Paginación

30/07/2024

Índice

Creación de arrays iniciales:

Primero realizamos nuestra consulta a la tabla (o tablas) que necesitemos y la guardamos dentro de una variable (`$consultas`).
Luego creamos un array dentro de otra variable en el cual ingresaremos los datos de la consulta (`$tabla_consultas`).

```php
$consultas = $mysqli->query(" ");

$tabla_consultas = array();
```

Para realizar el guardado, revisamos primero que existan datos en la tabla, tras esto, hacemos un bucle while, crearemos una variable nueva en la que se almacenan los datos de las consultas por fila(`$consulta`).

Luego dentro del while, hacemos un `array_push()` en el que guardaremos los datos en el array vacío que creamos antes (`$tabla_consultas`). Es importante que los datos los ingresemos como un nuevo array, pues queremos que los almacene con los índices de dicho array para conocer las filas, y dentro de este nuevo array asignaremos como **clave** el nombre del dato de la tabla, y como **valor** el dato obtenido (esto para tener un orden y un manejo similar a cuando hacemos tablas directamente con las consultas y el `fetch_assoc()`) .

```php
if ($consultas->num_rows > 0) {
    while ($consulta = $consultas->fetch_assoc()) {
        array_push($tabla_consultas, array("columna1" => $consulta['columna1'], "columna2" => $consulta['columna2'], "columna3" => $consulta['columna3'], ... , "culumnaN" => $consulta['culumnaN']));
    }
}
```

En este momento se podría decir que tenemos un array de la siguiente estructura:

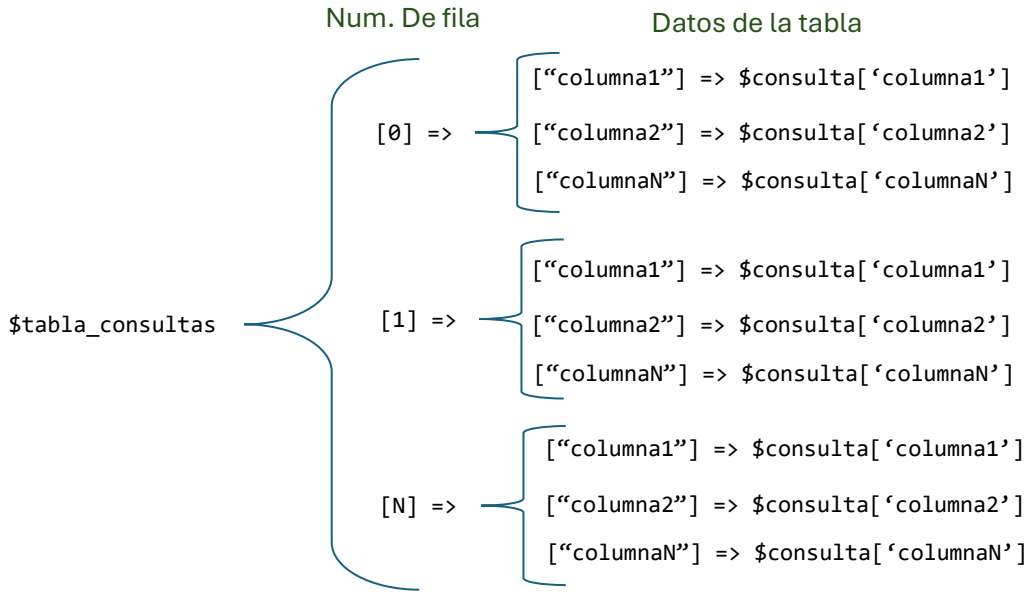

Figura 1.1

Ahora que ya tenemos los datos y las filas, tenemos que saber las páginas que tendremos para la paginación, para esto utilizaremos la función `array_chunck()` (https://www.php.net/manual/es/function.array-chunk.php).

