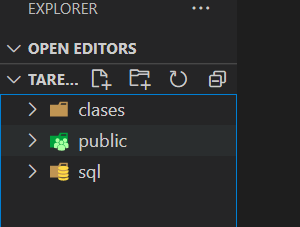
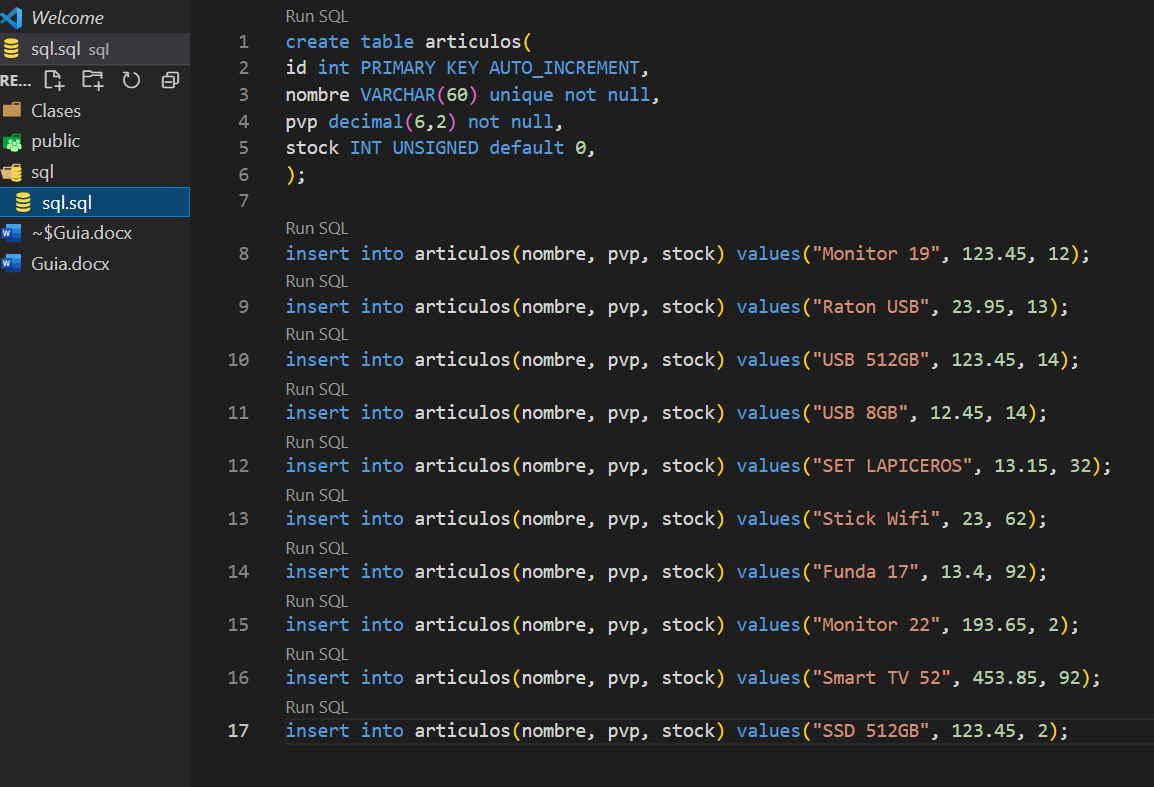
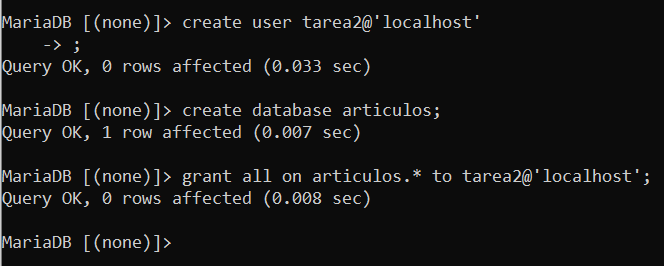
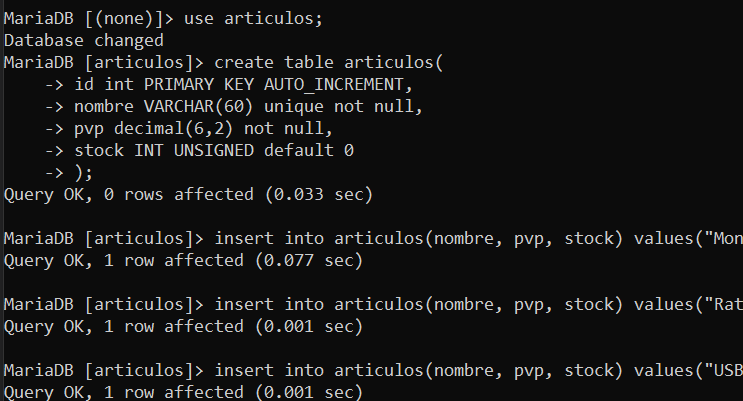
Lo primero que haremos será crear nuestra carpeta del proyecto en la que crearemos una estructura de proyecto con la carpetas usuales (public, src o clases, sql) y en este ultimo meteremos nuestra tabla:



