

Desarrollo web en entorno cliente - Tema 8. API REST

Objetivos

- Mecanismos de comunicación asíncrona.
- Objetos, propiedades y métodos relacionados.
- Recuperación remota de la información.
- Programación de aplicaciones con comunicación asíncrona.
- Modificación dinámica del documento utilizando comunicación asíncrona.
- Formatos para el envío y recepción de información.
- Librerías de actualización dinámica.

Contenido

- Bibliografía

1. Introducción

2. PHP con mysqli

3. PHP con PDO

Bibliografía

- Aprender a desarrollar con JavaScript - 2ª edición
 - Christian Vigouroux (2018) Eni
- Aprendiendo JavaScript: Desde cero hasta ECMAScript 6
 - Carlos Azaustre (2016) carlosazaustre.es
- https://developer.mozilla.org/es/docs/Web/API/Fetch_API/Using_Fetch

Contenido

- Bibliografía

1. Introducción

2. PHP con mysqli

3. PHP con PDO


1. Introducción

- La **API Fetch** proporciona una interfaz JavaScript para acceder y manipular partes del canal HTTP, tales como peticiones y respuestas. También provee un método global `fetch()` (en-US) que proporciona una forma fácil y lógica de obtener recursos de forma asíncrona por la red.
- Este tipo de funcionalidad se conseguía previamente haciendo uso de **XMLHttpRequest**. **Fetch** proporciona una alternativa mejor que puede ser empleada fácilmente por otras tecnologías como **Service Workers** (en-US). **Fetch** también aporta un único lugar lógico en el que definir otros conceptos relacionados con HTTP como CORS y extensiones para HTTP.

1. Introducción

- Una petición básica de **fetch** es realmente simple de realizar. Eche un vistazo al siguiente código:

```
fetch('http://example.com/movies.json')  
  .then(response => response.json())  
  .then(data => console.log(data));
```



Recibimos la respuesta en response y se la pasamos en formato json, luego el data la muestra por pantalla

1. Introducción

- Aquí estamos recuperando un archivo JSON a través de red e imprimiendo en la consola. El uso de `fetch()` más simple toma un argumento (la ruta del recurso que quieres obtener) y devuelve un objeto `Promise` conteniendo la respuesta, un objeto `Response`.
- Esto es, por supuesto, una respuesta HTTP no el archivo JSON. Para extraer el contenido en el cuerpo del JSON desde la respuesta, usamos el método `json()` (en-US) (definido en el mixin de `Body`, el cual está implementado por los objetos `Request` y `Response`).

1. Introducción

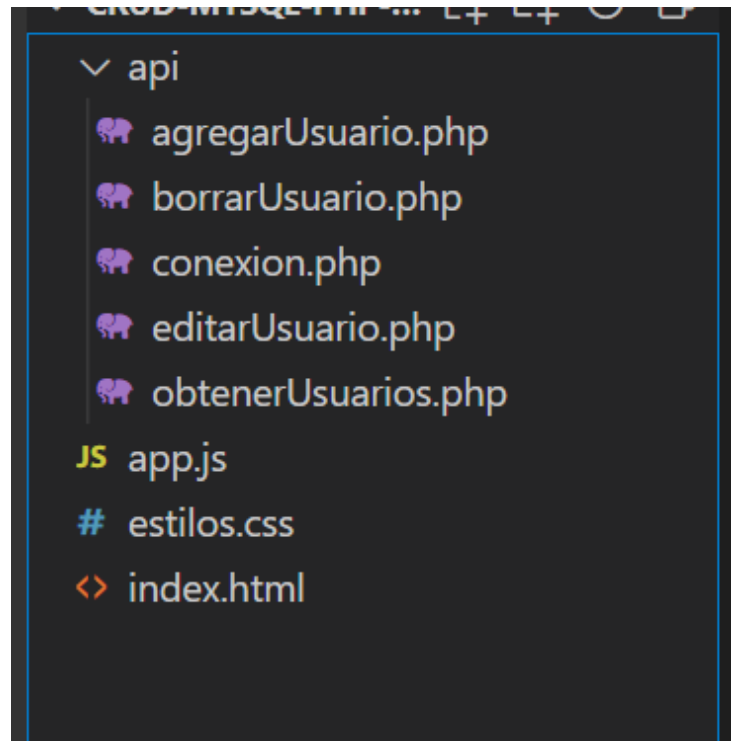
- Resto de documentación se puede encontrar aquí:
 - https://developer.mozilla.org/es/docs/Web/API/Fetch_API/Using_Fetch