Con esta función lo que haremos es crear un nuevo array el cual contendrá los arrays pasados separados en nuevos arrays, los cuales tendrán el tamaño de las filas que el usuario haya ingresado. Por lo que ahora podemos decir que tenemos un array con las páginas, dentro de las páginas tendremos las filas y dentro de las filas los datos, como se muestra en la figura 1.2 (que no se completaron todos los detalles por cuestiones de espacio, pero se puso lo suficiente para ser representativo).

```php
$tabla_consultas = array_chunk($tabla_consultas, $num_filas);
```

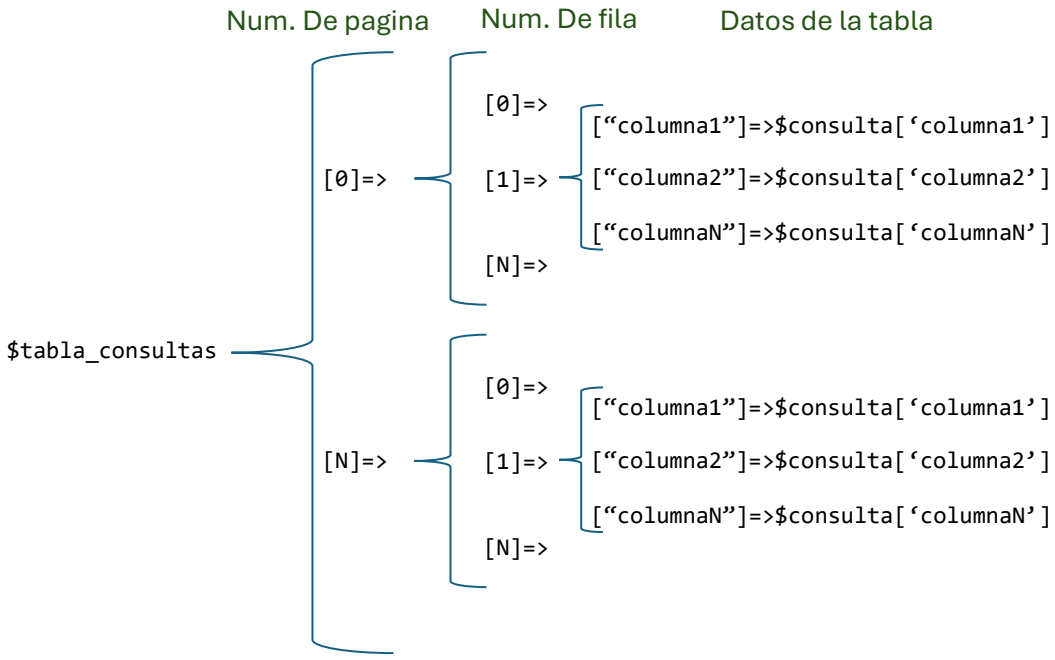

Figura 1.2

Por último, vamos a organizar la paginación, creando grupos de 5 números (puede modificarse dependiendo de las necesidades) para que solo se muestren 9 opciones: adelante, atrás, primera página, última página y los 5 números de cada grupo de páginas. En caso de que haya menos páginas por grupo solo se mostraran las que existan.

Para realizar esta parte primero realizaremos un arreglo del total de páginas que tenemos, para ello usaremos la función `count()` y le restaremos 1, para poder realizar un conteo en forma de array iniciando en cero.

Luego, crearemos un nuevo array vacío (`$grupos`)y con ayuda de un ciclo for le ingresaremos números desde el cero hasta el total de páginas.

Para finalizar, de nuevo usaremos `array_chunck()` para realizar la separación del total de páginas en grupos de 5 filas.

```php
//Guardar cantidad de paginas
$cantidad_pag = count($tabla_consultas) - 1;
//Crear grupos para la paginacion
$grupos = array();    //Crear el array vacío
for ($i = 0; $i <= $cantidad_pag; $i++) {
    array_push($grupos, $i);
}
$grupos = array_chunk($grupos, 5);
```

Quedando el array con la siguiente estructura:

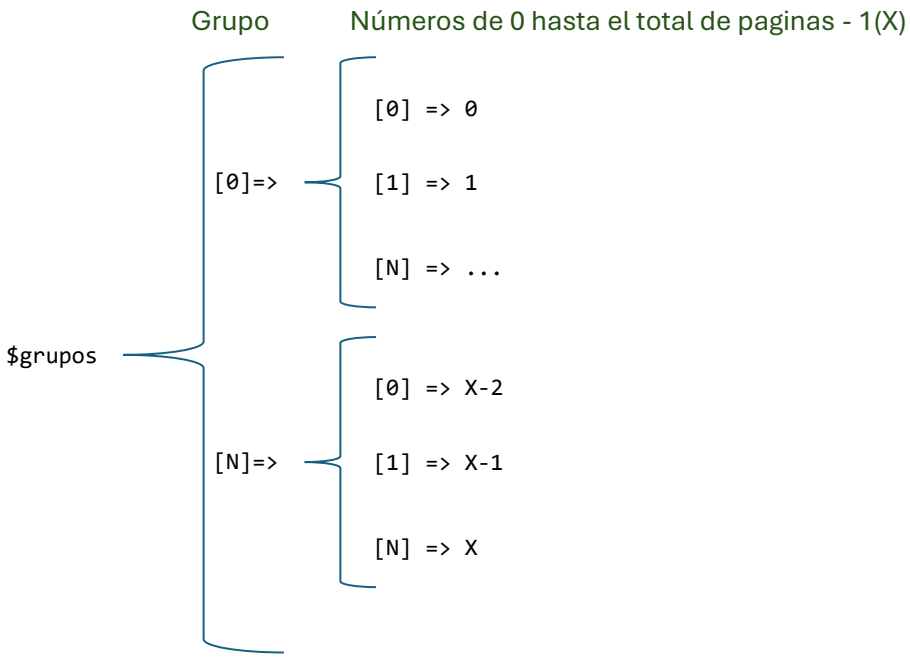

Figura 1.3

Un ejemplo suponiendo que queremos crear 3 filas por grupo teniendo un total de 6 números es el siguiente:

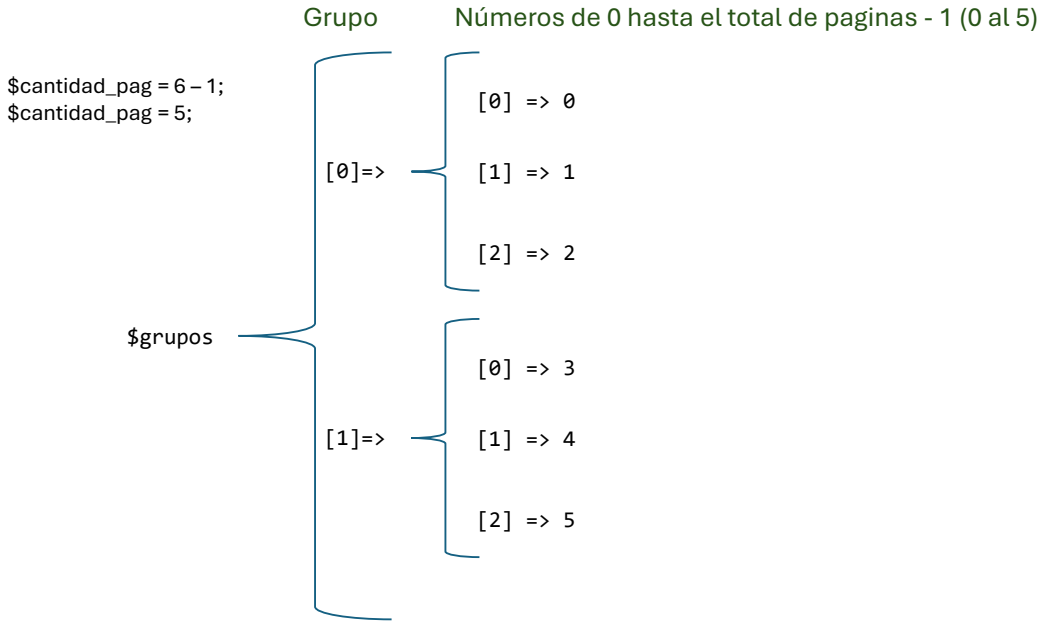

Figura 1.4

Estructura de la paginación:

Opciones número de filas:

Para la paginación primero debemos de darle las opciones al usuario que elija la cantidad de filas que quiere, para el caso default limitamos a que solo pueda escoger 25, 50, 100 y 250 filas. Esto lo logramos con un <select> en el que ingresaremos como opciones los números antes mencionados.