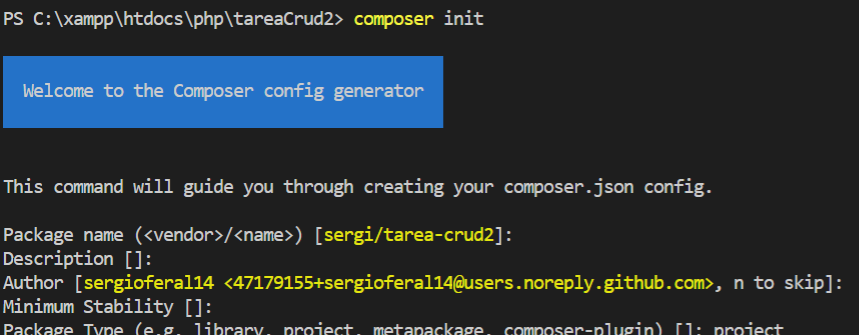


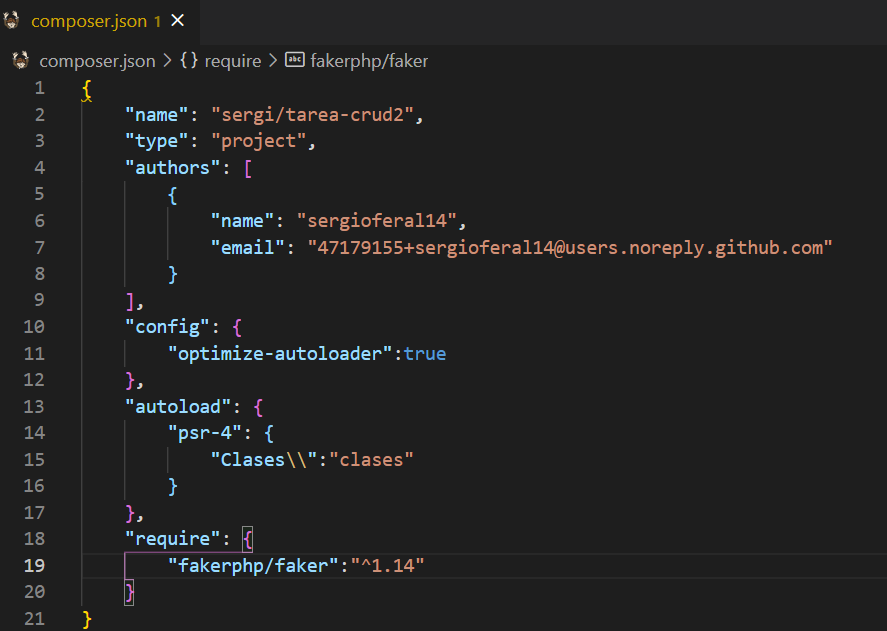
A continuación crearemos el usuario y la base de datos a la que asociaremos la tabla de datos del proyecto y le damos los permisos a ese usuario:





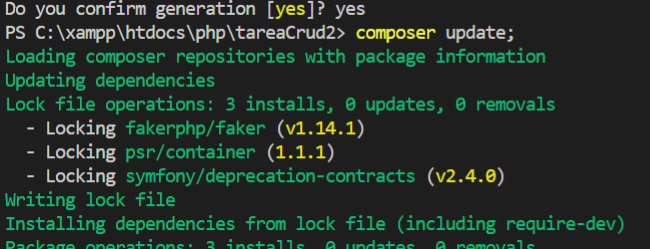
A continuación crearemos nuestro *composer.json* por medio de este comando en nuestra terminar del proyecto siguiendo las pautas y preguntas de la creación y le añadiremos el autoload y require para dar uso al faker:



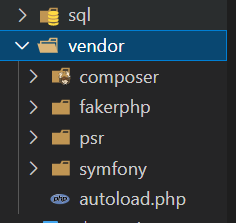


Teniendo ya creado el .json en la línea de comandos instalaremos la carpeta vendor correspondiente con la siguiente orden:

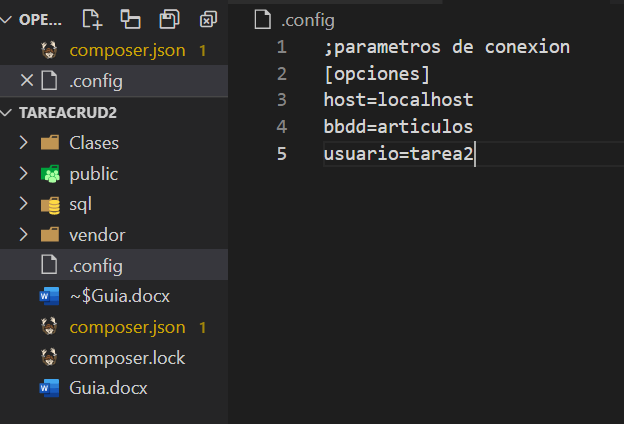
PS C:\xampp\htdocs\php\tareaCrud1> composer update



Y ya tendremos creada nuestra carpeta vendor con el paquete faker:



A continuación crearemos el archivo ***.config*** con el que indicaremos la base de datos que vamos a utilizar y el usuario:



Ahora crearemos la clase ***Conexion.php*** para poder conectarnos con la base de datos indicada del archivo ***.config:***

<?php

namespace Clases;

use PDO;

use PDOException;

//creo la clase, tiene que tener el mismo

//nombre que el archivo,en este caso Conexion

class Conexion{

    protected static $conexion;

    public function \_\_construct()

    {

        if(self::$conexion==null){

            self::crearConexion();

        }

    }

    public static function crearConexion(){

        //1.- leemos el archivo de conficuracion (.config)

        $opciones=parse\_ini\_file('../.config');

        $mibase=$opciones["bbdd"];

        $miUser=$opciones["usuario"];

        $miHost=$opciones["host"];

        //2.- creo el dsn (descriptor de servicio) con estos parametros

        $dns="mysql:host=$miHost;dbname=$mibase;charset=utf8mb4";

        //3.- Creo la conexion

        try{

            self::$conexion=new PDO($dns,$miUser);

            //lo siguiente solo para depurar errores

            self::$conexion->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

        }catch(PDOException $ex){

            die("Error al conectar a la BBDD, mansaje: ".$ex->getMessage());

        }

    }

}

Donde podemos encontrar excepciones y métodos para controlar los posibles errores de conexión con la base de datos y otros como el de borrar los creados anterirormente que lo desarrollaremos en otro archivo.

Ahora crearemos ***Articulos.php***  la cual extenderá de la clase conexión,declararemos la variables de los datos de cada articulo de nuestra base de datos.Crearemos sus respectivos setters, lafuncion \_\_construct y los métodos propios del crud:

<?php

namespace Clases;

class Articulos extends Conexion{

    private $nombre;

    private $pvp;

    private $stock;

    public function \_\_construct(){

        parent::\_\_construct();

    }

    //-------------CRUD-----------

    public function create(){

    }

    public function read(){

    }

    public function update(){

    }

    public function delete(){

    }

    //setters--------------

    public function setNombre($nombre)

    {

        $this->nombre = $nombre;

        return $this;

    }

    /\*\*

     \* Set the value of pvp

     \*

     \* @return  self

     \*/

    public function setPvp($pvp)

    {

        $this->pvp = $pvp;

        return $this;

    }

    /\*\*

     \* Set the value of stock

     \*

     \* @return  self

     \*/

    public function setStock($stock)

    {

        $this->stock = $stock;

        return $this;

    }

}

Ahora crearemos nuestro **index.php** donde mostraremos nuestros datos junto con los botones que le añadiremos sus funcionalidades mas adelante:

<!DOCTYPE html>

<html lang="es">

<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">

    <!-- CSS only -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpbzKgwra6" crossorigin="anonymous">

    <title>Artiulos</title>

</head>

<body style="background-color: lightblue;">

    <h3 class="text-center mt-3">Gestion Articulos</h3>

    <div class="container mt-3">

    </div>

</body>

</html>

Ahora nos iremos al método read() de nuestro **Articulos.php** para escribir la consulta que nos devolverá los datos de la tabla sql y un try para controlar la excepción en caso de que algo salga mal:

public function read()

    {

        $c = "select \* from articulos";

        $stmt = parent::$conexion->prepare($c);

        try {

            $stmt->execute();