Contenido

- Bibliografía
- 1. Introducción
- 2. PHP con mysqli
- 3. PHP con PDO














2. PHP con mysqli









- Está es la estructura que se va a tener. Se va a realizar un CRUD con la tabla usuarios en MySQL




2. PHP con mysqli

- Está es la estructura que se va a tener. Se va a realizar un CRUD con la tabla **usuarios** en MySQL

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/> 1	idUsuario 	int(11)			No	Ninguna		AUTO_INCREMENT	 Cambiar  Eliminar  Más
<input type="checkbox"/> 2	usuario	varchar(255)	latin1_swedish_ci		No	Ninguna			 Cambiar  Eliminar  Más
<input type="checkbox"/> 3	contrasena	varchar(255)	latin1_swedish_ci		No	Ninguna			 Cambiar  Eliminar  Más
<input type="checkbox"/> 4	email	varchar(255)	latin1_swedish_ci		No	Ninguna			 Cambiar  Eliminar  Más

 ☐ Seleccionar todo Para los elementos que están marcados:  Examinar  Cambiar  Eliminar  Primaria  Único  Índice  Texto con

 Eliminar de las columnas seleccionadas

2. PHP con mysqli

```
<?php

// Variables de la conexion a la DB
$mysqli = new mysqli("localhost","root","", "Tema8.Ejemplo1");

// Comprobamos la conexion
if($mysqli->connect_errno) {
    die("Fallo la conexion");
} else {
    //echo "Conexion exitosa";
}
```

Conexión

```
<?php
```

```
    header('Access-Control-Allow-Origin: *');  
    header("Access-Control-Allow-Headers: X-API-KEY, Origin, X-Requested-With,  
Content-Type, Accept, Access-Control-Request-Method");  
    header("Access-Control-Allow-Methods: GET, POST, OPTIONS, PUT, DELETE");  
    header("Allow: GET, POST, OPTIONS, PUT, DELETE");
```

```
require "conexion.php";
```

```
$sql = "SELECT * FROM usuarios";  
$query = $mysqli->query($sql);
```

```
$datos = array();
```

```
while($resultado = $query->fetch_assoc()) {  
    $datos[] = $resultado;  
}
```

```
echo json_encode($datos);  
//echo json_encode(array("usuarios" => $datos));
```

Obtener Usuarios

```
<?php
```

```
    header('Access-Control-Allow-Origin: *');  
    header("Access-Control-Allow-Headers: X-API-KEY, Origin, X-Requested-With, Content-  
Type, Accept, Access-Control-Request-Method");  
    header("Access-Control-Allow-Methods: GET, POST, OPTIONS, PUT, DELETE");  
    header("Allow: GET, POST, OPTIONS, PUT, DELETE");  
  
    require "conexion.php";  
  
    $json = file_get_contents("php://input");  
  
    $objEmpleado = json_decode($json);  
  
    $sql = "INSERT INTO usuarios(usuario, contrasena, email) VALUES('$objEmpleado->  
>usuario', '$objEmpleado->contrasena', '$objEmpleado->email')";  
  
    $query = $mysqli->query($sql);  
  
    $jsonRespuesta = array('msg' => 'OK');  
    echo json_encode($jsonRespuesta);
```

AgregarUsuario

```
header('Access-Control-Allow-Origin: *');
header("Access-Control-Allow-Headers: X-API-KEY, Origin, X-Requested-With, Content-
Type, Accept, Access-Control-Request-Method");
header("Access-Control-Allow-Methods: GET, POST, OPTIONS, PUT, DELETE");
header("Allow: GET, POST, OPTIONS, PUT, DELETE");

require "conexion.php";

$json = file_get_contents("php://input");

$objEmpleado = json_decode($json);

$sql = "UPDATE usuarios SET usuario='$objEmpleado->usuario',
contrasena='$objEmpleado->contrasena', email='$objEmpleado->email' WHERE
idUsuario='$objEmpleado->idUsuario'";

$query = $mysqli->query($sql);

$jsonRespuesta = array('msg' => 'OK');
echo json_encode($jsonRespuesta);
```

