

PHP Bases de datos CRUD (Create, Read, Update, Delete)

1. De acuerdo al patrón de arquitectura requerido, usaremos index.html para la presentación.
2. Hasta ahora tenemos el tablero con listado ordenamiento y filtro.
3. Agregamos HTML, estilos CSS y el código JS para obtener la funcionalidad necesaria para implementar CRUD.

Tareas:

HTML: Contenedor tabla ,contenedor formulario de alta, contenedor formulario de modi, contenedor de respuestas del servidor.

CSS: Definir estilos para los formularios de alta y modi, el contenedor de respuestas del servidor y las ventanas modales correspondientes.

Definir estilos para las nuevas columnas de botones modi y botones baja.

JS: Definir objetos para los nuevos elementos del DOM que intervengan en funciones y procedimientos de java script.

Definir funciones para los nuevos eventos: botón de alta, botones de modi y baja creados dinámicamente en cada ciclo de lectura de los registros JSON.

Definir funciones de evento para los botones de envío de los formularios de alta y de modi.

Definir procedimientos de evento para la validación de datos en los eventos keyUp y change de las entradas de los formularios.

Los formularios de alta y modificación se encuentran ocultos y se activan cuando se pulsan los botones respectivamente

Articulos.				Orden:	<input type="text" value="codArt"/>	<input type="button" value="Cargar datos"/>	<input type="button" value="Vaciar datos"/>	<input type="button" value="Limpiar filtros"/>	<input type="button" value="Alta registro"/>
Cod Art	familia	um	descrip	fecha Alta	Saldo stock	PDFs	Modis	Bajas	
		▼							
ALM00	FRSEC	KG	Almendras Peladas Orig Thai	2022-10-29	20004	PDF	Modi	Borrar	
ALM01	FRSEC	KG	Almendra Non Pareil Origen Filipinas	2021-04-30	153	PDF	Modi	Borrar	

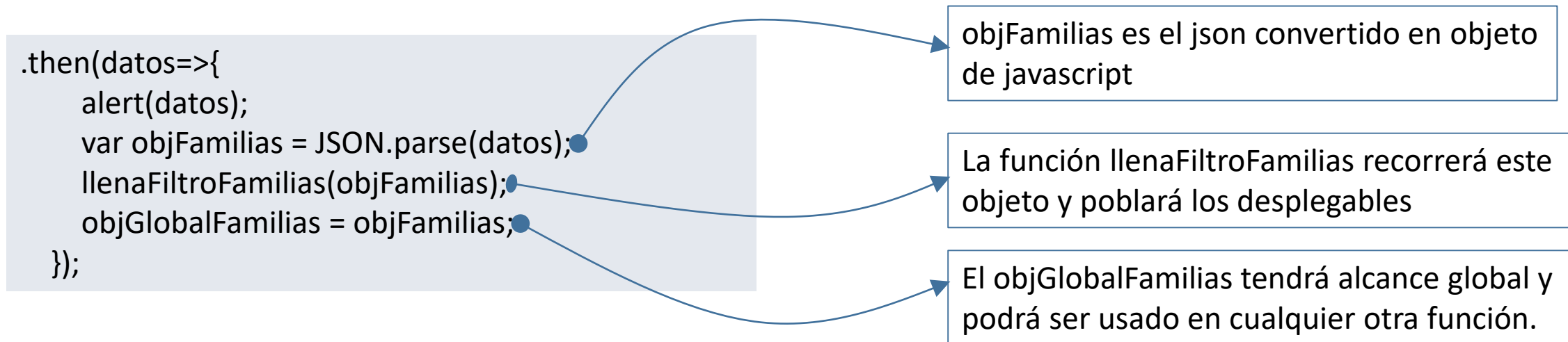
Las funciones para efectivizar las modi y las eliminaciones se construyen dinámicamente en el barrido del array de productos.

Poblamiento de los desplegables en los forms de alta y modi

No solo los filtros sino también los formularios de alta y modificación arman sus campos desplegables leyendo los objetos que contienen las listas desplegables.

Habrà que construir estos objetos a partir de los datos recibidos en json desde el servidor. Estos datos estaràn disponibles tanto para los filtros como para los formularios de alta y de modificaciones.