        } catch (PDOException $ex) {

            die("Error al recuperar todos los datos:" . $ex->getMessage());

        }

        return $stmt;

    }

Ahora en el index.php arriba del archivo abriremos un código php para poder enlazar la clase conexión con el método read() para poder mostrar los artículos haciendo uso de namespace para las Clases y guardándolos en una variable la cual será $datos:

<?php

require "../vendor/autoload.php";

use Clases\Articulos;

$losArticulos=new Articulos();

$datos=$losArticulos->read();

$losArticulos=null;

?>

Ahora sacaremos una plantilla de tabla de bootstrap:

<div class="container mt-3">

        <table class="table">

            <thead>

                <tr>

                    <th scope="col">#</th>

                    <th scope="col">Nombre</th>

                    <th scope="col">PVP</th>

                    <th scope="col">Stock</th>

                </tr>

            </thead>

            <tbody>

                <tr>

                    <th scope="row">1</th>

                    <td>Mark</td>

                    <td>Otto</td>

                    <td>@mdo</td>

                </tr>

            </tbody>

        </table>

    </div>

Seguidamente introduciremos código php para poder meter los datos de la base de datos en esta tabla:

<?php

            while($fila=$datos->fetch(PDO::FETCH\_OBJ)){

            echo <<<TXT

                <tr>

                    <th scope="row">{$fila->nombre}</th>

                    <td>{$fila->pvp}</td>

                    <td>{$fila->stock}</td>

                    <td>#</td>

                </tr>

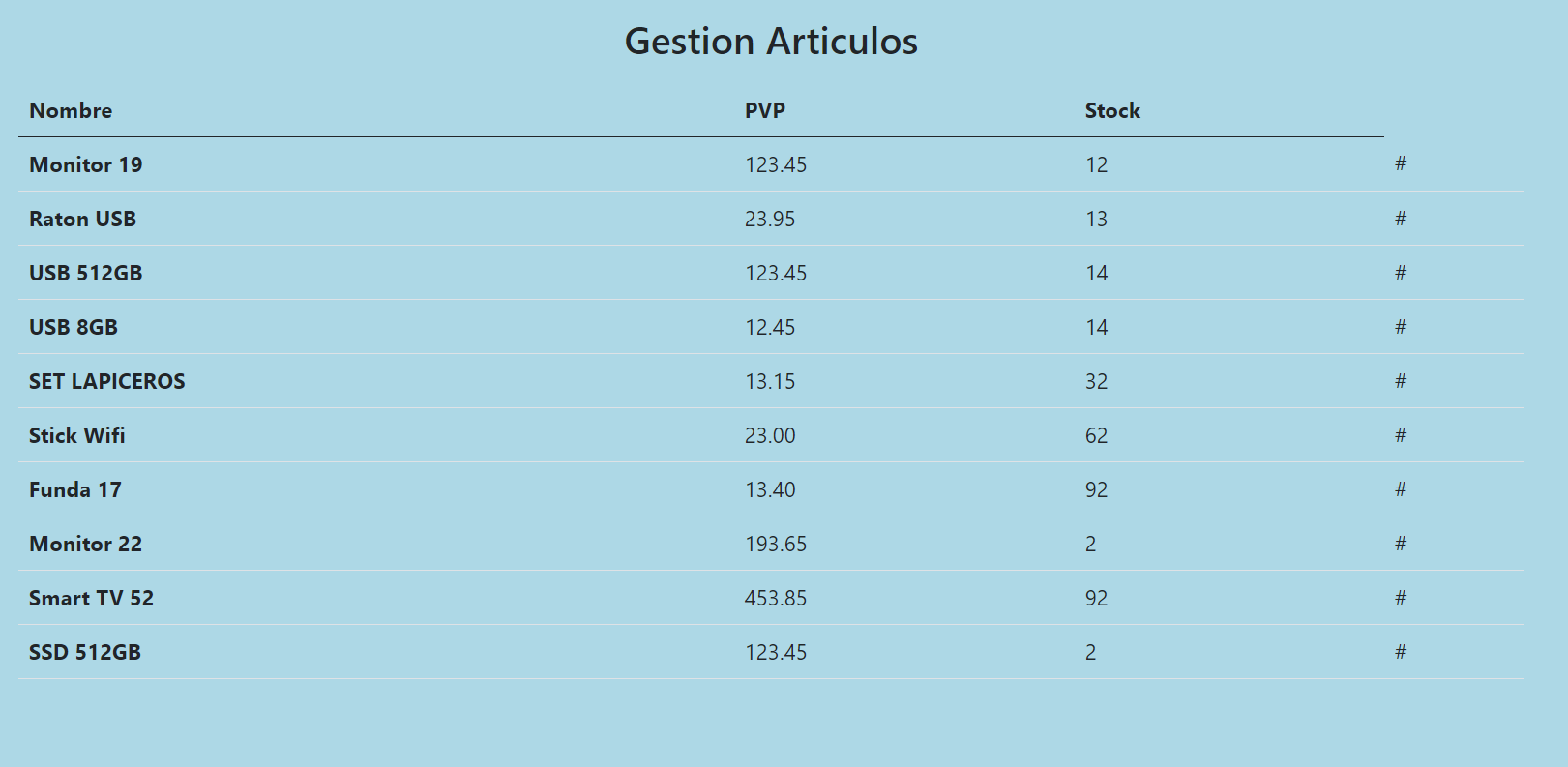
                TXT;