Actualizar usuarios


```
<?php
```

```
    header('Access-Control-Allow-Origin: *');  
    header("Access-Control-Allow-Headers: X-API-KEY, Origin, X-Requested-With,  
Content-Type, Accept, Access-Control-Request-Method");  
    header("Access-Control-Allow-Methods: GET, POST, OPTIONS, PUT, DELETE");  
    header("Allow: GET, POST, OPTIONS, PUT, DELETE");
```

```
require "conexion.php";
```

```
$json = file_get_contents("php://input");
```

```
$objId = json_decode($json);
```

```
$sql = "DELETE FROM usuarios WHERE idUsuario='$objId->idUsuario'";  
$query = $mysqli->query($sql);
```

```
$jsonRespuesta = array('msg' => 'OK');  
echo json_encode($jsonRespuesta);
```


Eliminar

```
const urlObtenerUsuarios = './api/obtenerUsuarios.php'  
const urlAgregarUsuario = './api/agregarUsuario.php'  
const urlEditarUsuario = './api/editarUsuario.php'  
const urlBorrarUsuario = './api/borrarUsuario.php'
```

JS


```
let listaEmpleados = []  
const objEmpleado = {  
  idUsuario: '',  
  usuario: '',  
  contraseña: '',  
  email: ''  
}
```

Array y objeto
donde guardar los
datos al
recuperarlos de la
BBDD




```
let editando = false  
const formulario = document.querySelector('#formulario')  
const usuarioInput = document.querySelector('#usuario')  
const contraseñaInput = document.querySelector('#contrasena')  
const emailInput = document.querySelector('#email')  
formulario.addEventListener('submit', validarFormulario)
```

Obtener los
campos del
formulario



```
function validarFormulario(e) {  
    e.preventDefault()  
  
    if([usuarioInput.value, contraseñaInput.value, emailInput.value].includes('')) {  
        alert('Todos los campos son obligatorios')  
        return  
    }  
  
    if(editando) {  
        editarEmpleado()  
        editando = false  
    } else {  
        objEmpleado.idUsuario = Date.now()  
        objEmpleado.usuario = usuarioInput.value  
        objEmpleado.contrasena = contraseñaInput.value  
        objEmpleado.email = emailInput.value  
  
        agregarEmpleado()  
    }  
}
```

Validar que los
campos estén
reellenos y si vamos
a editar o agregar



2. PHP con mysqli

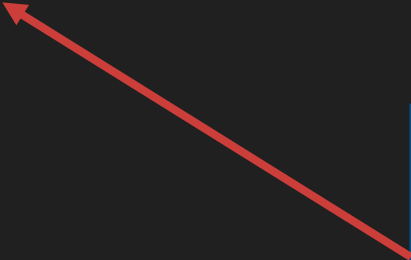
```
async function obtenerEmpleados() {  
  
    listaEmpleados = await fetch(urlObtenerUsuarios)  
    .then(respuesta => respuesta.json())  
    .then(datos => datos)  
    .catch(error => console.log(error))  
    mostrarEmpleados()  
  
}  
obtenerEmpleados()
```

Await permite esperar hasta obtener respuesta de la BBDD, de ese modo no sigue la ejecución hasta obtenerla


```
function mostrarEmpleados() {  
  const divEmpleados = document.querySelector('.div-empleados')  
  listaEmpleados.forEach(empleado => {  
    const {idUsuario, usuario, contrasena, email} = empleado  
    const parrafo = document.createElement('p')  
    parrafo.textContent = `${idUsuario} - ${usuario} - ${contrasena} - ${email}`  
    parrafo.dataset.id = idUsuario  
  
    const editarBoton = document.createElement('button')  
    editarBoton.onclick = () => cargarEmpleado(empleado)  
    editarBoton.textContent = 'Editar'  
    editarBoton.classList.add('btn', 'btn-editar')  
    parrafo.append(editarBoton)  
  
    const eliminarBoton = document.createElement('button');  
    eliminarBoton.onclick = () => eliminarEmpleado(idUsuario);  
    eliminarBoton.textContent = 'Eliminar';  
    eliminarBoton.classList.add('btn', 'btn-eliminar');  
    parrafo.append(eliminarBoton);  
    const hr = document.createElement('hr')  
    divEmpleados.appendChild(parrafo)  
    divEmpleados.appendChild(hr)  
  })  
}
```