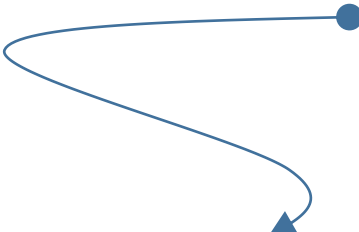
En este ejemplo el objeto objGlobalfamilias es el obtenido para ser utilizado tanto en el filtro como en los forms.



En este ejemplo cuando la cadena de promesas a partir del `fetch()` se cumpla, se podrán procesar los descolgables.

Aplicación del objeto global de familias

El arreglo de familias es barrido para crear dinámicamente los desplegables



```
objGlobalFamilias.familias.forEach(function(argValor,argIndice) {
```

```
    XXXXXXXXXXXXXXXXXXXX
```

```
    XXXXXXXXXXXXXXXXXXXX
```

```
    XXXXXXXXXXXXXXXXXXXX
```

```
    XXXXXXXXXXXXXXXXXXXX
```

```
});
```

**En el arranque cuando se termina de cargar el documento se
declaran y preparan los objetos del tablero:**

`objTbDatos=document.getElementById("tbDatos");`

`objCodArtAlta=document.getElementById("formArticulosEntCodArtAlta");`

.....

`$("#orden").val("saldoStock");` //suponiendo que al comienzo quisiera este orden

`$("#contenedorTablaArticulos").addClass("contenedorActivo");`

`$("#ventanaModalFormularioAlta").css("visibility","hidden");`
`$("#ventanaModalFormularioModi").css("visibility","hidden");`
`$("#ventanaModalRespuesta").css("visibility","hidden");`

`$("#btEnvioFormModi").attr("disabled",true);`
`$("#btEnvioFormAlta").attr("disabled",true);`

Definir objetos para usar con
java script puro.

Definir objetos para usar con
java script puro. En este caso
el input de codArt para el formulario de alta

Asignar valores iniciales como por ej.
El orden inicial para cargar la tabla por primera
vez.

Suponiendo que contenedorTablaArticulos sea el contenedor de
todo el tablero de control del maestro de artículos con encabezado y pie
de tablero además de la tabla en si misma, podríamos activarlo
al comienzo en la carga del documento . Para ello podemos aplicar la clase
contenedorActivo (otorgar opacidad y activar la sensibilidad de objetos frente
a los eventos)

Ocultar las ventanas de formularios y la ventana
de respuesta del server

Deshabilitar los botones de envío de los formularios para que
se vuelvan a habilitar con la función de validación.

Nuevas funciones o handlers para definir procedimientos de evento:

```
$(document).ready(function() {  
    $("#btEnvioFormModi").click(function() {  
        modi();  
    });  
});
```

Funciones de evento para los botones de **envío** de formularios. En este caso se llama a la función que dispara el requerimiento Ajax.

```
$(document).ready(function() {  
    $("#formArticulosEntCodArtAlta").keyup(function() {  
        todoListoParaAlta();  
    });  
}); //cierro ready
```

Funciones de evento para las validaciones. En este caso el keyup para el input codArt en el formulario de Alta.

todoListoParaAlta() habilita los botones de envío, en este caso del form de alta.

Funciones con nombre para definir procedimientos de validación (caso de validar formulario completo)

```
function todoListoParaAlta() {  
    if (document.getElementById("formArticulosAlta").checkValidity()){  
        $("#btEnvioFormAlta").attr("disabled",false);  
    }  
    else {  
        $("#btEnvioFormAlta").attr("disabled",true);  
    }  
}
```

Caso de activación de botón de envío en el formulario de alta.

La función cargaTabla ahora deberá crear columnas para bajas y modificaciones de cada fila:

```
var objTd=document.createElement("td");
```

Creación dinámica de una nueva celda

```
objTd.setAttribute("campo-dato","articulos_btModi");
```

Asignación de atributo campo-dato para la celda.
Es a fines de que todas las columnas tengan su propio atributo campo-dato.

```
objTd.innerHTML("<button class='btCelda'>Modi</button>");
```

Nodo de texto para el botón.

```
objTd.onclick=function() {
```

Manejador de evento para cuando el usuario haga click en el botón de modi

```
$("#contenedorTablaArticulos").addClass("contenedorPasivo");
```

El contenedor de tabla se vuelve pasivo.

```
$("#ventanaModalFormularioModi").css("visibility","visible");
```

La ventana de formulario de modi se hace visible.