            }

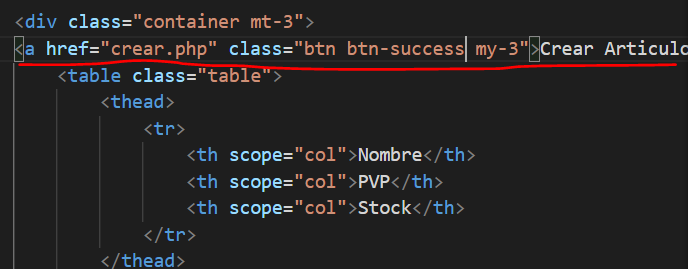
                ?>

            </tbody>

Y esto daría el siguiente resultado:



Ahora empezaremos con los botones y sus funcionalidades comenzando con el de crearañadirndo primero el botón al index:





Ahora iremos a crear dicha pagina para que no nos salte el error al pinchar en el botón partiendo de la misma platilla utilizada para el index cambiendo los titulos:

<!DOCTYPE html>

<html lang="es">

<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">

    <!-- CSS only -->

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpbzKgwra6" crossorigin="anonymous">

    <title>Crear Articulo</title>

</head>

<body style="background-color: lightblue;">

    <h3 class="text-center mt-3">Articulo Nuevo</h3>

    <div class="container mt-3">

    <form action="<?php echo $\_SERVER['PHP\_SELF'] ?>" method="POST">

    </form>

    </div>

</body>

</html>

Buscaremos una platilla de formulario de bootstrap para ingresar los datos necesarios para crear cada articulo y finalmente quedaría asi:

<!DOCTYPE html>

<html lang="es">

<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">

    <!-- CSS only -->

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpbzKgwra6" crossorigin="anonymous">