Mostramos los
empleados de la
lista. Cada
empleado tiene un
botón para
eliminar o editar

```
async function agregarEmpleado() {  
  
    const res = await fetch(urlAgregarUsuario,  
        {  
            method: 'POST',  
            body: JSON.stringify(objEmpleado)  
        })  
        .then(respuesta => respuesta.json())  
        .then(data => data)  
        .catch(error => alert(error))  
  
    if(res.msg === 'OK') {  
        alert('Se registro exitosamente')  
        limpiarHTML()  
        obtenerEmpleados()  
  
        formulario.reset()  
        limpiarObjeto()  
    }  
}
```



Indicamos que es
método POST



Limpiamos el
formulario y
obtenemos los
nuevos datos

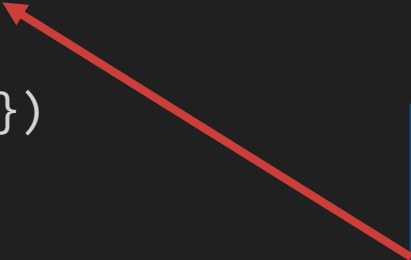
```
async function editarEmpleado() {  
  objEmpleado.usuario = usuarioInput.value  
  objEmpleado.contrasena = contrasenaInput.value  
  objEmpleado.email = emailInput.value  
  const res = await fetch(urlEditarUsuario, {  
    method: 'POST',  
    body: JSON.stringify(objEmpleado)  
  })  
  .then(respuesta => respuesta.json())  
  .then(data => data)  
  .catch(error => alert(error))  
  
  if(res.msg === 'OK') {  
    alert('Se actualizó correctamente')  
    limpiarHTML()  
    obtenerEmpleados()  
    formulario.reset()  
    limpiarObjeto()  
  }  
  formulario.querySelector('button[type="submit"]').textContent = 'Agregar';  
  editando = false  
}
```

Indicamos que es método POST. En el body pasamos el objeto


Limpiamos el formulario y obtenemos los nuevos datos

Cambiamos editando a false

```
async function eliminarEmpleado(id) {  
  
    const res = await fetch(urlBorrarUsuario,  
        {  
            method: 'POST',  
            body: JSON.stringify({'idUsuario': id})  
        })  
    .then(respuesta => respuesta.json())  
    .then(data => data)  
    .catch(error => alert(error))  
  
    if(res.msg === 'OK') {  
        alert('Se borró exitosamente')  
  
        limpiarHTML()  
        obtenerEmpleados()  
        limpiarObjeto()  
    }  
  
}
```



Indicamos que es
método POST.
Indicamos en el
body su id



Limpiamos el
formulario y
obtenemos los
nuevos datos

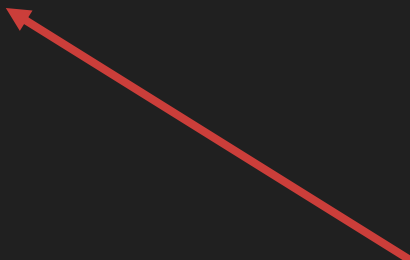

```
function cargarEmpleado(empleado) {  
  const {idUsuario, usuario, contrasena, email} = empleado  
  usuarioInput.value = usuario  
  contrasenaInput.value = contrasena  
  emailInput.value = email  
  objEmpleado.idUsuario = idUsuario  
  formulario.querySelector('button[type="submit"]').textContent = 'Actualizar'  
  editando = true  
}
```

Mostrar empleado



```
function limpiarHTML() {  
  const divEmpleados = document.querySelector('.div-empleados');  
  while(divEmpleados.firstChild) {  
    divEmpleados.removeChild(divEmpleados.firstChild)  
  }  
}
```

