



---

# Crud **PhP**

Proyecto realizado en php para manejo de BootStrap, PDO, SQL, namespaces y autoload optimizado de composer.

**Autor:**

Francisco Javier Barquier Martínez

**Profesor que imparte el módulo:**

Francisco Javier Fernández Arévalo

# Índice

1. Instalación composer.....	1
2. Configurar Base de Datos.....	3
2.1 Crear base de datos.....	3
2.2 Crear usuario base de datos.....	3
2.3 Dar permisos a nuestro usuario.....	3
2.4 Insertar datos.....	4
2.5 Crear archivo .config.....	4
3. Clases.....	5
4. Public.....	6

# 1. Instalación composer

Instalamos composer en nuestro proyecto con el comando `composer init`

El valor dado a los campos requeridos en mi caso son:

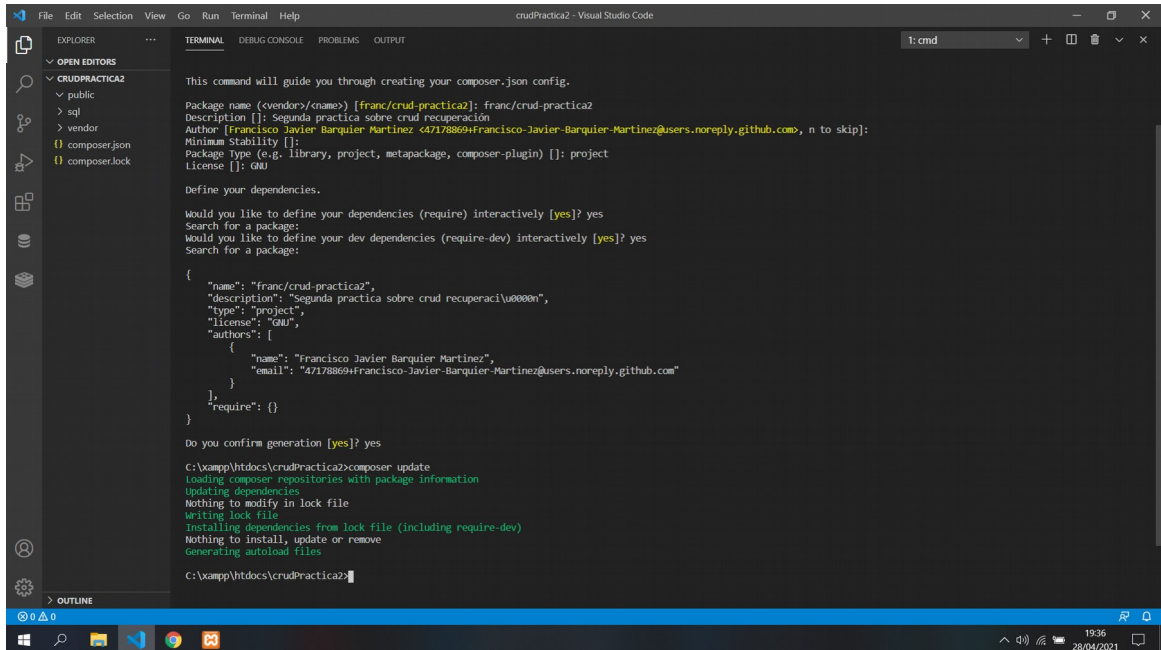
```
Package name: franc/crud-practica2
Description: Segunda práctica sobre crud recuperación
Author: Francisco Javier Barquier Martínez
Minimum Stability [] : (campo vacío)
Package Type: project
License []: GNU

Define your Dependencies
Would you like to define your dependencies (require) interactively ? : Yes
Search for a package: (campo vacío)
Would you like to define your dependencies(require-dev)interactively?: Yes
Search for a package: (campo vacío)

{
    "name": "franc/crud-practica2",
    "description": "Segunda practica sobre crud recuperacion"
    ,
    "type": "project",
    "license": "GNU",
    "authors": [
        {
            "name": "Francisco Javier Barquier Martinez",
            "email": "47178869+Francisco-Javier-Barquier-
Martinez@users.noreply.github.com"
        }
    ],
    "require": {}
}

Do you confirm generation?: Yes
```