    <title>Crear Articulo</title>

</head>

<body style="background-color: lightblue;">

    <h3 class="text-center mt-3">Articulo Nuevo</h3>

    <div class="container mt-3">

        <form action="<?php echo $\_SERVER['PHP\_SELF'] ?>" method="POST">

            <div class="mb-3">

                <label for="nombre" class="form-label">Nombre</label>

                <input type="text" class="form-control" id="nombre" name="nombre" required>

            </div>

            <div class="mb-3">

                <label for="pvp" class="form-label">PVP</label>

                <input type="number" class="form-control" id="pvp" name="pvp" required>

            </div>

            <div class="mb-3">

                <label for="stock" class="form-label">Stock</label>

                <input type="number" class="form-control" id="stock" name="stock" required>

            </div>

            <button type="submit" class="btn btn-primary mr-4" name="crear">Crear</button>

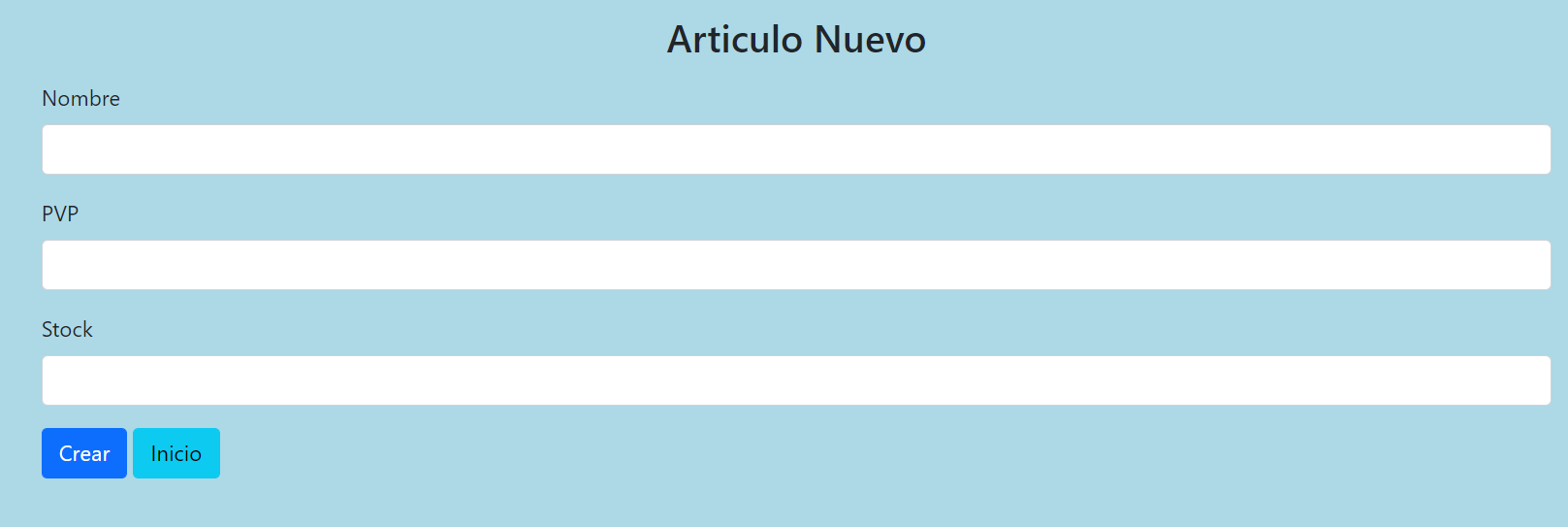
            <a href="index.php" class="btn btn-info">Inicio</a>

        </form>

    </div>

</body>

</html>



Añadiremos el control de errores en caso de darle al botón crear y que estén vacios los campos abrindo un apartado de código php al princiio y cerrando el else del controlador al final del archivo para que asi si algo sale mal sigamos estando en la misma pagina:

<html lang="es">

<?php

    session\_start();

    require "../vendor/autoload.php";

    function mostrarError($m){

        $\_SESSION['error']=$m;

        header("Location:crear.php");

    }

    if(isset($\_POST['crear'])){

       $nombre=trim($\_POST['nombre']);

       $pvp=trim($\_POST['pvp']);

       $stock=trim($\_POST['stock']);

       if(strlen($nombre)==0 || strlen($pvp)==0 || strlen($stock)==0){

            mostrarError("Rellene los campos!!!");

       }else{

       }

    }else{

?>

<head>

</body>

<?php  }?>

</html>

Despues añadimos el codigo que se encargara de relacionar los datos que introduzcamos con los setters:

$nuevoArticulo=new Articulos();

       $nuevoArticulo->setNombre($nombre);

       $nuevoArticulo->setPvp($pvp);

       $nuevoArticulo->setStock($stock);

       $nuevoUsuario->create();

       $\_SESSION['mensaje']="Usuario creado correctamente";

       $nuevoUsuario=null;

       header("Location:index.php");

Y nos iremos al método crear() en **Articulos.php** para darle la funcionalidad:

public function create()

    {

        $c="insert into articulos(nombre,pvp,stock) value(:n,:p,:s )";

        $stmt = parent::$conexion->prepare($c);

        try {

            $stmt->execute([

                ':n'=>$this->nombre,

                ':p'=>$this->pvp,

                ':s'=>$this->stock

            ]);