Elimina todo
dentro del
contenedor 'div-
empleados'



```
function limpiarObjeto() {  
  objEmpleado.idUsuario = ''  
  objEmpleado.usuario = ''  
  objEmpleado.contrasena = ''  
  objEmpleado.email = ''  
}
```

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>CRUD MySQL, PHP y JS</title>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="./estilos.css">
</head>
<body>
  <div class="div-titulo">
    <h1>Tema 8. Ejemplo 1. CRUD MySQL, PHP y JS</h1>
  </div>

  <div class="contenedor">

    <div class="div-formulario">
      <h2>Formulario</h2>
```

Index.html

```
        <form action="#" id="formulario">
            <input type="text" id="usuario" placeholder="Ingresa tu usuario">
            <input type="text" id="contrasena" placeholder="Ingresa tu
contraseña">
            <input type="text" id="email" placeholder="Ingresa tu email">
            <button type="submit" id="btnAgregar">Agregar</button>
        </form>
    </div>

    <div class="div-listado">
        <h2>Listado Empleados</h2>
        <div class="div-empleados">

            </div>
        </div>

    </div>

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

```

* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
body {
  background-color: bisque;
}
.div-titulo {
  width: 100%;
  height: 100px;
  display: flex;
  justify-content: center;
  align-items: center;
}
.div-titulo h1 {
  font-weight: 300;
}
.contenedor {
  width: 100%;
  height: calc(100vh - 100px);
  display: flex;
  justify-content: center;
}

```

```

/* ESTILOS FORMULARIO */
.contenedor .div-formulario {
  width: 30%;
  height: 40%;
  margin: 15px 15px;
  background-color: rgb(135,
132, 132);
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  border-radius: 10px;
  box-shadow: 10px 10px rgba(0,
0, 0, 0.3);
}

.contenedor .div-formulario h2 {
  margin-top: 10px;
  font-weight: 300;
  font-size: 2em;
  color: whitesmoke;
}

```

CSS

2. PHP con mysqli

```
.contenedor .div-formulario form {
  width: 100%;
  height: 100%;
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
}
.contenedor .div-formulario form
input,
.contenedor .div-formulario form
button {
  width: 60%;
  padding: 10px 10px;
  margin: 10px 10px;
}
```

```
/* ESTILOS LISTADO */
.contenedor .div-listado {
  width: 55%;
  height: calc(100% - 50px);
  margin: 15px 15px;
  background-color: rgb(135,
132, 132);
  display: flex;
  flex-direction: column;
  align-items: center;
  border-radius: 10px;
  box-shadow: 10px 10px rgba(0,
0, 0, 0.3);
}

.contenedor .div-listado h2 {
  margin-top: 10px;
  font-weight: 300;
  font-size: 2em;
  color: whitesmoke;
}
```

```
.contenedor .div-listado .div-  
empleados {  
  width: 100%;  
  height: 100%;  
  display: flex;  
  flex-direction: column;  
  align-items: center;  
  margin: 15px;  
  padding: 0 30px;  
}  
.contenedor .div-listado .div-  
empleados p {  
  margin: 10px 0;  
  font-size: 1em;  
  color: #ddd;  
}  
hr {  
  width: 100%;  
  height: 1px;  
}
```