Seguidamente ejecutamos el comando `composer update`, como resultado se nos han creado dos archivos, `composer.lock` y `composer.json`. Nuestro proyecto se vería tal que así.



```
File Edit Selection View Go Run Terminal Help
crudPractica2 - Visual Studio Code

EXPLORER
OPEN EDITORS
CRUDPRACTICA2
  > public
  > sql
  > vendor
  composer.json
  composer.lock

TERMINAL
1: cmd

This command will guide you through creating your composer.json config.

Package name (vendor/<name>) [franc/crud-practica2]: franc/crud-practica2
Description []: Segunda practica sobre crud recuperaci n
Author [Francisco Javier Barquier Martinez <47178869@francisco-javier-barquier-martinez@users.noreply.github.com>, n to skip]:
Minimum Stability []:
Package Type (e.g. library, project, metapackage, composer-plugin) []: project
License []: GNU

Define your dependencies.

Would you like to define your dependencies (require) interactively [yes]? yes
Search for a package:
Would you like to define your dev dependencies (require-dev) interactively [yes]? yes
Search for a package:

{
  "name": "franc/crud-practica2",
  "description": "Segunda practica sobre crud recuperaci n",
  "type": "project",
  "license": "GNU",
  "authors": [
    {
      "name": "Francisco Javier Barquier Martinez",
      "email": "47178869@francisco-javier-barquier-martinez@users.noreply.github.com"
    }
  ],
  "require": {}
}

Do you confirm generation [yes]? yes
C:\xampp\htdocs\crudPractica2>composer update
Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Writing lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
Generating autoload files
C:\xampp\htdocs\crudPractica2>
```

Antes de continuar, a adiremos este fragmento de c digo al archivo `composer.json`.

```
"config": {
    "optimize-autoloader": true
},
"autoload": {
    "psr-4": {
        "Clases\\": "src"
    }
},
```

## 2. Configurar Base de Datos

### 2.1. Crear base de datos

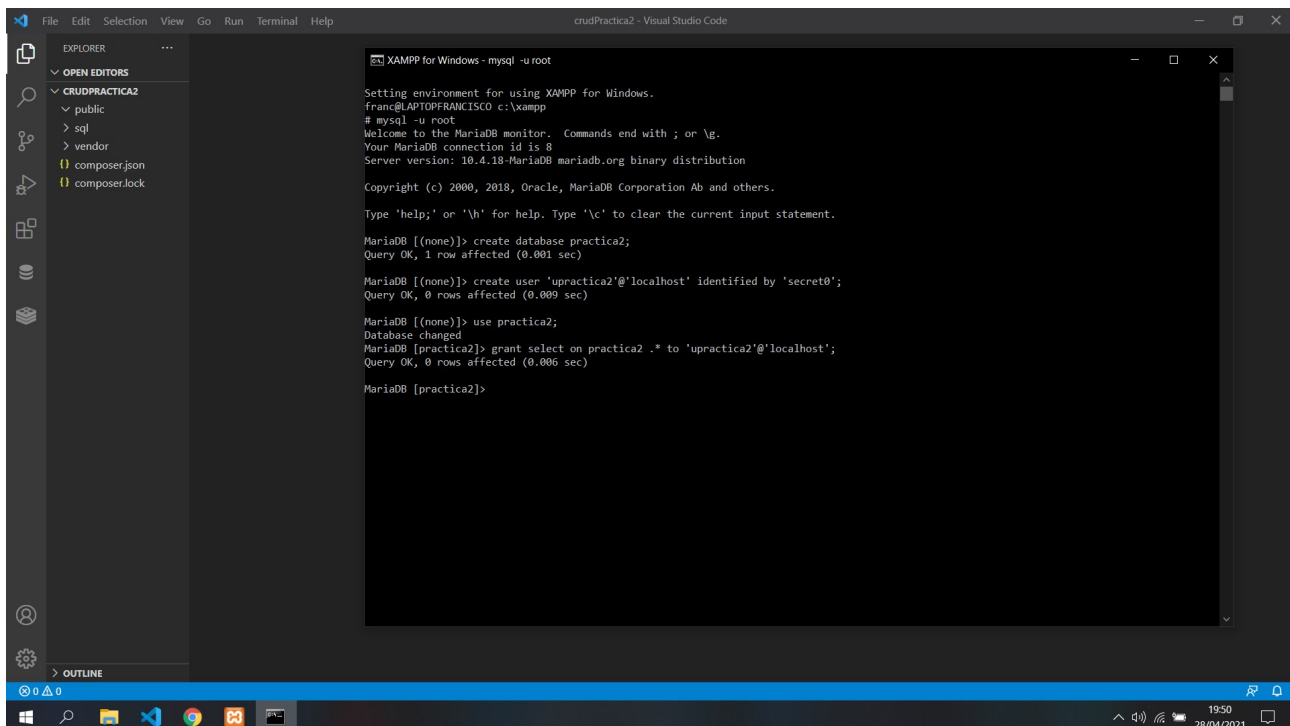
```
MariaDB > create database practica2;
```