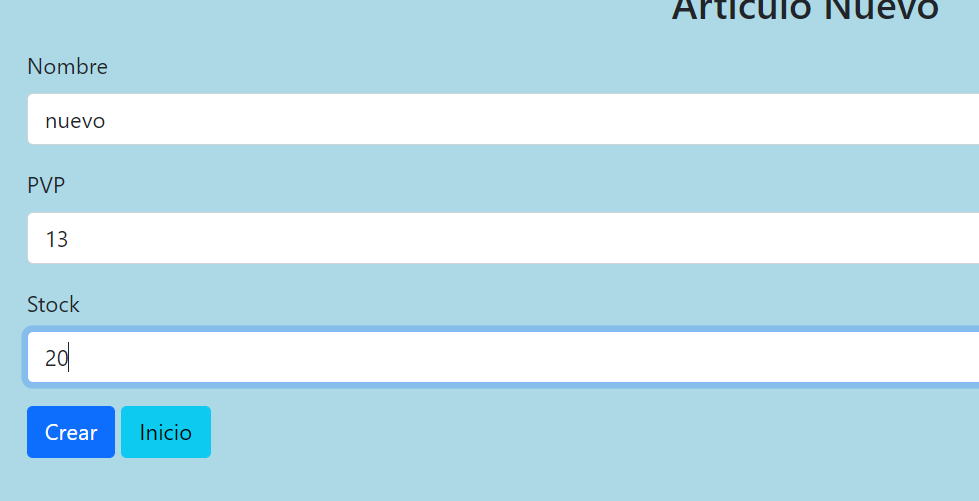
        } catch (PDOException $ex) {

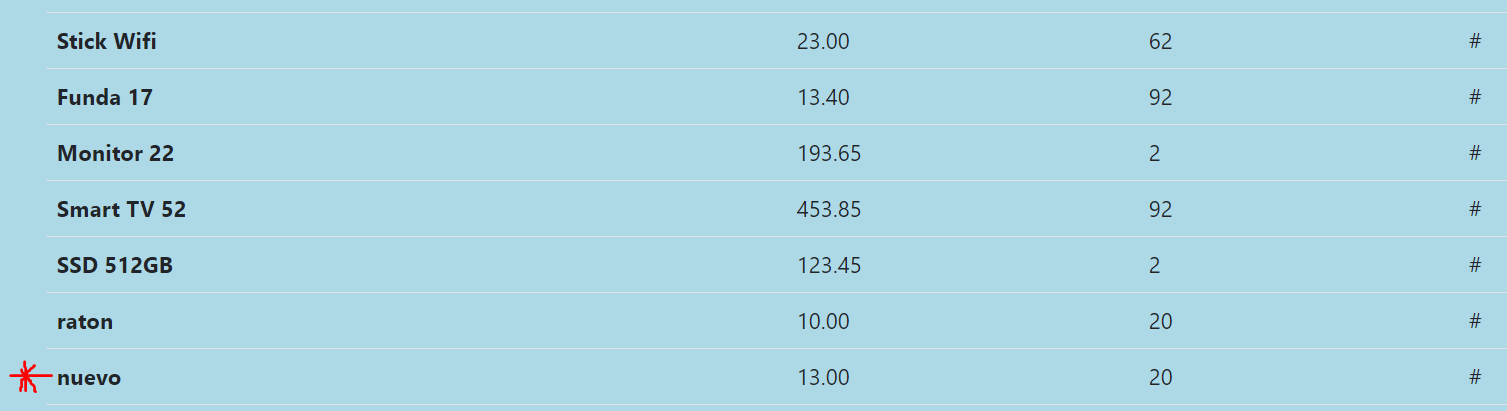
            die("Error al crear el articulo:" . $ex->getMessage());

        }

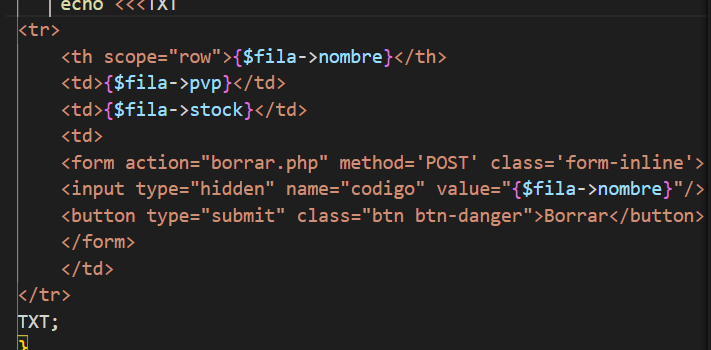
        return $stmt;

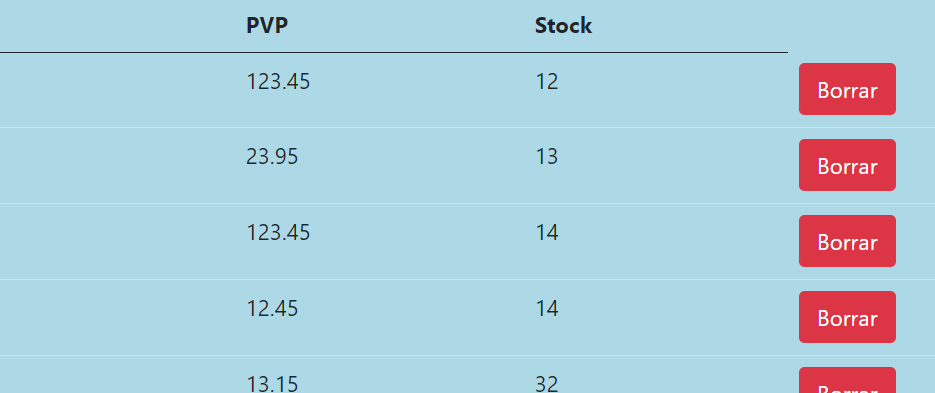
    }





Ahora crearemos el botón y la acción de borrar que es el mas sencillo empezando por crear el botón dentro del espacio que habíamos dejado en la tabla:





Ahora vamos con su funcionalidad creando el archivo borrar.php en public el cual contendrá lo siguiente:

session\_start();

if(isset($\_POST['codigo'])){

    header("Location:index.php");

}

require "../vendor/autoload.php";

use Clases\Articulos;

$nombre=$\_POST['nombre'];

$articulo=new Articulos();

$articulo->setNombre($nombre);

$articulo->delete();

$articulo=null;

$\_SESSION["mensaje"]="Articulo borrado correctamente";

header("Location:index.php");

Seguidamente volveríamos al Articulos.php a rellenar el método delete() el cual se quedaría asi:

public function delete()

    {

        $c = "delete from articulos where nombre=:n";

        $stmt = parent::$conexion->prepare($c);

        try {

            $stmt->execute([

                ':n'=>$this->id

            ]);

        } catch (PDOException $ex) {

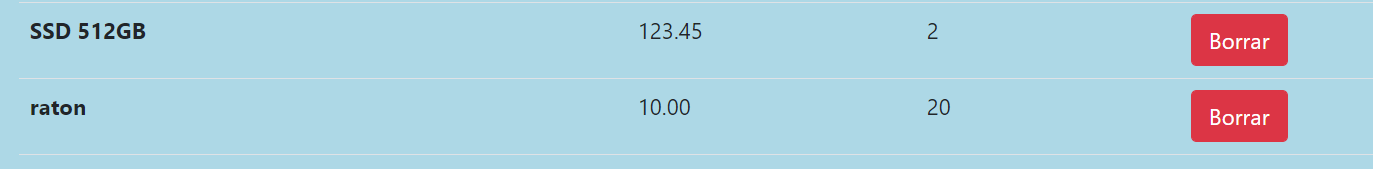
            die("Error al borrar el articulo:" . $ex->getMessage());

        }

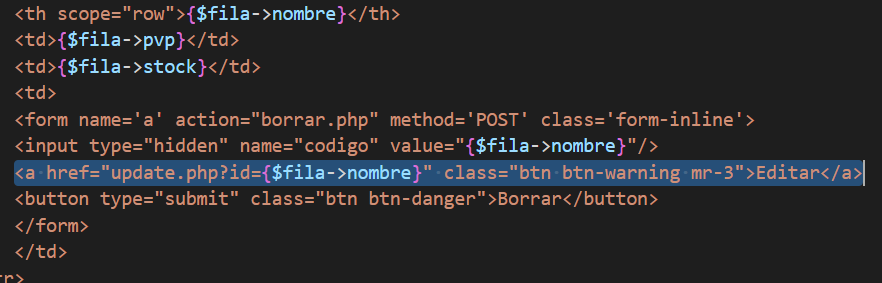
        return $stmt;

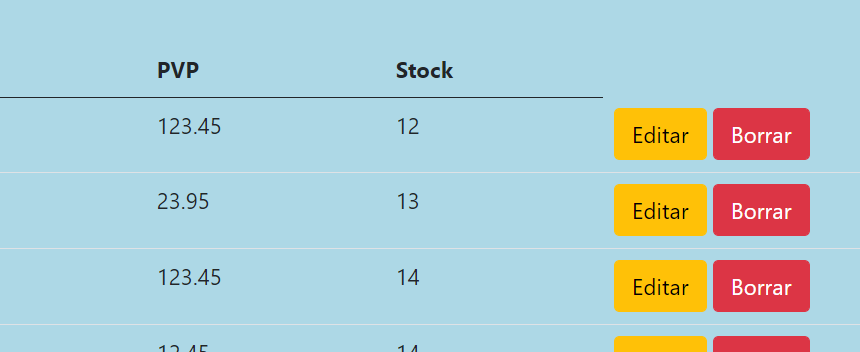
    }

Y como podemos ver ya no se encuentra el articulo “nuevo”:



Ya por ultimo procederemos a crear el **“editar”** poniendo primero el botón en el index:





Y ahora crearemos el archivo **update.php** usando lo que ya tenemos en el **crear.php**:

***No funciona correctamente***