

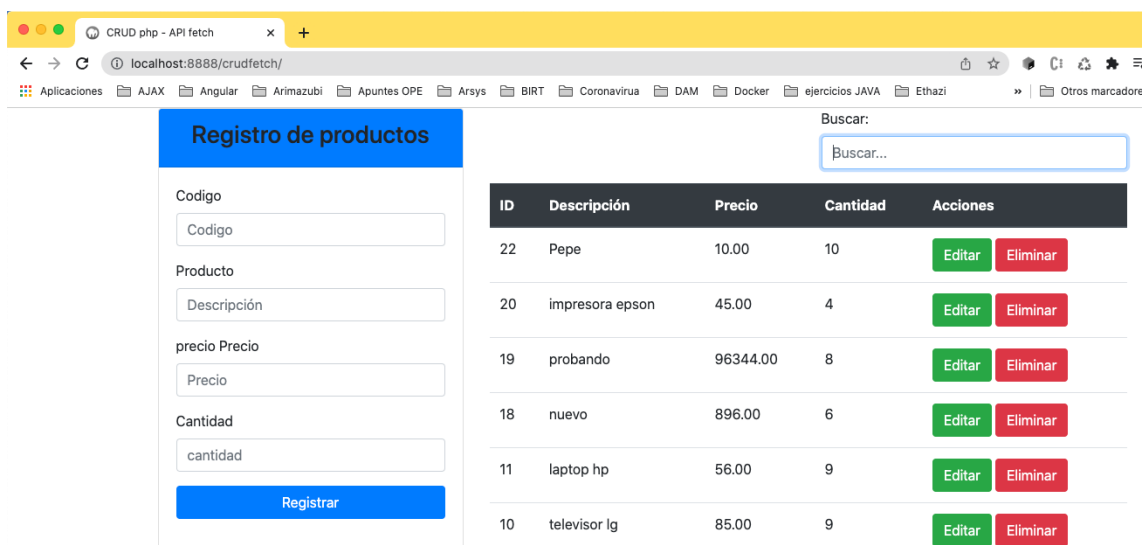
## CRUD paso a paso con fetch

El producto a realizar es un CRUD con PHP y MySQL utilizando la tecnología fetch de javascript.

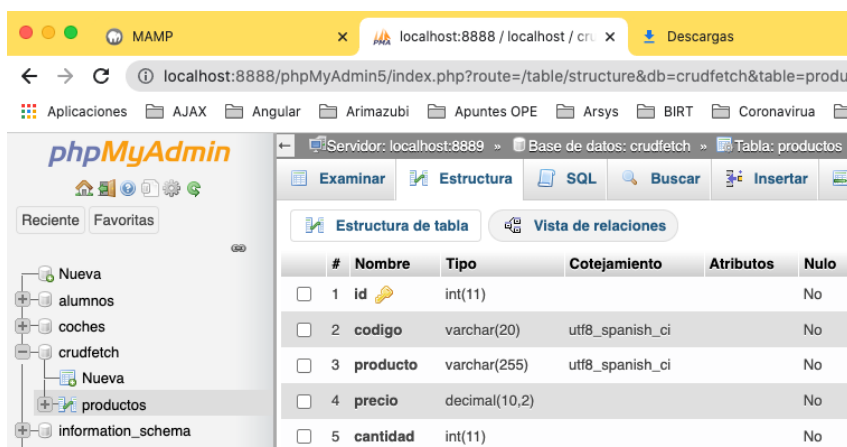
Cuando enviamos un formulario con una base de datos de MySQL con PHP la página se recarga automáticamente y esto es bastante incómodo para los clientes.

Para ello deberíamos realizar las operaciones asíncronas mediante AJAX o fetch. El CRUD AJAX ya lo hemos visto en el anterior tutorial. El objetivo ahora es construir el siguiente tutorial mediante tecnologías fetch.

Este es el producto.



### Paso 1. Importar la base de datos mysql.



El contenido de la tabla es:

id	codigo	producto	precio	cantidad
10	8523	televisor lg	85.00	9
11	963	laptop hp	56.00	9
18	7897685	nuevo	896.00	6
19	75545	probando	96344.00	8
20	78799	impresora epson	45.00	4

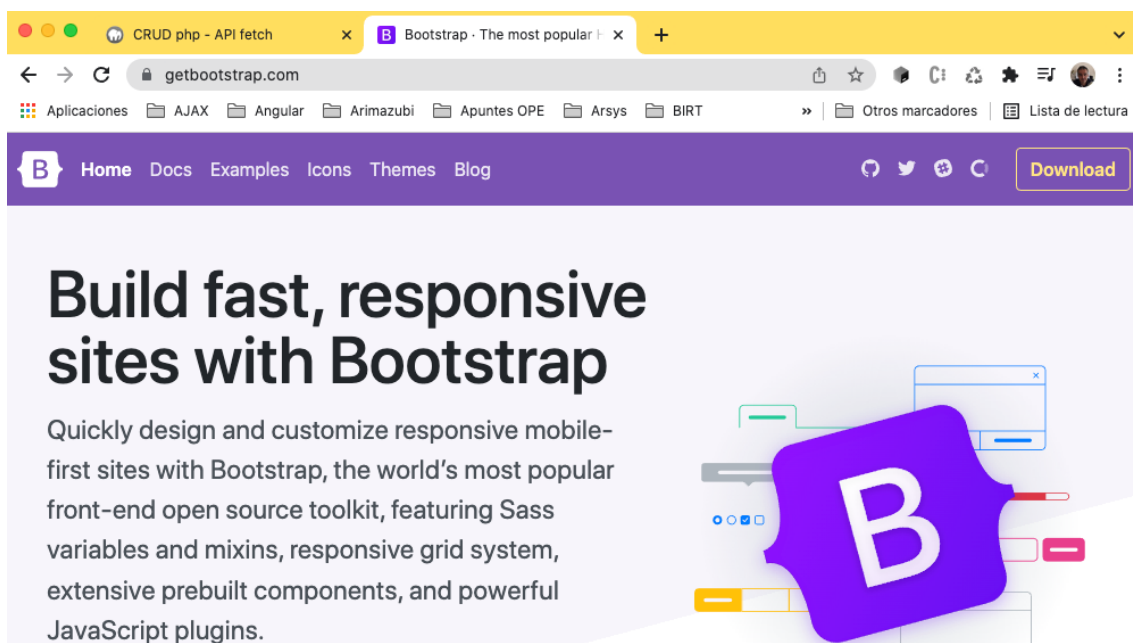
Paso 2: Diseño de la homepage:

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CRUD fetch</title>
  <script src="js/app.js"></script>
</head>
<body>
  </body>
</html>

```

Para los estilos usamos bootstrap. En el head colocamos la importación de las librerías de bootstrap. Solo copiamos la librería de css:



Tras las modificaciones:

```
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CRUD fetch</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
</head>
<body>
```

También importaremos las librerías sweetalert2:



En el final de la página:

```
<link href="//cdn.jsdelivr.net/npm/@sweetalert2/theme-dark@4/dark.css" rel="stylesheet">
<script src="//cdn.jsdelivr.net/npm/sweetalert2@11/dist/sweetalert2.min.js"></script>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
```

Ahora en el body de la página creamos esta estructura:

```
14 <div class="container">
15   <div class="row">
16     <div class="col-lg-4">
17       <div class="card">
18         <div class="card-header bg-primary">
19           <h3 class="text-center">Registro de productos</h3>
20         </div>
21         <div class="card-body">
```

Cada uno de los inputs.

```
<div class="card-body">
  <form action="" method="post" id="frm">
    <div class="form-group">
      <label for="">Codigo</label>
      <input type="hidden" name="idp" id="idp" value="">
      <input type="text" name="codigo" id="codigo" placeholder="Codigo" class="form-control">
    </div>
```

Guardamos y abrimos en el navegador:

The screenshot shows a web browser window with the address bar displaying 'localhost:8888/ajax/crudfetch/'. The page title is 'Registro de productos'. The form contains the following fields and labels:

- Codigo**: Input field with placeholder 'Codigo'.
- Producto**: Input field with placeholder 'Descripción'.
- precio**: Input field with placeholder 'Precio'.
- Cantidad**: Input field with placeholder 'cantidad'.
- Registrar**: A blue button at the bottom of the form.

La siguiente columna es para mostrar el listado del contenido de la base de datos:

```
<div class="col-lg 8">
  <table class="table table-hover table-responsive">
    <thead class="thead-dark">
      <tr>
        <th>ID</th>
        <th>Descripción</th>
        <th>Precio</th>
        <th>Cantidad</th>
        <th>Acciones</th>
      </tr>
    </thead>
    <tbody id="resultado">
    </tbody>
  </table>
</div>
```

Resultado:

The screenshot shows the same web browser window as before, but now with a table displayed below the form. The table has the following headers:

ID	Descripción	Precio	Cantidad	Acciones
----	-------------	--------	----------	----------

The form fields and the 'Registrar' button remain visible on the left side of the page.

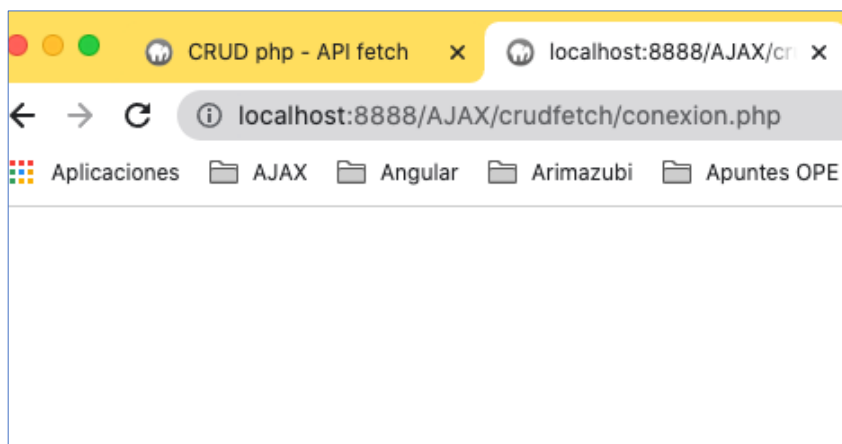
### Paso 3. Conexión con la base de datos mysql.

Creamos el archivo conexion.php:

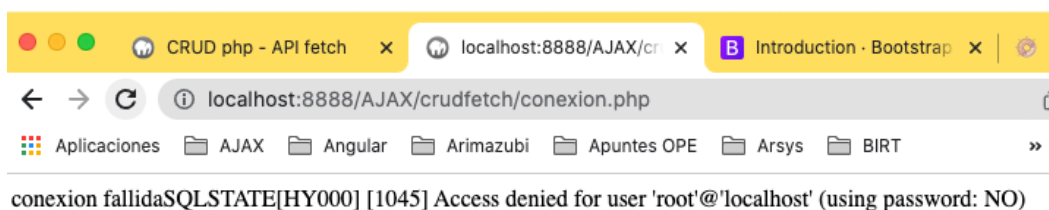
conexion.php