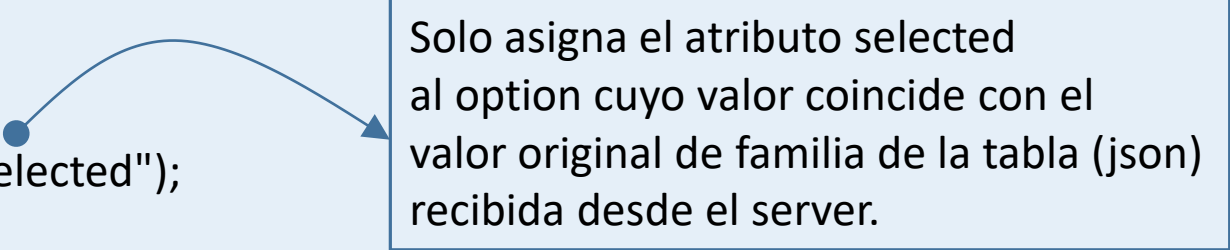
```
CompletaFichaArticulo(argValor);
```

```
};
```

Llama a la función que llena el formulario de modi de artículos con los datos del articulo. Esta función recibe como argumento el objeto articulo con todas sus propiedades que toma valor en este ciclo correspondiente a esta fila.

Función para el llenado de los datos del formulario de modi:
(Esta función recibe como argumento el objeto articulo con todas sus propiedades ya que es llamada desde adentro de un ciclo del barrido del array de artículos arribado la respuesta http al ajax)

```
function CompletaFichaArticulo(argArticulo) {  
  
    $("#formArticulosEntCodArtModi").val(argArticulo.codArt);  
    $("#formArticulosEntUmModi").val(argArticulo.um);  
    ...  
    //Para el desplegable de familias solo se barre el array objGlobalfamilias.familias:  
  
    objGlobalFamilias.familias.forEach(function(argValorFamilia,argIndice) {  
        var objOption= document.createElement("option");  
        objOption.setAttribute("class","elementoOptionSelect");  
        objOption.setAttribute("value", argValorFamilia.codFamilia);  
        objOption.innerHTML=argValorFamilia.codFamilia + argValorFamilia.descripcionFamilia;  
  
        if(objOption.value == argArticulo.familia) {  
            objOption.setAttribute("selected","selected");  
        }  
        document.getElementById("formArticulosEntFamiliaModi").appendChild(objOption);  
    }  
}
```



Solo asigna el atributo selected al option cuyo valor coincide con el valor original de familia de la tabla (json) recibida desde el server.

Funciones alta() , modi() y baja()

Estas funciones podrían solo modificar datos en el servidor. Pero en este ejercicio deberán enviar al cliente parte del estado de dichas operaciones.

- alta()
Produce el disparo de ajax enviando como datos todos los completados en el formulario de alta.
- modi()
Produce el disparo de ajax enviando como datos todos los modificados en el formulario de modi.
- baja()
Produce el disparo de ajax enviando como dato el id de la fila que se desea eliminar.

La respuesta del server:

Las tres funciones mencionadas llevarán en el código ubicado dentro del then() de la ultima promesa la respuesta del servidor y colocación de la misma en el contenedor dispuesto para ese fin.

En este ejemplo el div ventanaModalRespuesta tendrá en su interior un header, un footer y un contenidoModalRespuesta.

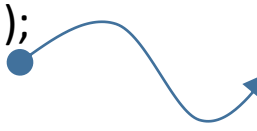
```
$("#ventanaModalRespuesta").css("visibility","visible");  
$("#contenidoModalRespuesta").empty();  
$("#contenidoModalRespuesta").append(respuestaDelServer);
```

En el handler de la cadena de promesas del fetch() se lee la respuesta del server, se hace visible la ventana de respuesta y se adiciona dicha respuesta

Antes de terminar estas funciones se puede volver a ejecutar cargarTabla() para que quede actualizada en el tablero de ABM. No es el caso de este ejercicio donde solo se debe cargar la tabla pulsando el botón designado para tal fin