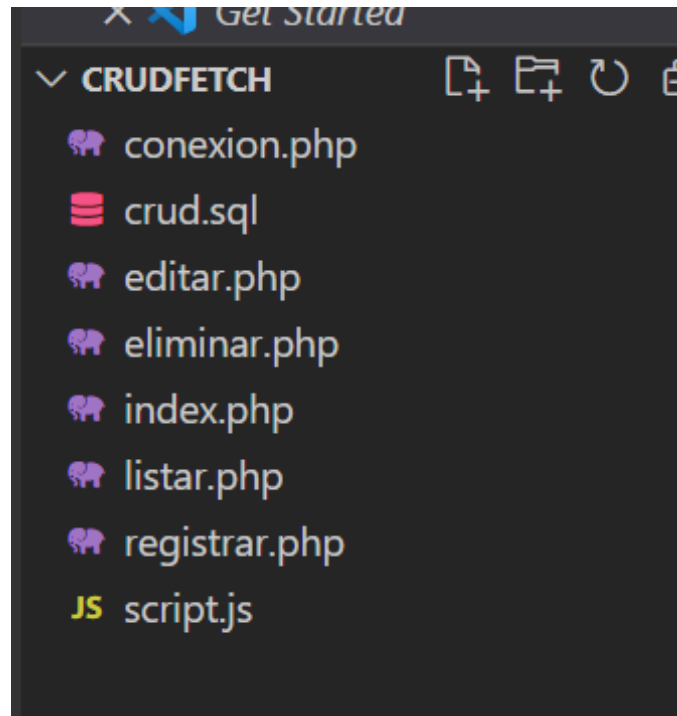
```
.btn {  
  padding: 5px 10px;  
  border: none;  
  margin: 0 5px;  
  color: white;  
  border-radius: 50px;  
}  
.btn-editar {  
  background-color: green;  
}  
.btn-eliminar {  
  background-color: red;  
}
```

Contenido

- Bibliografía
- 1. Introducción
- 2. PHP con mysqli
- 3. PHP con PDO

















3. PHP con PDO

- Mismo esquema, pero distinta forma de realizar el API REST



3. PHP con PDO

- Tabla productos

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/>	1	id 	int(11)			No	Ninguna		AUTO_INCREMENT	 Cambiar  Eliminar  Más
<input type="checkbox"/>	2	codigo	varchar(20)	utf8_spanish_ci		No	Ninguna			 Cambiar  Eliminar  Más
<input type="checkbox"/>	3	producto	varchar(255)	utf8_spanish_ci		No	Ninguna			 Cambiar  Eliminar  Más
<input type="checkbox"/>	4	precio	decimal(10,2)			No	Ninguna			 Cambiar  Eliminar  Más
<input type="checkbox"/>	5	cantidad	int(11)			No	Ninguna			 Cambiar  Eliminar  Más

3. PHP con PDO

```
<?php
    $servidor = "mysql:dbname=Tema8.Ejemplo2;host=localhost";
    $user = "root";
    $pass = "";
    try {
        $pdo = new PDO($servidor, $user, $pass,
array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8"));
    } catch (PDOException $e) {
        echo "conexion fallida" . $e->getMessage();
    }

?>
```

Conexión

```

<?php
$data = file_get_contents("php://input");
require "conexion.php";
$consulta = $pdo->prepare("SELECT * FROM productos ORDER BY id DESC");
$consulta->execute();
if ($data != "") {
    $consulta = $pdo->prepare("SELECT * FROM productos WHERE id LIKE '%" . $data . "%'
OR producto LIKE '%" . $data . "%' OR precio LIKE '%" . $data . "%'");
    $consulta->execute();
}
$resultado = $consulta->fetchAll(PDO::FETCH_ASSOC);
foreach ($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>";
}

```

Para el buscador



listar

```
<?php
if (isset($_POST)) {
    $codigo = $_POST['codigo'];
    $producto = $_POST['producto'];
    $precio = $_POST['precio'];
    $cantidad = $_POST['cantidad'];
    require("conexion.php");
    if (empty($_POST['idp'])) {
        $query = $pdo->prepare("INSERT INTO productos (codigo, producto, precio,
cantidad) VALUES (:cod, :pro, :pre, :cant)");
        $query->bindParam(":cod", $codigo);
        $query->bindParam(":pro", $producto);
        $query->bindParam(":pre", $precio);
        $query->bindParam(":cant", $cantidad);
        $query->execute();
        $pdo = null;
        echo "ok";
    }
}
```

registrar


3. PHP con PDO

```
}else{
    $id = $_POST['idp'];
    $query = $pdo->prepare("UPDATE productos SET codigo = :cod, producto = :pro,
precio =:pre, cantidad = :cant WHERE id = :id");
    $query->bindParam(":cod", $codigo);
    $query->bindParam(":pro", $producto);
    $query->bindParam(":pre", $precio);
    $query->bindParam(":cant", $cantidad);
    $query->bindParam("id", $id);
    $query->execute();
    $pdo = null;
    echo "modificado";
}
}
```