```
<?php
    $servidor = "mysql:dbname=crudfetch;host=localhost";
    $user = "root";
    $pass = "root";
    try {
        $pdo = new PDO($servidor, $user, $pass,
            array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8"));
    } catch (PDOException $e) {
        echo "conexion fallida" . $e->getMessage();
    }
?>
```

Para probarlo, lo colocamos en la url directamente:



No muestra ningún mensaje porque no hay errores. Si cambio la contraseña:



#### Paso 4: Trabajar con el método registrar producto.

Colocamos nuestro fichero app.js:

```

65 |
66 |     </div>
67 |
68 |     <script src="js/app.js"></script>
69 |
70 | </body>
71 | </html>

```

Capturamos el evento click del botón registrar:

```

JS app.js  ●  registrar.php  conexion.php
js > JS app.js > registrar.addEventListener("click") callback
1  registrar.addEventListener("click", () =>{
2      fetch("registrar.php",{
3
4      })
5  });
6

```

Definimos los parámetros:

```

JS app.js  ●  registrar.php  conexion.php
js > JS app.js > ...
1  registrar.addEventListener("click", () =>{
2      fetch("registrar.php",{
3          method: "POST",
4          body: new FormData(frm)
5      }).then( response =>
6          response.text()).
7          then(response=>{
8              console.log(response);
9          })
10 });

```

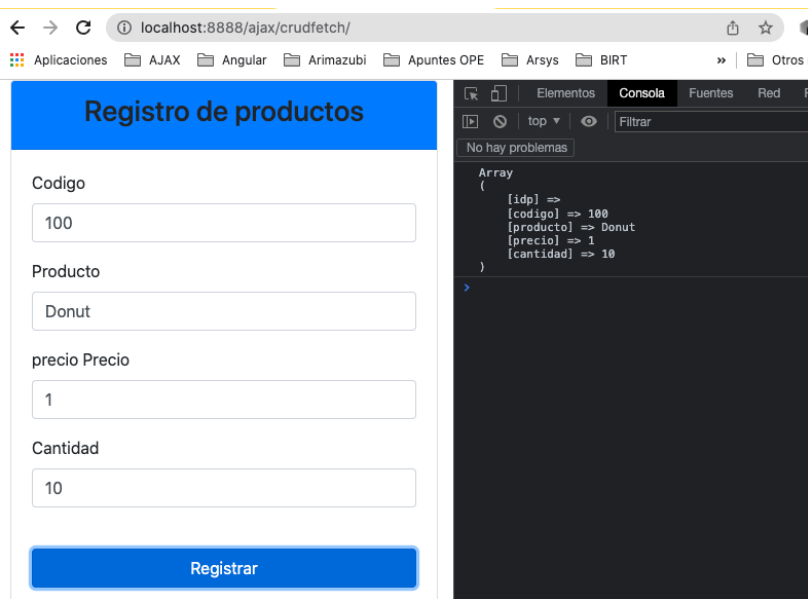
Simplemente registrar.php mostraremos la información pasada mediante POST:

```

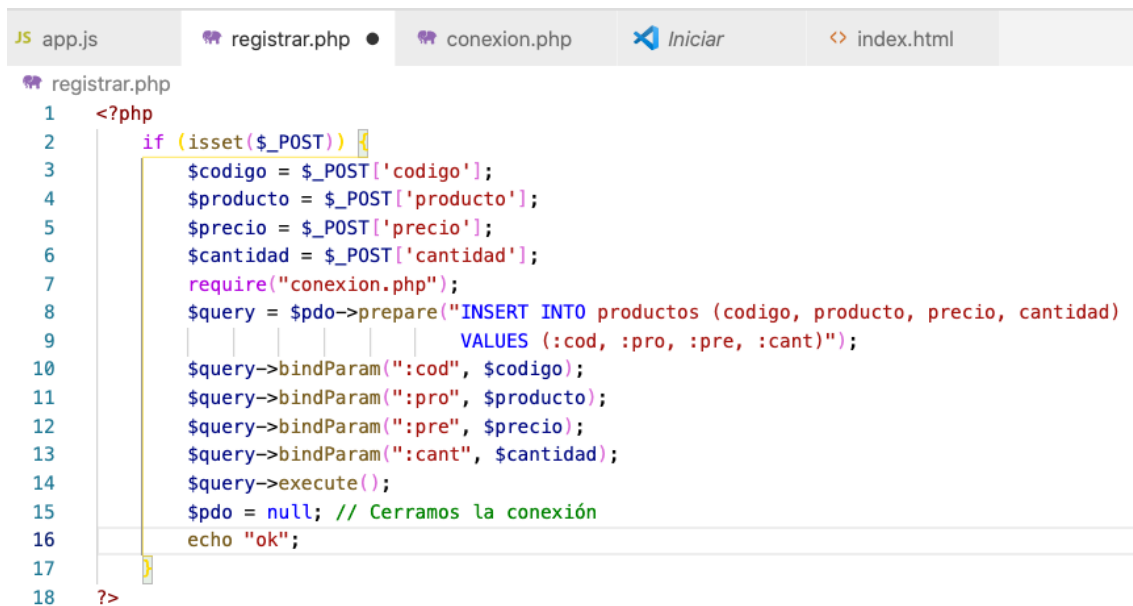
JS app.js  registrar.php  ×
registrar.php
1  <?php
2      print_r($_POST);
3  ?>

```

En el navegador y en la consola:



Este arreglo lo vamos a tratar en `registrar.php`, accediendo a cada uno de los:



Ahora en nuestro `app.js` mostramos la respuesta mediante un sweet alert.



Entonces:

```
JS app.js > ...
  registrar.addEventListener("click", () =>{
    fetch("registrar.php",{
      method: "POST",
      body: new FormData(frm)
    }).then( response =>
      response.text()).
      then(response=>{
        if( response == "ok"){
          Swal.fire({
            icon: 'success',
            title: 'Registrado',
            showConfirmButton: false,
            timer: 1500
          })
        }
      })
  });
```

Para limpiar el formulario una vez insertado hacemos frm.reset():

```

    then(response=>{
      if( response == "ok"){
        Swal.fire({
          icon: 'success',
          title: 'Registrado',
          showConfirmButton: false,
          timer: 1500
        })
      }
    })
    frm.reset();
```

Comprobamos que en la base de datos está el donut (el registro insertado).

esta tabla	Sort by key:	Ninguna			
id	codigo	producto	precio	cantidad	
10	8523	televisor lg	85.00	9	
11	963	laptop hp	56.00	9	
18	7897685	nuevo	896.00	6	
19	75545	probando	96344.00	8	
20	78799	impresora epson	45.00	4	
24	199	Donut	1.00	100	



Paso 5: Trabajar con el método listar productos.

En app.js creamos una función para listar todos los productos de la base de datos (listarProductos).

```
> JS app.js > ...
1 function listarProductos(){
2     fetch("listar.php",{
3         method: "POST"
4     })
5 }
-
```

Debemos tratar la respuesta que devuelve listar.php:

```
> JS app.js > listarProductos
1 function listarProductos(){
2     fetch("listar.php",{
3         method: "POST"
4     }).then(response => response.text()).then(
5         response => {
6             console.log(response);
7         })
8 }
```

El script php de listar es:

```
listar.php
1 <?php
2     require "conexion.php";
3     $consulta = $pdo->prepare("SELECT * FROM productos ORDER BY id DESC");
4     $consulta->execute();
5
6     $resultado = $consulta->fetchAll(PDO::FETCH_ASSOC);
7     foreach ($resultado as $data) {
8         echo "<tr>
9             <td>" . $data['id'] . "</td>
10            <td>" . $data['producto'] . "</td>
11            <td>" . $data['precio'] . "</td>
12            <td>" . $data['cantidad'] . "</td>
13            <td>
14                <button type='button' class='btn btn-success'>Editar</button>
15                <button type='button' class='btn btn-danger' >Eliminar</button>
16            </td>
17            </tr>";
18 }
```

Esta función listarProductos la vamos a llamar al principio del script app.js para que cargue la tabla:

```
> JS app.js > ...
1 listarProductos();
2
3 function listarProductos(){
```

Guardamos y cargamos:

Registro de productos

Codigo

Codigo

Producto

Descripción

precio Precio

Precio

Cantidad

cantidad

Registrar

ID	Descripción	Precio	Cantidad	Acciones
----	-------------	--------	----------	----------

El esquema de color preferido por el sistema ha cambiado. Para aplicar este cambio a DevTools, vuelve a cargarlo.

Volver a cargar DevTools

```

<tr>
  <td>24</td>
  <td>donut</td>
  <td>1.00</td>
  <td>100</td>
  <td>
    <button type="button" class="btn btn-success" onclick=editar('24')>editar</button>
    <button type="button" class="btn btn-danger" onclick=eliminar('24')>eliminar</button>
  </td>
</tr>
<tr>
  <td>20</td>
  <td>impresora epson</td>
  <td>45.00</td>
  <td>4</td>
  <td>
    <button type="button" class="btn btn-success" onclick=editar('20')>editar</button>
    <button type="button" class="btn btn-danger" onclick=eliminar('20')>eliminar</button>
  </td>
</tr>
<tr>
  <td>19</td>
  <td>probando</td>
  <td>96344.00</td>
  <td>8</td>
  <td>
    <button type="button" class="btn btn-success" onclick=editar('19')>editar</button>
    <button type="button" class="btn btn-danger" onclick=eliminar('19')>eliminar</button>
  </td>
</tr>
<tr>
  <td>18</td>
  <td>nuevo</td>
  <td>896.00</td>
  <td>6</td>
  <td>
    <button type="button" class="btn btn-success" onclick=editar('18')>editar</button>
    <button type="button" class="btn btn-danger" onclick=eliminar('18')>eliminar</button>
  </td>
</tr>
<tr>
  <td>11</td>
  <td>laptop hp</td>
  <td>56.00</td>
  <td>9</td>
  <td>
    <button type="button" class="btn btn-success" onclick=editar('11')>editar</button>
    <button type="button" class="btn btn-danger" onclick=eliminar('11')>eliminar</button>
  </td>
</tr>
<tr>
  <td>10</td>
  <td>televisor lg</td>
  <td>85.00</td>
  <td>9</td>
  <td>
    <button type="button" class="btn btn-success" onclick=editar('10')>editar</button>
    <button type="button" class="btn btn-danger" onclick=eliminar('10')>eliminar</button>
  </td>
</tr>

```

Esta salida la vamos a agregar a la tabla de id=resultado que está en index.php en vez de mostrarlo por consola:

```

app.js > listarProductos > then() callback
listarProductos();

function listarProductos(){
  fetch("listar.php",{
    method: "POST"
  }).then(response => response.text()).then(
    response => {
      resultado.innerHTML = response;
    }
  )
}

```

Guardamos, cerramos la consola y actualizamos la página:

CRUD fetch

crudfetch/

Arimazubi

Apuntes OPE

Arsys

BIRT

Coronavirus

DAM

Docker

ejercicios JAVA

Ethazi

Euskara

Fetch

GitHub-Ionic

Registro de productos

Codigo

Codigo

Producto

Descripción

precio Precio

Precio

Cantidad

cantidad

Registrar

ID	Descripción	Precio	Cantidad	Acciones
24	Donut	1.00	100	<div>Editar</div> <div>Eliminar</div>
20	impresora epson	45.00	4	<div>Editar</div> <div>Eliminar</div>
19	probando	96344.00	8	<div>Editar</div> <div>Eliminar</div>
18	nuevo	896.00	6	<div>Editar</div> <div>Eliminar</div>
11	laptop hp	56.00	9	<div>Editar</div> <div>Eliminar</div>
10	televisor lg	85.00	9	<div>Editar</div> <div>Eliminar</div>

También llamamos a la función **listarproductos** desde la función registrar para actualizar el contenido de la tabla tras una inserción:

```

12  registrar.addEventListener("click", () =>{
13      fetch("registrar.php",{
14          method: "POST",
15          body: new FormData(frm)
16      }).then( response =>
17          response.text()).
18          then(response=>{
19              if( response == "ok"){
20                  Swal.fire({
21                      icon: 'success',
22                      title: 'Registrado',
23                      showConfirmButton: false,
24                      timer: 1500
25                  })
26              }
27          })
28      listarProductos();
29      frm.reset();
30  });

```

Ahora cada vez que insertemos un producto se reflejará automáticamente en la tabla. Si insertamos un reloj appleWatch:

ID	Descripción	Precio	Cantidad	Acciones	
26	Apple watch	450.00	100	Editar	Eliminar
25	HP Pavillion	1200.00	10	Editar	Eliminar
20	impresora epson	45.00	4	Editar	Eliminar
11	laptop hp	56.00	9	Editar	Eliminar
10	televisor lg	85.00	9	Editar	Eliminar

A estas alturas todavía nos falta implementar editar, eliminar y buscar ya que insertar y leer ya están implementados.

## Paso 6: Trabajar con el método eliminar productos.

Asignamos eventos a los botones.

```
<td>
  <button type='button' class='btn btn-success'>Editar</button>
  <button type='button' class='btn btn-danger' onclick='eliminar('.$data['id'].')' >Eliminar</button>
</td>
```

La data la debemos concatenar para la b'squeda:

```
<button type='button' class='btn btn-danger' onclick='eliminar('.$data['id'].')' >Eliminar</button>
```

Nos vamos a app.js y creamos la función eliminar utilizando librerías de sweet alert2.:

A confirm dialog, with a function attached to the "Confirm"-button

Try me!

```
Swal.fire({
  title: 'Are you sure?',
  text: "You won't be able to revert this!",
  icon: 'warning',
  showCancelButton: true,
  confirmButtonColor: '#3085d6',
  cancelButtonColor: '#d33',
  confirmButtonText: 'Yes, delete it!'
}).then((result) => {
  if (result.isConfirmed) {
    Swal.fire(
      'Deleted!',
      'Your file has been deleted.',
      'success'
    )
  }
})
```

La función queda:

```
32 function eliminar(id){
33   Swal.fire({
34     title: 'Are you sure?',
35     text: "You won't be able to revert this!",
36     icon: 'warning',
37     showCancelButton: true,
38     confirmButtonColor: '#3085d6',
39     cancelButtonColor: '#d33',
40     confirmButtonText: 'Yes, delete it!'
41   }).then((result) => {
42     if (result.isConfirmed) {
43       Swal.fire(
44         'Deleted!',
45         'Your file has been deleted.',
46         'success'
47       )
48     }
49   })
50 }
```

Ahora lo arreglamos un poco. Lo primero será traducirlos al español:

```

38  function Eliminar(id) {
39      Swal.fire({
40          title: 'Esta seguro de eliminar?',
41          icon: 'warning',
42          showCancelButton: true,
43          confirmButtonColor: '#3085d6',
44          cancelButtonColor: '#d33',
45          confirmButtonText: 'Si!',
46          cancelButtonText: 'NO'
47      }).then((result) => {
48          if (result.isConfirmed) {
49              fetch("eliminar.php", {
50                  method: "POST",
51                  body: id
52              }).then(response => response.text()).then(response => {
53                  if (response == "ok") {
54                      ListarProductos();
55                      Swal.fire({
56                          icon: 'success',
57                          title: 'Eliminado',
58                          showConfirmButton: false,
59                          timer: 1500
60                      })
61                  }
62              })
63          }
64      })
65  }
66  }

```

La salida es:

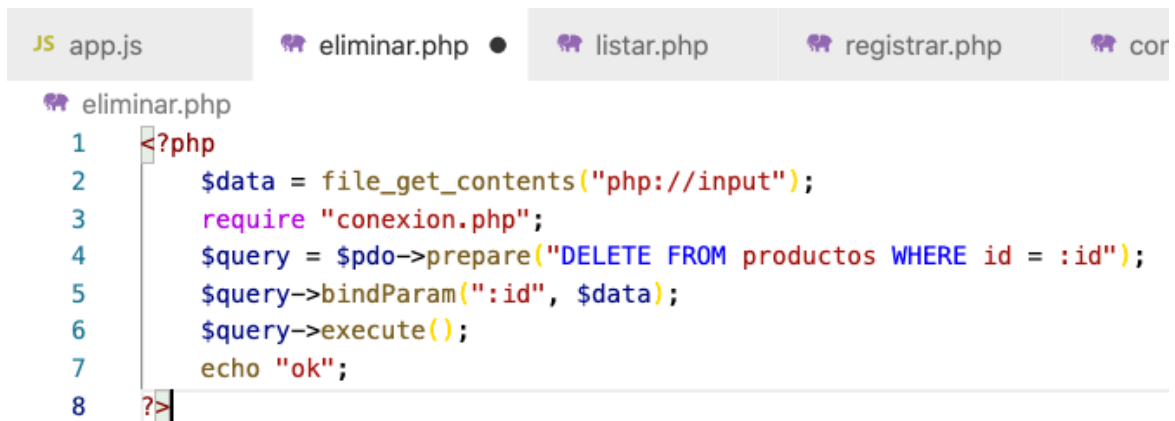


**Esta seguro de eliminar?**

Si!

NO

Creamos el archivo eliminar.php:



```

1  <?php
2  $data = file_get_contents("php://input");
3  require "conexion.php";
4  $query = $pdo->prepare("DELETE FROM productos WHERE id = :id");
5  $query->bindParam(":id", $data);
6  $query->execute();
7  echo "ok";
8  ?>

```

Ahora lo invocamos desde nuestro app.js en la promesa de la respuesta de la llamada fetch (.then):

```

41  }).then((result) => {
42      if (result.isConfirmed) {
43          fetch("eliminar.php",{
44              method: "POST",
45              body: id
46          }).then(response => response.text()).then( response =>{
47              if(response == "ok"){
48                  console.log(response);
49                  Swal.fire(
50                      'Eliminado!'
51                  )
52              }
53          })
54      }
55  })
56  }

```

La salida por consola:



```

ok
app.js:48

```

Luego debo actualizar la página para que se vea la actualización (sustituto el console.log por listarProductos):

```

if (result.isConfirmed) {
  fetch("eliminar.php",{
    method: "POST",
    body: id
  }).then(response => response.text()).then( response =>{
    if(response == "ok"){
      listarProductos();
      Swal.fire({
        icon: 'success',
        title: 'Eliminado',
        showConfirmButton: false,
        timer: 1500
      })
    }
  })
}
}

```

Probamos y tras borrar algunos registros:

### Registro de productos

Codigo

Producto

precio Precio

Cantidad

Registrar

ID	Descripción	Precio	Cantidad	Acciones
36	Samsung Gear S4	300.00	100	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
35	Tablet Lenovo	200.00	100	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>

## Paso 7: Trabajar con el método actualizar productos.

En listar.php hay que agregar el evento click al botón edit. En listar.php en el código de los botones:

```

<button type="button" class="btn btn-success" onclick=editar("'" . $data['id'] . "'>Editar</button>
<button type="button" class="btn btn-danger" onclick=eliminar("'" . $data['id'] . "'>Eliminar</button>

```

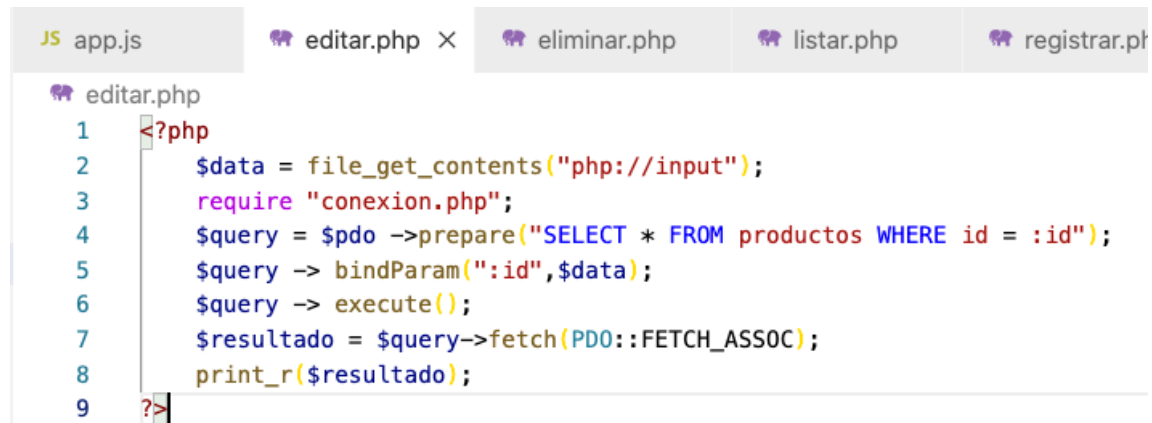
La función editar en app.js:

```

32 > function eliminar(id){...
60 }
61
62 function editar(id){
63     fetch("editar.php",{
64         method: "POST",
65         body: id
66     }).then(response => response.text()).then(
67         response => console.log(response)
68     )
69 }

```

La función editar.php



```

1 <?php
2 $data = file_get_contents("php://input");
3 require "conexion.php";
4 $query = $pdo ->prepare("SELECT * FROM productos WHERE id = :id");
5 $query -> bindParam(":id",$data);
6 $query -> execute();
7 $resultado = $query->fetch(PDO::FETCH_ASSOC);
8 print_r($resultado);
9 ?>

```

La salida:

```

Array
(
    [id] => 36
    [codigo] => 40
    [producto] => Samsung Gear S4
    [precio] => 300.00
    [cantidad] => 100
)

```

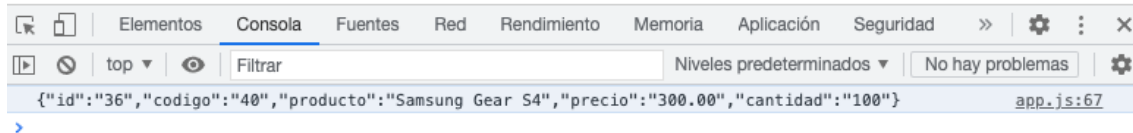


Ahora vamos a enviarlo en un JSON en editar.php para que app.js trate la respuesta:

```

7      $resultado = $query->fetch(PDO::FETCH_ASSOC);
8      echo json_encode($resultado);
9  
```

La salida:



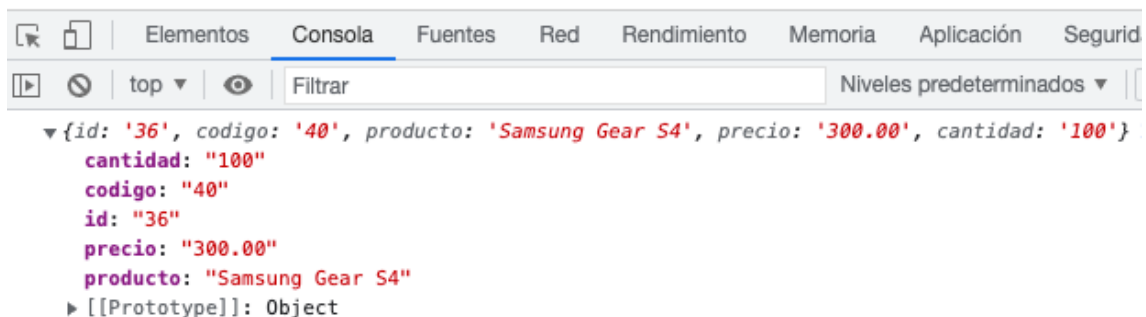
Para utilizarlo hay que hacer estas modificaciones en app.js, de response.text() a response.json():

```

62  function editar(id){
63      fetch("editar.php",{
64          method: "POST",
65          body: id
66      }).then(response => response.json()).then(
67          response => console.log(response)
68      )
69  }

```

En la consola podemos comprobar que ya tenemos un objeto JSON:



Ahora vamos a acceder a cada uno de los elementos para colocarlos en el formulario a través del objeto response:

```

62  function editar(id){
63      fetch("editar.php",{
64          method: "POST",
65          body: id
66      }).then(response => response.json()).then(
67          response => {
68              codigo.value = response.codigo;
69              producto.value = response.producto;
70              precio.value = response.precio;
71              cantidad.value = response.cantidad;
72          }
73      )
74  }

```

Tras darle al botón “Editar” vemos como se cargan los valores del formulario con los valores de la base de datos:

### Registro de productos

Código

Producto

precio Precio

Cantidad

Registrar

ID	Descripción	Precio	Cantidad	Acciones
36	Samsung Gear S4	300.00	100	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>

Debemos cambiar el value del botón de “Registrar” a “Actualizar” incluimos el idp (identificador de producto) en la llamada (`idp.value = response.id`).

```

70 function Editar(id) {
71     fetch("editar.php", {
72         method: "POST",
73         body: id
74     }).then(response => response.json()).then(response => {
75         idp.value = response.id;
76         codigo.value = response.codigo;
77         producto.value = response.producto;
78         precio.value = response.precio;
79         cantidad.value = response.cantidad;
80         registrar.value = "Actualizar"
81     })
82 }

```

Resultado de los últimos campos:

precio Precio

Cantidad

Actualizar

En registrar.php podemos ver que si existe el idp del registro se trata se una operación de actualización, si no es una inserción.

```

registrar.php
1  <?php
2  if (isset($_POST)) {
3      $codigo = $_POST['codigo'];
4      $producto = $_POST['producto'];
5      $precio = $_POST['precio'];
6      $cantidad = $_POST['cantidad'];
7      require("conexion.php");
8      if (empty($_POST['idp'])){
9          $query = $pdo->prepare("INSERT INTO productos (codigo, producto, precio, cantidad)
10             VALUES (:cod, :pro, :pre, :cant)");
11          $query->bindParam(":cod", $codigo);
12          $query->bindParam(":pro", $producto);
13          $query->bindParam(":pre", $precio);
14          $query->bindParam(":cant", $cantidad);
15          $query->execute();
16          $pdo = null;
17          echo "ok";
18      }else{
19          $idp = $_POST['idp'];
20          $query = $pdo->prepare("UPDATE productos SET
21             codigo = :cod,
22             producto = :pro,
23             precio =:pre,
24             cantidad = :cant WHERE id = :idp");
25          $query->bindParam(":cod", $codigo);
26          $query->bindParam(":pro", $producto);
27          $query->bindParam(":pre", $precio);
28          $query->bindParam(":cant", $cantidad);
29          $query->bindParam("id", $id);
30          $query->execute();
31          $pdo = null;
32          echo "modificado";
33      }
34  }
35  }

```

Utilizamos dos queries PHP de modo que uno nos sirve para insertar y el otro para actualizar.

Ahora hacemos una modificación en la función registrar en app.js para incorporar las modificaciones y que el texto del botón cambie según el tipo de operación:

```
registrar.addEventListener("click", () => {
  fetch("registrar.php", {
    method: "POST",
    body: new FormData(frm)
  }).then(response => response.text()).then(response => {
    if (response == "ok") {
      Swal.fire({
        icon: 'success',
        title: 'Registrado',
        showConfirmButton: false,
        timer: 1500
      })
      frm.reset();
      ListarProductos();
    } else {
      Swal.fire({
        icon: 'success',
        title: 'Modificado',
        showConfirmButton: false,
        timer: 1500
      })
    }
    registrar.value = "Registrar";
    ListarProductos();
    frm.reset();
  });
});
```

Una vez actualizado el registro el botón debe contener el valor de registrar (el último idp utilizado). Pero tal como está si hemos actualizado e intentamos insertar un nuevo registro, no lo vamos a conseguir porque sólo actualiza el último registro actualizado en vez de insertar.

### Registro de productos

Codigo

Producto

precio Precio

Cantidad

Registrar

ID	Descripción	Precio	Cantidad	Acciones
48	Pencil	100.00	100	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
40	HP	120.00	120	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
36	Samsung Gear	300.00	120	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>

Le damos a registrar:

ID	Descripción	Precio	Cantidad	Acciones	
49	Trash	100.00	10	Editar	Eliminar
48	Pencil	100.00	100	Editar	Eliminar
40	HP	120.00	120	Editar	Eliminar
36	Samsung Gear	300.00	120	Editar	Eliminar

Y en vez de insertar un registro actualiza el último insertado. Tal y como advertimos para evitarlo, hacemos `idp.value = ""`