Generación del requerimiento Ajax. Código similar tanto para el alta() como para la modi()

```
var objFormulario = document.getElementById("formArticulosModi");
var objDatosFormulario = new FormData(objFormulario);
const options = { //opciones de configuracion del fetch
    method:'post',
    headers: {},
    body: objDatosFormulario, //es el body del requerimiento http
}
fetch('./modi.php', options)
.then(respuesta=>{
    return respuesta.text();
})
.then(datos=>{
    alert(datos);//imprime la respuesta al pedido de modi en un simple alert()
    $("#ventanaModalRespuesta").css("visibility","visible");
    $("#contenidoModalRespuesta").empty();
    $("#encabezadoModalRespuesta").append("Respuesta del server: ");
    $("#contenidoModalRespuesta").append(datos);
});
```



imprime la misma respuesta usada en el alert() pero ahora dentro de una ventana modal

Html para el formulario de modi

```
<form id="formArticulosModi" method="post" enctype="multipart/form-data">
<ul>
<li>
<label>codArt: </label>
<input id="formArticulosEntCodArtModi" name="codArt" required />
</li>
<li>
<label>Descripción: </label>
<input id="formArticulosEntDescripcionModi" name="descripcion" required />
</li>
<li>
<label>Familia de artículo: </label>
<select id="formArticulosEntFamiliaModi" name="familia" required></select>
</li>
```

.....

```
<li>
<label>Documento Pdf: </label>
<input type="file" id="formArticulosEntDocumentoPdfModi" name="documentoPdf" />
</li>
</ul>
</form>
```

PHP para la respuesta desde el server al requerimiento

Archivos:

1. alta.php
2. modi.php
3. baja.php
4. error.log
5. traeBinario.php

Atención: Estos scripts definen la funcionalidad de cada modulo dentro de la vista el patrón arquitectónico elegido para el ejercicio. En este caso no usaremos un controlador del lado del server.

Preparación, vinculación y ejecución de sentencia SQL Válido para altas, bajas y modificaciones

Previamente se asigna a una variable la sentencia SQL que corresponda

```
try {  
    $stmt = $dbh->prepare($sql);  
    $respuesta_estado = $respuesta_estado . "\nPreparacion exitosa!";  
} catch (PDOException $e) {  
    $respuesta_estado = $respuesta_estado . "\n" . $e->getMessage();  
}  
try {  
    $stmt->bindParam(':codArt', $codArt);  
    $respuesta_estado = $respuesta_estado . "\nBinding exitoso!";  
} catch (PDOException $e) {  
    $respuesta_estado = $respuesta_estado . "\n" . $e->getMessage();  
}  
try {  
    $stmt->execute();  
    $respuesta_estado = $respuesta_estado . "\nEjecucion exitosa!";  
} catch (PDOException $e) {  
    $respuesta_estado = $respuesta_estado . "\n" . $e->getMessage();  
}
```

Preparación de la sentencia

Asignar una variable para contener todos los datos que creamos necesarios a la hora de depurar errores. Esta variable irá concatenando datos de resultado. \$respuesta_estado solo sera enviada al lado cliente con el objeto de encontrar errores lógicos de resultado incorrecto.

Etapas de vinculación de la sentencia SQL
Se asocian las variables referenciados con «:» en la sentencia cruda, a las variables que realmente contienen los datos.

Ejecución de la sentencia

Script alta.php

```
$respuesta_estado=$respuesta_estado . "\nRespuesta del servidor al alta. Entradas recibidas en el req http:";  
$respuesta_estado=$respuesta_estado . "\ncodArt: " . $codArt;  
$respuesta_estado=$respuesta_estado . "\nfamilia: " . $familia;
```

...

Asignar una variable para contener todos los datos que creamos necesarios a la hora de depurar errores. \$respuesta_estado solo sera enviada al lado cliente con el objeto de encontrar errores lógicos de resultado incorrecto.

```
$sql="insert into articulos (codArt,familia,descripcion,um,fechaAlta,saldoStock)  
values (:codArt,:familia,:descripcion,:um,:fechaAlta,:saldoStock);";
```

...