Las opciones deben de ponerse en forma de un array, esto con dos objetivos principales, el primero es que sea más fácil modificar el código en caso de querer agregar o quitar opciones, y el segundo objetivo tiene el fin de poder realizar un barrido con un foreach y agregar un condicional para mostrar y resaltar la opción que sea seleccionada por el usuario, como se muestra en el código.

```
<label>Número de filas: </label>
      <select class="form-control input-sm"
id="num_filas"  onchange="javascript:mostrar_alta(value, 0)" >
            <?php
            $filas = array(25, 50, 100, 250);
            foreach($filas as $opt_num_filas){
                  if($opt_num_filas != $num_filas){
                  ?>
                        <option value="<?= $opt_num_filas?>"><?=
$opt_num_filas?>
                        </option>
                  <?php
                  }else{
                  ?>
                        <option value="<?= $opt_num_filas?>"
                           selected><?= $opt_num_filas?>
                        </option>
                  <?php
                  }
               }
               ?>
      </select>
```

Mostrar las opciones de la paginación (menú de paginación):

https://getbootstrap.com/docs/4.0/components/pagination/

Nosotros no utilizaremos la etiqueta <nav></nav>. Tampoco dentro de las etiquietas <a> usaremos las clases "page-link" o los atributos href debido a que no trabajaremos urls.

Para realizar esta parte debemos de usar dos etiquetas principales:

- : esta etiqueta nos ayuda a crear lista no ordenadas, en nuestro caso es importante agregar el atributo *class="pagination"* para indicar a la librería Bootstrap que haremos una paginación
- : dentro de estas etiquetas ingresaremos los que queremos que se muestre en las paginaciones y lo que se realizará con ellas. Importante agregar el atributo *class="page-item"*
- <a>: estas etiquetas en teoría sirven para ingresar enlaces, en nuestro caso solo la utilizaremos para el formato y dentro de ellas agregamos el atributo *onclick* para dentro ingresar nuestra función de JavaScript que contenga el submitPost() de nuestra tabla.

Dentro de las etiquetas <ul> podemos definir tres secciones de , la primera sección se encargará de las opciones para regresar un grupo o regresar a la primera página que existe, la segunda sección se encargará de mostrar el grupo de números al que pertenezca nuestra página, y la tercera se encargará de mostrar las opciones para avanzar un grupo o avanzar hasta la última página. ($tabla_consultas).

Antes de entrar a las secciones, debemos de descubrir el grupo al que pertenece nuestra página, esto con el fin de saber cuál grupo de números mostrar, y obtener la clave del grupo para poder hacer el cambio por grupos de paginación. Para ello utilizaremos dos foreach anidados. El primero recorrerá todos los valores del array $grupos obteniendo su $key y su $grupo, aquí es donde entra el segundo foreach en el cual recorreremos los valores numéricos que pertenezcan a dicho grupo. Si el numero en el que va el barrido de los foreach ($num) es igual a la página actual ($pagina) se creará una nueva variable llamada $grupo_actual la cual guardará el valor de la variable $grupo, y una variable llamada $key_grupo donde guardaremos su clave ($key) funcionando como una memoria para almacenar el grupo al que pertenece la página actual (nos sirve al avanzar o retroceder por grupos).

```
foreach ($grupos as $key => $grupo) {
    foreach ($grupo as $num) {
        if ($pagina == $num) {
            $grupo_actual = $grupo;
            $key_grupo = $key;
            break;
```

```
                    }
                }
            }
```

1. **Primera sección**:
   En esta encontramos un condicional que se encarga de revisar si la página actual es la primera (o default), si nos encontramos en una página mayor, mostraremos una opción que nos regresa a la página default (<<), mandando un cero a nuestra función que contenga el `submitPost()`. Y mostraremos la segunda opción que se encargará de regresarnos una página atrás (<), mandando la página actual menos 1, y la clave 0 de dicha página a nuestra función que contenga el `submitPost()`.