### 2.2. Crear usuario base de datos

```
MariaDB > create database practica2;
```

### 2.3. Dar permisos a nuestro usuario

```
MariaDB > create database practica2;
```



```
XAMPP for Windows - mysql -u root

Setting environment for using XAMPP for Windows.
franc@LAPTOPFRANCISCO c:\xampp
# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.4.18-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database practica2;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> create user 'upractica2'@'localhost' identified by 'secret0';
Query OK, 0 rows affected (0.009 sec)

MariaDB [(none)]> use practica2;
Database changed
MariaDB [practica2]> grant select on practica2.* to 'upractica2'@'localhost';
Query OK, 0 rows affected (0.006 sec)

MariaDB [practica2]>
```

Imagen configuración base de datos.

## 2.4. Insertar datos.

Una vez dentro solo tendremos que copiar y pegar los siguientes datos.

```
create table articulos(  
  id int PRIMARY KEY AUTO_INCREMENT,  
  nombre VARCHAR(60) unique not null,  
  pvp decimal(6, 2) not null,  
  stock INT UNSIGNED default 0,  
);  
insert into  
  articulos(nombre, pvp, stock)  
values("Monitor 19", 123.45, 12);  
insert into  
  articulos(nombre, pvp, stock)  
values("Raton USB", 23.95, 13);  
insert into  
  articulos(nombre, pvp, stock)  
values("USB 512GB", 123.45, 14);  
insert into  
  articulos(nombre, pvp, stock)  
values("USB 8GB", 12.45, 14);  
insert into  
  articulos(nombre, pvp, stock)  
values("SET LAPICEROS", 13.15, 32);  
insert into  
  articulos(nombre, pvp, stock)  
values("Stick Wifi", 23, 62);  
insert into  
  articulos(nombre, pvp, stock)  
values("Funda 17", 13.4, 92);  
insert into  
  articulos(nombre, pvp, stock)  
values("Monitor 22", 193.65, 2);  
insert into  
  articulos(nombre, pvp, stock)  
values("Smart TV 52", 453.85, 92);  
insert into  
  articulos(nombre, pvp, stock)  
values("SSD 512GB", 123.45, 2);
```

## 2.5. Crear archivo .config

En este archivo pasaremos los parámetros para conectar a la base de datos.

```
;parametros para conexion  
[opciones]  
host=localhost  
bbdd=practica2  
usuario=upractica2  
pass=secret0
```

## 3. Clases

### 3.1. Articulos.php

Articulos.php contiene los métodos para el funcionamiento de todas los archivos del fichero public, desde leer y extraer el contenido de la base de datos hasta acciones de borrar, actualizar o crear nuevo contenido para ésta.

```
//-----CRUD-----
public function create()
{
    $i = "insert into articulos(nombre, pvp, stock) values(:n, :p, :s)";
    $stmt = parent::$conexion->prepare($i);
    try {
        $stmt->execute([
            ':n' => $this->nombre,
            ':p' => $this->pvp,
            ':s' => $this->stock
        ]);
    } catch (PDOException $ex) {
        die("Error al crear Tag: " . $ex->getMessage());
    }
}

public function read()
{
    $c = "select * from articulos where id=:i";
    $stmt = parent::$conexion->prepare($c);
    try {
        $stmt->execute([
            ':i' => $this->id
        ]);
    } catch (PDOException $ex) {
        die("Error al leer un articulo: " . $ex->getMessage());
    }
    $fila = $stmt->fetch(PDO::FETCH_OBJ);
    return $fila;
}

public function update()
{
    $u = "update articulos set nombre=:n, pvp=:p, stock=:s where id=:i";
    $stmt = parent::$conexion->prepare($u);
    try {
        $stmt->execute([
            ':i' => $this->id,
```