Sentencia SQL para la inserción de nuevos registros. Observar los valores que se referencian con el carácter «:» Es conveniente no incluir los campos binarios binarios dentro de la sentencia de alta. Lo podemos hacer al final con una sentencia de actualización.

```
$stmt = $dbh->prepare($sql);
```

Preparación de la sentencia que deberá ser anidada en un try/catch para poder capturar una posible excepción

```
$stmt->bindParam(':codArt', $codArt);  
$stmt->bindParam(':familia', $familia);  
$stmt->bindParam(':descripcion', $descripcion);  
...
```

Etapa de vinculación de la sentencia SQL
Se asocian los valores referenciados con «:» en la sentencia cruda,
a las variables que los contienen.

```
$stmt->execute();
```

Ejecución de la sentencia de alta.
Deberá
ser anidada en un try/catch para poder
capturar una posible excepción

Hasta aquí se produjo el alta del nuevo registro pero sin sus atributos de tipo binario como contenido de imagenes o pdf's.

En una segunda etapa realizaremos un update para modificar dichos atributos binarios.

Atributos de tipo binario.

Es conveniente: actualizar los atributos binarios luego de producida el alta y no en el proceso de gestión de la misma.

```
$codArt = $dbh->lastInsertId();
```

Para el caso de que la clave primaria sea autoincremental como por ejemplo alguna tabla de movimientos

```
if(!isset($_FILES['documentoPdf'])) {
```

```
    $respuesta_estado=$respuesta_estado . "No se inicializó global $_FILES";
```

```
}
```

```
else {
```

```
    if (empty($_FILES['documentoPdf']['name'])) {
```

```
        $respuesta_estado = $respuesta_estado . "<br />No ha sido seleccionado ningun file para enviar!";
```

```
    }
```

```
    else {
```

```
        $respuesta_estado=$respuesta_estado . "Trae documentoPdf asociado a codArt: " . $codArt;
```

```
        $contenidoPdf = file_get_contents($_FILES['documentoPdf']['tmp_name']);
```

```
        $sql="update articulos set documentoPdf=:contenidoPdf where codArt=:codArt;";
```

Procesos de preparacion, bind y ejecución de la sentencia de modificación

```
    }
```

```
}
```

EL type de `$_FILES['documentoPdf']` no es una variable simple «string» que contiene el nombre del archivo subido desde el input de java script con nombre documentoPdf sino un array (para verlo se puede usar `var_dump()`. El elemento name en la 2da dimension de `$_FILES` si contiene el nombre de archivo original) .

Si en el req http llegan atributos binarios, entonces se procede a cargar en una variable `$contenidoPdf` el contenido de la imagen o pdf leído.

`$dbh = null; /*para cerrar la conexion*/`

No olvidar cerrar la conexión con el motor de base de datos al final

`echo $respuesta_estado;`

Comentar o descomentar la entrega de la variable que contiene concatenada toda la info para detectar errores lógicos de resultados.
Esto se agrega en el ejercicio para ser mostrado como respuesta de resultado a los procesos de ABM.

Script modi.php

```
$respuesta_estado = "Parte Modificacion simple de datos <br />\n";
```

Asignar una variable para contener todos los datos que creamos necesarios a la hora de depurar errores.
\$respuesta_estado solo sera enviada al lado cliente con el objeto de encontrar errores lógicos de resultado incorrecto.

```
$sql="update articulos set codArt=:codArt,familia=:familia,descripcion=:descripcion,um=:um,  
fechaAlta=:fechaAlta,saldoStock=:saldoStock where codArt=:codArt";
```

Preparación de la sentencia.
Esta deberá ser anidada en un try/catch para poder capturar una posible excepción

```
$stmt = $dbh->prepare($sql);
```

Sentencia SQL para la modificación de registros.
Observar los valores que se referencian con el carácter «:»
Es conveniente no incluir los atributos binarios dentro de la sentencia de modi.
Lo podemos hacer al final con una nueva sentencia de actualización.

```
$stmt->bindParam(':codArt', $codArt);  
$stmt->bindParam(':familia', $familia);
```

Etapas de vinculación de la sentencia SQL
Se asocian los valores referenciados con «:» en la sentencia cruda, a las variables que los contienen.

...

```
$stmt->execute();
```

Ejecución de sentencia
Esta deberá ser anidada en un try/catch para poder capturar una posible excepción

Atributos de tipo binario.

