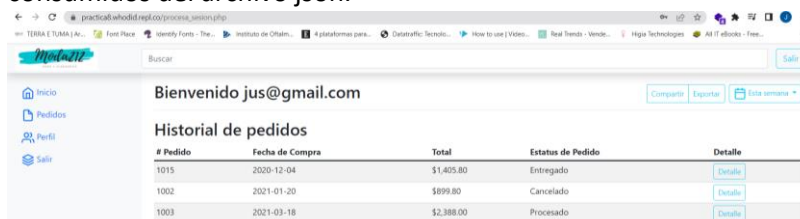
```
111
112 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
113   alpha1/dist/js/bootstrap.bundle.min.js" integrity="sha384-
114   w76AqPfdKMBDXo30js1SgeZ6pr3x5MLQ1ZAGC+nuZB+EYdgRZgiwXhTBTkF7CXvN"
115   crossorigin="anonymous"></script>
116 <script type="text/javascript" src="js/orders.js"></script>
117 </body>
118 </html>
119
120
```

- d) Ahora abre el archivo `orders.js` y captura el siguiente código de Javascript que consulta en el archivo `Json` por los pedidos del usuario que ha iniciado sesión y genera dinámicamente cada fila con la información de cada pedido. El código tiene comentarios para clarificar el desarrollo de las tareas.

```
js/orders.js x
1 window.onload=function(){
2
3   //Se obtiene el valor del correo del usuario que ha iniciado sesión
4   var mail = document.getElementById("correo");
5
6   // Se solicita al servidor el archivo de usuarios JSON
7   var xhttp = new XMLHttpRequest();
8   xhttp.onreadystatechange = function() {
9     if (this.readyState == 4 && this.status == 200) {
10       leerJSON(this) //Función que procesa los datos
11     }
12   };
13
14   xhttp.open("GET", "data/usuarios.json", true) //Se solicita el recurso de archivo JSON
15   xhttp.send() //Se envia la solicitud al servidor
16
```

```
16
17 //Cuando el servidor responde se procesa la información.
18 obteniendo los datos del pedido del usuario*/
19 function leerJSON(respuestaJSON)
20 {
21     //Se obtiene la respuesta del servidor (archivo JSON), en un objeto de javascript
22     var objJson = JSON.parse(respuestaJSON.responseText);
23     var i=0, j;
24     //Se obtiene el cuerpo de la tabla de donde se añadirán las filas de cada pedido
25     var tabla=document.getElementById("tabla")
26     //Se recorre cada usuario
27     for ( i in objJson.usuarios)
28     {
29         //Se valida que el correo corresponda al correo del usuario que ha iniciado sesión
30         if (objJson.usuarios[i].correo == mail.innerHTML.trim() )
31         {
32             j=0;
33             //Cuando el correo coincide se obtiene los pedidos del usuario
34             //y se crean los elementos del DOM HTML para formar las filas de cada
35             //pedidos/
36             for(j in objJson.usuarios[i].pedidos)
37             {
38                 var tr= document.createElement("tr")
39                 var td_pedido= document.createElement("td")
40                 //Se obtiene el número del pedido
41                 td_pedido.innerHTML=objJson.usuarios[i].pedidos[j].numero
42                 var td_fecha= document.createElement("td")
43                 //Se obtiene la fecha del pedido
44                 td_fecha.innerHTML= objJson.usuarios[i].pedidos[j].fecha
45                 var td_total= document.createElement("td")
46                 //Se obtiene el monto del pedido
47                 td_total.innerHTML= objJson.usuarios[i].pedidos[j].total
48                 var td_estatus= document.createElement("td")
49                 //Se obtiene el estatus del pedido
50                 td_estatus.innerHTML= objJson.usuarios[i].pedidos[j].estatus
51                 var td_boton= document.createElement("td")
52
53                 //Se genera el botón que va en la ultima fila
54                 var boton= document.createElement("button")
55                 boton.innerHTML="Detalle"
56                 boton.className="btn btn-sm btn-outline-info"
57                 boton.type="button"
58                 boton.id=objJson.usuarios[i].pedidos[j].numero
59                 //Se añaden todos los elementos a una fila y la fila se agrega
60                 //a la tabla de pedidos/
61                 td_boton.appendChild(boton)
62                 tr.appendChild(td_pedido)
63                 tr.appendChild(td_fecha)
64                 tr.appendChild(td_total)
65                 tr.appendChild(td_estatus)
66                 tr.appendChild(td_boton)
67                 tabla.appendChild(tr)
68             }
69         }
70     }
71 }
72
73
```

- e) Ahora prueba tu formulario completo, haciendo las siguientes pruebas:
- Captura un correo o contraseña incorrectos, debe mostrar el mensaje de error y no iniciar el procesamiento.
 - Captura correo y contraseña de usuario disponible en archivo .json, debe iniciar el procesamiento y mostrar los pedidos asociados al usuario, cuyos datos son consumidos del archivo json.
 - Edita el archivo json, agregando un nuevo usuario con un pedido, vuelve a probar el formulario, y debe funcionar correctamente.
 - En caso de errores, revisa la consola para atender los errores.



# Pedido	Fecha de Compra	Total	Estatus de Pedido	Detalle
1015	2020-12-04	\$1,405.80	Entregado	Detalle
1002	2021-01-20	\$899.80	Cancelado	Detalle
1003	2021-03-18	\$2,388.00	Procesado	Detalle