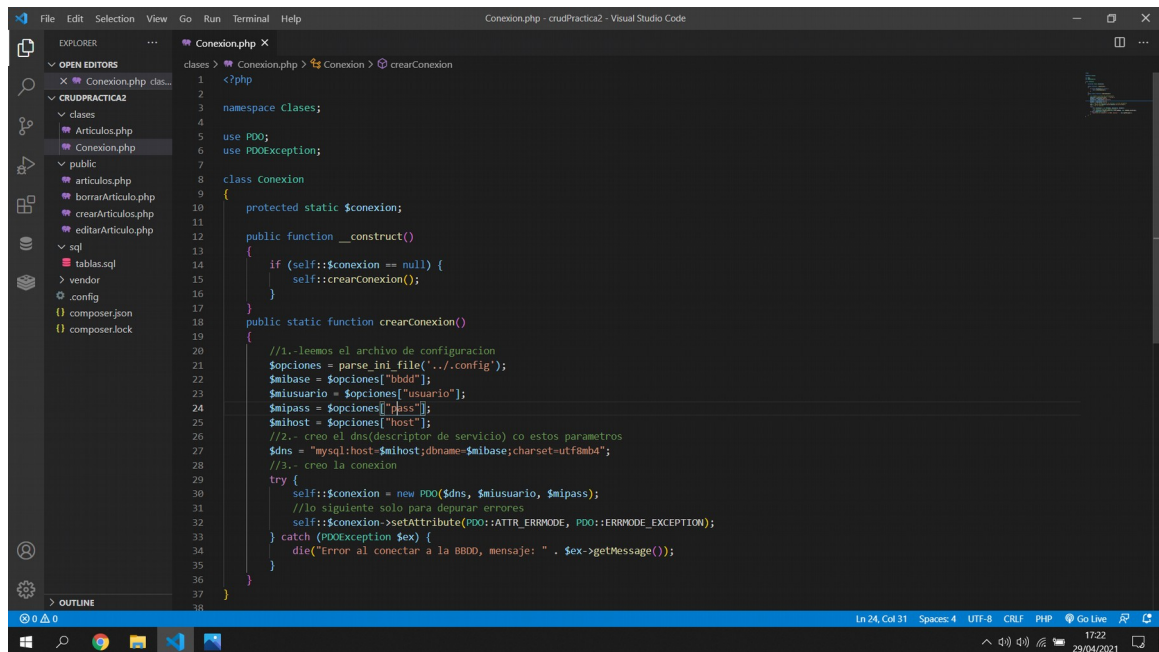
```
public function delete()
{
    $del = "delete from articulos where id=:id";
    $stmt = parent::$conexion->prepare($del);
    try {
        $stmt->execute([':id' => $this->id]);
    } catch (PDOException $ex) {
        die("Error al borrar el Articulo: " . $ex->getMessage());
    }
}

public function readAll()
{
    $i = "select * from articulos";
    $stmt = parent::$conexion->prepare($i);
    try {
        $stmt->execute();
    } catch (PDOException $ex) {
        die("Error al crear articulos: " . $ex->getMessage());
    }
    return $stmt;
}

public function comprobar($nombre)
{
    $c = "select * from articulos where nombre=:n";
    $stmt = parent::$conexion->prepare($c);
    try {
        $stmt->execute([
            ':n' => $nombre
        ]);
    } catch (PDOException $ex) {
        die("Error al comprobar existencia de articulo: " . $ex->getMessage());
    }
    $fila = $stmt->fetch(PDO::FETCH_OBJ);
    return ($fila == null) ? false : true;
}
```

## 3.2. Conexion.php

La clase conexión se encarga de establecer la conexión con la base de datos con los parámetros pasados en archivo .config anteriormente creado siendo la clase más relevante.