Es conveniente: actualizar los atributos binarios luego de producida el modi de los atributos comunes (no binarios)

```
if(!isset($_FILES['documentoPdf'])) {  
    $respuesta_estado=$respuesta_estado . "No se inicializó global $_FILES";  
}  
else {  
    if (empty($_FILES['documentoPdf']['name'])) {  
        $respuesta_estado = $respuesta_estado . "<br />No ha sido seleccionado ningun file para enviar!";  
    }  
    else {  
        $respuesta_estado=$respuesta_estado . "Trae documentoPdf asociado a codArt: " . $codArt;  
  
        $contenidoPdf = file_get_contents($_FILES['documentoPdf']['tmp_name']);  
  
        $sql="update articulos set documentoPdf=:contenidoPdf where codArt=:codArt;";  
  
        Procesos de preparacion, bind y ejecución de la sentencia de modificación  
  
    }  
}
```

EL type de \$_FILES['documentoPdf'] no es una variable simple «string» que contiene el nombre del archivo subido desde el input de java script con nombre documentoPdf sino un array (para verlo se puede usar var_dump()). El elemento name en la 2da dimension de \$_FILES si contiene el nombre de archivo original) .

Si en el req http llegan atributos binarios, entonces se procede a cargar en una variable \$contenidoPdf el contenido de la imagen o pdf leído.

`$dbh = null; /*para cerrar la conexion*/`

No olvidar cerrar la conexión con el motor de base de datos al final


`echo $respuesta_estado;`

Comentar o descomentar la entrega de la variable que contiene concatenada toda la info para detectar errores lógicos de resultados.
Esto se agrega en el ejercicio para ser mostrado como respuesta de resultado a los procesos de ABM.

Script baja.php

```
$sql = "delete from articulos where codArt=:codArt;";
```


Sentencia SQL de baja



Preparación, vinculación y ejecución de la sentencia.

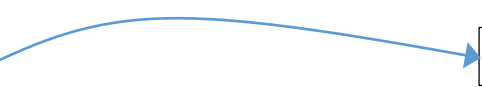
```
$dbh = null;
```

Cerrar la conexión con el motor

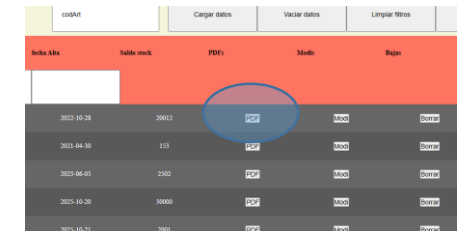


```
echo $respuesta_estado;
```

Devolver estado de la consulta



Script traeBinario.php



Fecha Alta	Fecha Baja	PDF	Estado	Baja
2023-10-28	20012	PDF	Activo	Quitar
2023-10-26	131	PDF	Activo	Quitar
2023-10-26	1302	PDF	Activo	Quitar
2023-10-26	30009	PDF	Activo	Quitar
2023-10-26	13011	PDF	Activo	Quitar

Script previsto para responder al requerimiento de una imagen.

Las líneas de código fundamentales para este proceso son las siguientes:

```
$sql="select documentoPdf from articulos where codArt = :codArt";
```

Sentencia sql que deberá ser procesada en sus tres etapas de preparación, binding y ejecución.

```
$respuesta_estado = $respuesta_estado . "\n<br />Sentencia sql a ser aplicada: " + $sql;
```

```
try {  
    $stmt = $dbh->prepare($sql);  
    $respuesta_estado = $respuesta_estado . "\n<br />preparacion exitosa";  
} catch (PDOException $e) {  
    $respuesta_estado = $respuesta_estado . "\n<br />" . $e->getMessage();  
}
```

..... Luego vienen las funciones de vinculación y ejecución correspondientes

```
$fila=$stmt->fetch();
```

Obtengo los resultados de cada fila que en este caso debería ser una sola de correspondiente al registro del cual se quiere leer el contenido binario

```
$objArticulo = new stdClass();
```

Creo un objeto nuevo para almacenar el articulo

```
$objArticulo->documentoPdf=base64_encode($fila['documentoPdf']);
```

Codifico en base 64
(caracteres de codificados en 6 bits)
para lograr que circule por la red
sin problemas.

```
$salidaJson = json_encode($objArticulo,JSON_INVALID_UTF8_SUBSTITUTE);
```

Codifico en json.
Pero también teniendo en cuenta
que la función de php no agregue
caracteres por fuera del rango de
base 64.

```
$dbh = null
```

Cerrar la conexión con el motor

```
echo $salidaJson;
```

Envio el resultado en la respuesta http