```

registrar.addEventListener("click", () => {
    fetch("registrar.php", {
        method: "POST",
        body: new FormData(frm)
    }).then(response => response.text()).then(response => {
        if (response == "ok") {
            Swal.fire({
                icon: 'success',
                title: 'Registrado',
                showConfirmButton: false,
                timer: 1500
            })
            frm.reset();
            ListarProductos();
        } else {
            Swal.fire({
                icon: 'success',
                title: 'Modificado',
                showConfirmButton: false,
                timer: 1500
            })
            registrar.value = "Registrar";
            ListarProductos();
            idp.value = "";
            frm.reset();
        }
    })
});

```

Ahora podemos insertar sin problemas. Nos falta el buscador.

## Paso 8: Trabajar con el método buscar productos.

Para agregar el buscador a nuestro crud, vamos a index.php e insertamos una caja de texto para introducir la cadena de búsqueda.:

```
<div class="row">
  <div class="col-lg-6">
    <form action="" method="post">
      <div class="form-group">
        <label for="buscar">
          Buscar:
        </label>
        <input type="text" name="buscar" id="buscar"
          placeholder="Buscar...">
      </div>
    </form>
  </div>
</div>
```

Cargamos la página:

The screenshot shows a web browser at localhost:8888/ajax/fetchcrud/. The page has a blue header 'Registro de productos'. On the left, there are input fields for 'Codigo', 'Producto' (with a description placeholder), and 'precio Precio'. On the right, there is a search bar labeled 'Buscar:' and a table of products.

ID	Descripción	Precio	Cantidad	Acciones
52	79798789	10.00	10	<button>Editar</button> <button>Eliminar</button>
51	block	10.00	10	<button>Editar</button> <button>Eliminar</button>
50	Printer	400.00	10	<button>Editar</button> <button>Eliminar</button>

Lo modificamos para posicionarlo a la derecha (`class = "ml-auto"`):

```
<div class="col-lg-8">
  <div class="row">
    <div class="col-lg-6 ml-auto">
      <form action="" method="post">
        <div class="form-group">
          <label for="buscar">
            Buscar:
          </label>
          <input type="text" name="buscar" id="buscar"
            placeholder="Buscar..." class="form-control">
        </div>
      </form>
    </div>
  </div>
</div>
```

Cargamos de nuevo:

Registro de productos

Buscar:

Debemos añadir un evento para cada vez que se escriba en la caja de texto (keyup):

```

82 buscar.addEventListener("keyup", ()=>{
83     const valor = buscar.value;
84     if(valor== ""){
85         listarProductos();
86     }else{
87         listarProductos(valor);
88     }
89 });

```

En app.js en la función listarProductos mostramos la respuesta en el formulario:

```

2 function listarProductos(búsqueda) {
3     fetch("listar.php", {
4         method: "POST",
5         body: búsqueda
6     }).then(response => response.text()).then(response => {
7         resultado.innerHTML = response;
8     })
9 }

```

En listar.php defino la variable data (el contenido de la caja de texto) y la muestro:

```

1 <?php
2 $data = file_get_contents("php://input");
3
4 print_r($data);
5 require "conexion.php";
6 $consulta = $pdo->prepare("SELECT * FROM productos ORDER BY id DESC");
7 $consulta->execute();
8 $resultado = $consulta->fetchAll(PDO::FETCH_ASSOC);

```

Probamos:

Buscar:

ID	Descripción	Precio	Cantidad	Acciones
block				
52	79798789	10.00	10	<div>Editar</div> <div>Eliminar</div>

Después de ejecutar la consulta debemos crear una condicional que verifique que data es diferente de vacío:

```
listar.php
1  <?php
2  $data = file_get_contents("php://input");
3
4  require "conexion.php";
5  $consulta = $pdo->prepare("SELECT * FROM productos ORDER BY id DESC");
6  $consulta->execute();
7
8  if ($data !== "") {
9      $consulta = $pdo->prepare("SELECT * FROM productos WHERE id LIKE '%" . $data . "%'
0      OR producto LIKE '%" . $data . "%' OR precio LIKE '%" . $data . "%'");
1      $consulta->execute();
2  }
3  $resultado = $consulta->fetchAll(PDO::FETCH_ASSOC);
4  .
```

Luego para mostrarla en formato tabla hacemos como en apartados anteriores:

```
foreach ($resultado as $data) {
    echo "<tr>
        <td>" . $data['id'] . "</td>
        <td>" . $data['producto'] . "</td>
        <td>" . $data['precio'] . "</td>
        <td>" . $data['cantidad'] . "</td>
        <td>
            <button type='button' class='btn btn-success' onclick=editar('" . $data['id'] . "')>Editar</button>
            <button type='button' class='btn btn-danger' onclick=eliminar('" . $data['id'] . "')>Eliminar</button>
        </td>
    </tr>";
}
```

El resultado de esta consulta es:

Buscar:

ID	Descripción	Precio	Cantidad	Acciones
53	ACME	10.00	78	<div> <div>Editar</div> <div>Eliminar</div> </div>