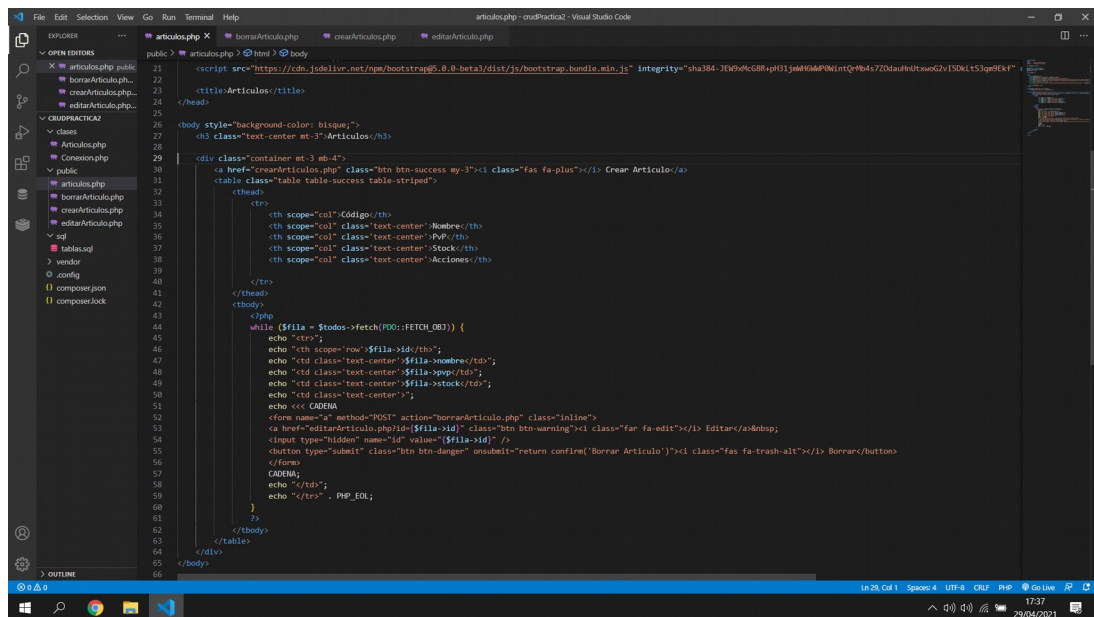
```
1 <?php
2
3 namespace Clases;
4
5 use PDO;
6 use PDOException;
7
8 class Conexion
9 {
10     protected static $conexion;
11
12     public function __construct()
13     {
14         if (self::$conexion == null) {
15             self::crearConexion();
16         }
17     }
18     public static function crearConexion()
19     {
20         //1.- leemos el archivo de configuración
21         $opciones = parse_ini_file("../.config");
22         $mihost = $opciones["host"];
23         $miusuario = $opciones["usuario"];
24         $mipass = $opciones["pass"];
25         $mihost = $opciones["host"];
26         //2.- creo el dns(descriptor de servicio) co estos parametros
27         $dns = "mysql:host=$mihost;dbname=$mihost;charset=utf8mb4";
28         //3.- creo la conexión
29         try {
30             self::$conexion = new PDO($dns, $miusuario, $mipass);
31             //lo siguiente solo para depurar errores
32             self::$conexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
33         } catch (PDOException $ex) {
34             die("Error al conectar a la BBDD, mensaje: " . $ex->getMessage());
35         }
36     }
37 }
```

## 4. Public

### 4.1. articulos.php

Este archivo es el encargado de mostrar en una tabla todos los datos almacenados en la base de datos y dar la opción mediante 3 botones de crear, borrar o editar cualquiera de los datos.





## 4.2. borrarArticulo.php

Borra el artículo seleccionado usando el método delete de la clase Articulos.php.

```
<?php

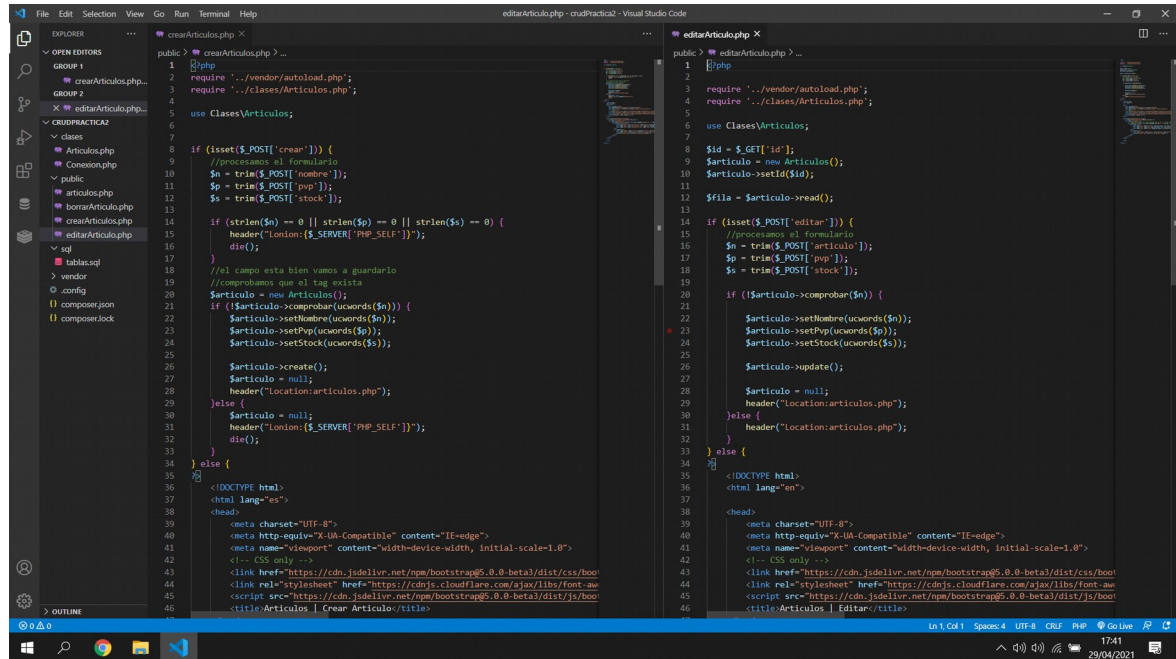
require '../vendor/autoload.php';
require '../clases/Articulos.php';

use Clases\Articulos;

$articulos = new Articulos();
$articulos->setId($_POST['id']);
$articulos->delete();
$articulos=null;
header("Location:articulos.php");
```