   En caso contrario, mostraremos las opciones, pero ambas estarán desactivadas. Esto solo para que se vea estético y no solo desaparezcan dichas opciones.

```
<?php if ($pagina > 0) { ?>
            <li class="page-item">
                  <a
onclick="javascript:mostrar_alta(document.getElementById('num_filas').value, <?=
0?>)">
                  <<
                  </a>
            </li>
            <li class="page-item">
                  <a
onclick="javascript:mostrar_alta(document.getElementById('num_filas').value, <?=
$grupos[$key_grupo - 1][0] ?>)">
                  <
                  </a>
            </li>
      <?php
      } else { ?>
            <li class="page-item disabled">
                  <a>
                  <<
                  </a>
            </li>
            <li class="page-item disabled">
                  <a>
                  <
                  </a>
            </li>
```

```
    <?php
    }
```

2. **Segunda sección**:
Aquí mostramos el grupo al que pertenece nuestra página dentro del menú de paginación.

Primero queremos saber el tamaño del grupo al que pertenece nuestra página, usaremos un `count()` (recordando que `$grupo` es un array) y el resultado lo guardaremos en una variable (`$tam_vec_pag`). Con esto podemos usarlo como argumento para un `for`, el cual tiene la función de mostrar los números del grupo en el menú.

Dentro del `for` agregaremos un `if` que tiene la misión de encontrar la página en la que se encuentra el usuario, para ellos solo usamos un condicional que revisa si la variable `$pagina` (recordando que es una variable que guarda la página actual) es igual al número perteneciente al grupo actual en el que se encuentra el `for`, usando nuestro array de grupo actual junto con una clave dada por el `for` (`$grupo_actual[$i]`). En caso de que se cumpla esto, dentro nuestras etiquetas `<li></li>` debemos agregar la clase *class=”page-item active”* para indicar que esa opción de nuestra lista es la seleccionada. Además, dejaremos sin atributos a la etiqueta `<a></a>` pues no queremos que pase nada al dar click en esta opción.

En caso de que no se cumpla la condición de nuestro `if`, haremos que se muestren las opciones del grupo actual con ayuda del `for` y haremos que al dar click en alguna opción, mande el valor de la página clickeada a nuestra función que contenga el `submitPost()`.

```
$tam_vec_pag = count($grupo_actual);
        for ($i = 0; $i < $tam_vec_pag; $i++) {
                if ($pagina == $grupo_actual[$i]) {
                    ?>
                    <li class="page-item active">
                            <a><?= $grupo_actual[$i] + 1 ?></a>
                    </li>
                <?php
                } else {
                ?>
                    <li class="page-item">
                            <a
onclick="javascript:mostrar_alta(document.getElementById('num_filas').value, <?=
$grupo_actual[$i]  ?>)"><?= $grupo_actual[$i] + 1 ?>
                            </a>
                    </li>
```

```
            <?php
            }
    }
```

3. **Tercera sección**: la tercera sección es un espejo de la primera, solo que ahora revisamos si la página en la que nos encontramos es menor a la última página, habilita la opción de avanzar un grupo (mandar a nuestra función que contenga el `submitPost()` el valor de nuestro grupo más uno). Y agregamos la opción de ir a la página final, en la cual mandaremos el valor de la variable `$cantidad_pag` (así como guarda el total de páginas que tenemos, nos dice cuál es nuestra última página) a nuestra función que tenga el `submitPost()`.