editar

3. PHP con PDO

Hace que se
muestren todos
los datos en el
formulario



```
<?php
    $data = file_get_contents("php://input");
    require "conexion.php";
    $query = $pdo->prepare("SELECT * FROM productos WHERE id = :id");
    $query->bindParam(":id", $data);
    $query->execute();
    $resultado = $query->fetch(PDO::FETCH_ASSOC);
    echo json_encode($resultado);
?>
```

editar

3. PHP con PDO

Eliminar

```
<?php
    $data = file_get_contents("php://input");
    require "conexion.php";
    $query = $pdo->prepare("DELETE FROM productos WHERE id = :id");
    $query->bindParam(":id", $data);
    $query->execute();
    echo "ok";
?>
```

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

JS

Body ponemos los
datos

Librería para hacer
alert:
<https://sweetalert2.github.io/>

Resetar formulario
y cargar productos

3. PHP con PDO

```
    }else{  
        Swal.fire({  
            icon: 'success',  
            title: 'Modificado',  
            showConfirmButton: false,  
            timer: 1500  
        })  
        registrar.value = "Registrar";  
        idp.value = "";  
        ListarProductos();  
        frm.reset();  
    }  
})  
});
```

En caso de editar

Valores a
predeterminados
para poder
registrar

3. PHP con PDO

```
function Eliminar(id) {  
    Swal.fire({  
        title: 'Esta seguro de eliminar?',  
        icon: 'warning',  
        showCancelButton: true,  
        confirmButtonColor: '#3085d6',  
        cancelButtonColor: '#d33',  
        confirmButtonText: 'Si!',  
        cancelButtonText: 'NO'  
    }).then((result) => {  
        if (result.isConfirmed) {  
            fetch("eliminar.php", {  
                method: "POST",  
                body: id  
            }).then(response => response.text()).then(response => {
```


Indicar si está
seguro de eliminar

3. PHP con PDO


```
if (response == "ok") {  
    ListarProductos();  
    Swal.fire({  
        icon: 'success',  
        title: 'Eliminado',  
        showConfirmButton: false,  
        timer: 1500  
    })  
}  
})  
}  
})  
}
```

```
function Editar(id) {  
    fetch("editar.php", {  
        method: "POST",  
        body: id  
    }).then(response => response.json()).then(response => {  
        idp.value = response.id;  
        codigo.value = response.codigo;  
        producto.value = response.producto;  
        precio.value = response.precio;  
        cantidad.value = response.cantidad;  
        registrar.value = "Actualizar"  
    })  
}  
buscar.addEventListener("keyup", () => {  
    const valor = buscar.value;  
    if (valor == "") {  
        ListarProductos();  
    }else{  
        ListarProductos(valor);  
    }  
});
```

Obtener todos los
datos en el
formulario para
editar



Añadimos al botón
buscar su
funcionalidad



3. PHP con PDO

Index.php

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CRUD php - API fetch</title>
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"
crossorigin="anonymous">
</head>
```

```

<body>
  <div class="container">
    <div class="row">
      <div class="col-lg-4">
        <div class="card">
          <div class="card-header bg-primary">
            <h3 class="text-center">Registro de productos</h3>
          </div>
          <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>
              <div class="form-group">
                <label for="">Producto</label>
                <input type="text" name="producto" id="producto"
placeholder="Descripción" class="form-control">
              </div>
            </form>
          </div>
        </div>
      </div>
    </div>
  </div>

```

Id oculto para
pasarlos

3. PHP con PDO

```
        <div class="form-group">precio
            <label for="">Precio</label>
            <input type="text" name="precio" id="precio"
placeholder="Precio" class="form-control">
        </div>
        <div class="form-group">
            <label for="">Cantidad</label>
            <input type="text" name="cantidad" id="cantidad"
placeholder="cantidad" class="form-control">
        </div>
        <div class="form-group">
            <input type="button" value="Registrar" id="registrar"
class="btn btn-primary btn-block">
        </div>
    </form>
</div>
</div>
</div>
```

3. PHP con PDO

```
<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>
```



```

        <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>
</div>
<script src="script.js"></script>
<script src="https://cdn.jsdelivr.net/npm/sweetalert2@10"></script>
</body>

</html>

```