### 4.3. crearArticulo.php y editarArticulo.php

Estos dos archivos son muy parecidos, con la diferencia de que uno hace una llamada al método `create()` y otro al método `update()`;



```
public > crearArticulo.php > ...
1 <?php
2 require '../vendor/autoload.php';
3 require '../classes/Articulos.php';
4 use Clases\Articulos;
5
6
7
8 if (isset($_POST['crear'])) {
9     //procesamos el formulario
10    $n = trim($_POST['nombre']);
11    $p = trim($_POST['pvp']);
12    $s = trim($_POST['stock']);
13
14    if (strlen($n) == 0 || strlen($p) == 0 || strlen($s) == 0) {
15        header("Location:{$$_SERVER['PHP_SELF']}");
16        die();
17    }
18    //el campo esta bien vamos a guardarlo
19    //comprobamos que el tag exista
20    $articulo = new Articulos();
21    if ($articulo->comprobar(ucwords($n))) {
22        $articulo->setNombre(ucwords($n));
23        $articulo->setPvp(ucwords($p));
24        $articulo->setStock(ucwords($s));
25        $articulo->create();
26        $articulo = null;
27        header("Location:articulos.php");
28    } else {
29        $articulo = null;
30        header("Location:{$$_SERVER['PHP_SELF']}");
31        die();
32    }
33 } else {
34
35 }
36 <!DOCTYPE html>
37 <html lang="es">
38 <head>
39     <meta charset="UTF-8">
40     <meta http-equiv="X-UA-Compatible" content="IE=edge">
41     <meta name="viewport" content="width=device-width, initial-scale=1.0">
42     <!-- CSS only -->
43     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css">
44     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.min.js" rel="script">
45     <title>Articulos | Crear Articulo</title>
46
```

```
public > editarArticulo.php > ...
1 <?php
2 require '../vendor/autoload.php';
3 require '../classes/Articulos.php';
4 use Clases\Articulos;
5
6
7
8 $id = $_GET['id'];
9 $articulo = new Articulos();
10 $articulo->setId($id);
11 $fila = $articulo->read();
12
13
14 if (isset($_POST['editar'])) {
15     //procesamos el formulario
16    $n = trim($_POST['articulo']);
17    $p = trim($_POST['pvp']);
18    $s = trim($_POST['stock']);
19
20    if (!$articulo->comprobar($n)) {
21        $articulo->setNombre(ucwords($n));
22        $articulo->setPvp(ucwords($p));
23        $articulo->setStock(ucwords($s));
24        $articulo->update();
25        $articulo = null;
26        header("Location:articulos.php");
27    } else {
28        $articulo = null;
29        header("Location:{$$_SERVER['PHP_SELF']}");
30        die();
31    }
32 } else {
33
34 }
35 <!DOCTYPE html>
36 <html lang="en">
37 <head>
38     <meta charset="UTF-8">
39     <meta http-equiv="X-UA-Compatible" content="IE=edge">
40     <meta name="viewport" content="width=device-width, initial-scale=1.0">
41     <!-- CSS only -->
42     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css">
43     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.min.js" rel="script">
44     <title>Articulos | Editar</title>
45
```