   En caso contrario, significa que nos encontramos en nuestra última página, por lo que deshabilitaremos las opciones.

```
if ($pagina < $cantidad_pag) { ?>
                    <li class="page-item">
                        <a
onclick="javascript:mostrar_alta(document.getElementById('num_filas').value, <?=
$grupos[$key_grupo + 1][0] ?>?>)"> +</a>
                    </li>
                    <li class="page-item">
                        <a
onclick="javascript:mostrar_alta(document.getElementById('num_filas').value, <?=
$cantidad_pag ?>)"> >></a>
                    </li>
                <?php
                } else { ?>
                    <li class="page-item disabled">
                        <a> > </a>
                    </li>
                    <li class="page-item disabled">
                        <a> >> </a>
                    </li>
                <?php
                }
                ?>
```

Creación de la tabla:

La tabla se realiza de forma similar a lo común, primero debemos de poner nuestro <thead></thead> con el nombre de las columnas que necesitamos de forma normal. (no olvidar que todo va dentro de etiquetas <table></table >.

```
<thead>
      <tr>
            <th>columna1</th>
            <th>columna2</th>
            <th>columna3</th>
            <th>...</th>
            <th>culumnaN</th>
            <th style="text-align: center;">Boton_funcion</th>
      </tr>
</thead>
```

Lo siguiente es lo que cambia, primero contaremos el tamaño del array `$tabla_consultas` con la primera clave de página usando la variable de nuestra página actual (`$pagina`), esto contará el numero de filas que se encuentran en dicha página (revisar tabla 1.2).

Tras esto realizaremos un `for`, el cual nos ayudará a recorrer todas las filas que pertenezcan a nuestra página. Para todo usaremos nuestro array `$tabla_consultas,` recordando que sus claves representan:

`$tabla_consultas[‘página que queremos consultar’][‘fila de dicha página’][‘Datos que queremos’]`

```
<tbody>
      <?php
      $tam_array = count($tabla_consultas[$pagina]);
      for ($i = 0; $i <= $tam_array - 1; $i++) { ?>
      tr>
            <td> <?= $tabla_consultas[$pagina][$i]['columna1'] ?></td>
            <td> <?= $tabla_consultas[$pagina][$i]['columna2'] ?></td>
            <td> <?= $tabla_consultas[$pagina][$i]['columna3'] ?></td>
            <td> <?= $tabla_consultas[$pagina][$i]['...'] ?></td>
            <td> <?= $tabla_consultas[$pagina][$i]['culumnaN'] ?></td>
            <td align="center">
                  <button type=" button" class="btn btn-mini btn-warning "
onClick="javascript:funcion(<?= $tabla_consultas[$pagina][$i]['columna1'] ?>, <?=
$pagina ?>)"> <span class="glyphicon glyphicon-pencil"></span>
                  </button>
```

```
            </td>
        </tr>
        <?php
        }
        ?>
</tbody>
```

Además, agregamos un botón para poder mostrar la forma en la que se tendrían que mandar los datos, lo cual sería respetando las claves del vector y buscando no confundirnos con ellas.

Posibles problemas:

Problemas relacionados a los arreglos:

El principal problema que creo que se puede encontrar es con los arreglos, tal vez nos falte un +1 o -1 y eso puede ocasionar problemas a la hora de mostrar tanto la tabla como el menú de paginación. Esto se soluciona tomando un tiempo para analizar en si el arreglo y lo que hace falta, tal vez el problema tenga que ver con que estamos haciendo un for a alguna clave que no existe.

Datos de tipo *null*:

En ocasiones al mandar datos a través de funciones como nuestros submitPost o incluso en algunos process podemos tener problemas en que algunos datos nos lleguen *null*, esto generalmente es porque hicimos mal algo al mandar el dato de nuestro arreglo, generalmente un problema relacionado a las claves del arreglo.

La paginación no funciona:

Si nuestra paginación no funciona primero debemos revisar si estamos mandando bien nuestra variable `$pagina` y si estamos haciendo bien lo que queremos con ella, por ejemplo si queremos avanzar una página revisar que si estemos sumando +1 a nuestra variable a la hora de mandarla a nuestra función con el process.

En caso de que no sea un problema de la variable deberíamos revisar si en la tabla estamos poniendo bien nuestra clave de página.

Si esto también está bien el problema va desde la creación de nuestro array, por lo que debemos revisar el principio del código, guiarnos un poco con la sección "Problemas relacionados a los arreglos:"