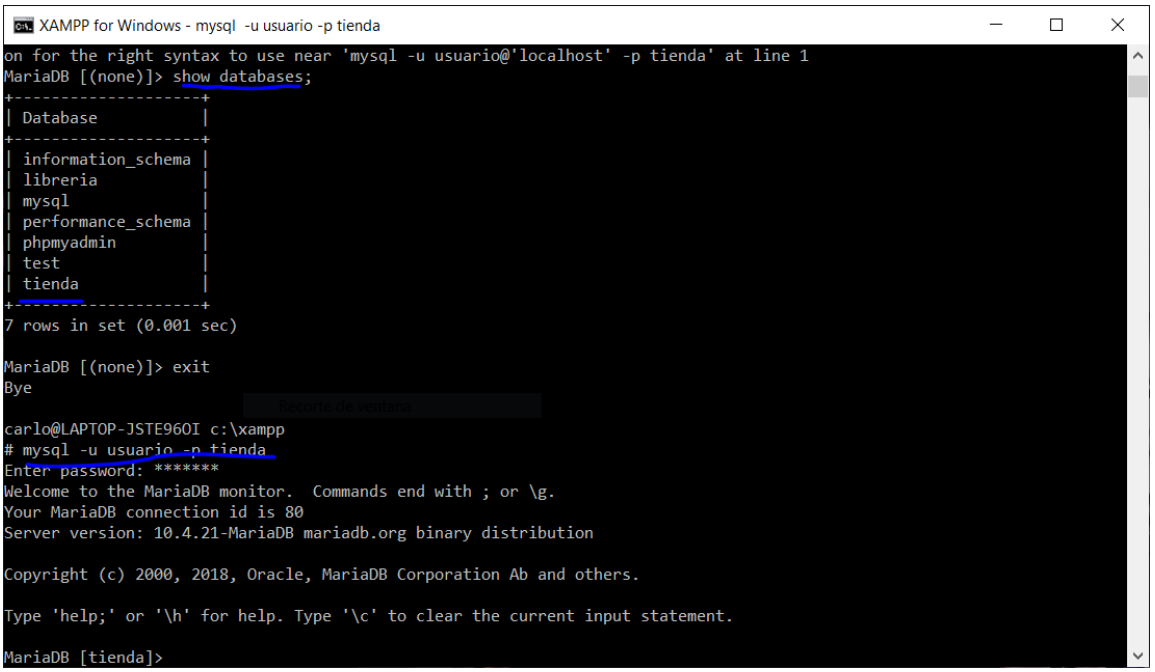
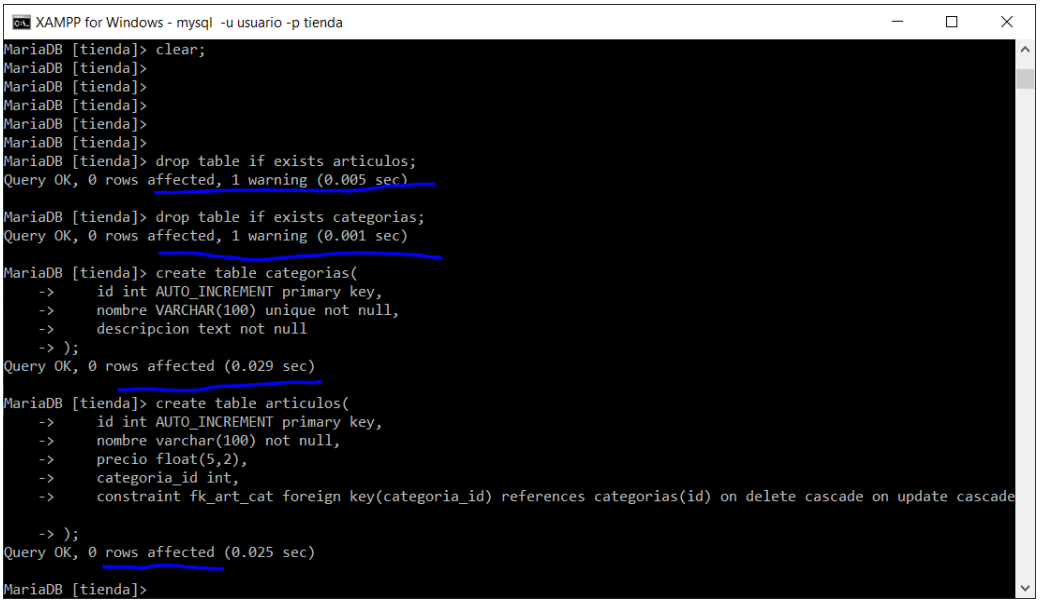
Documentación Práctica Tienda

Voy a comentar solamente los puntos más importantes y el motivo del porqué de algunas elecciones.

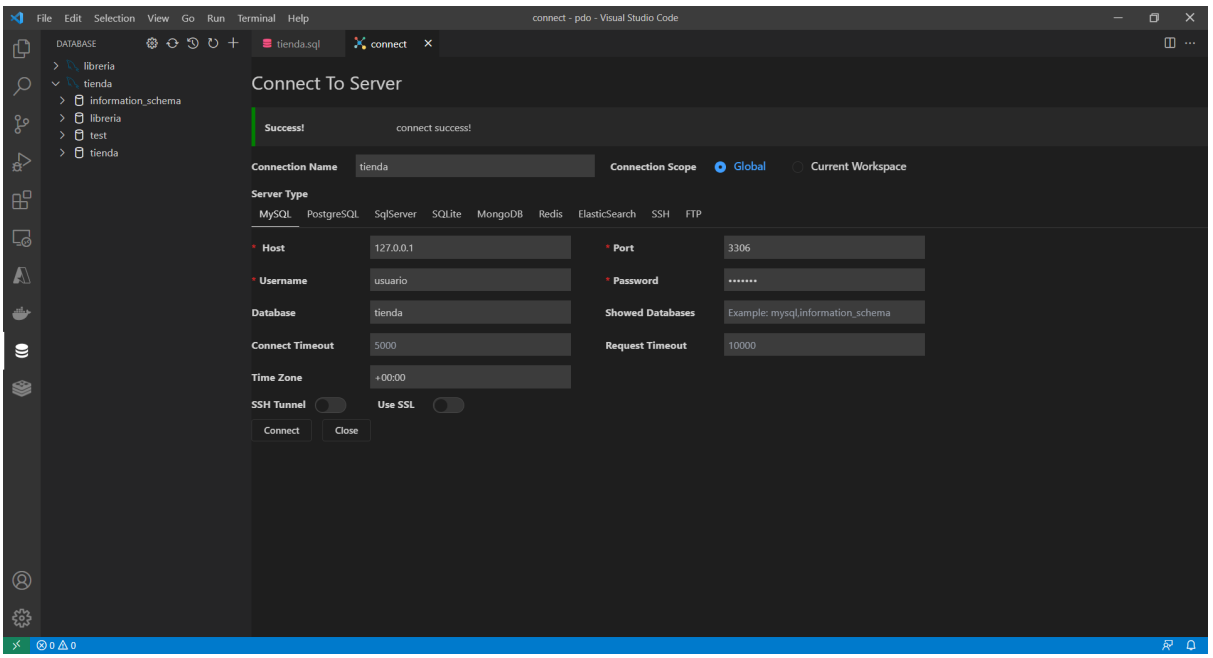
Primero vamos con la creación de la base de datos



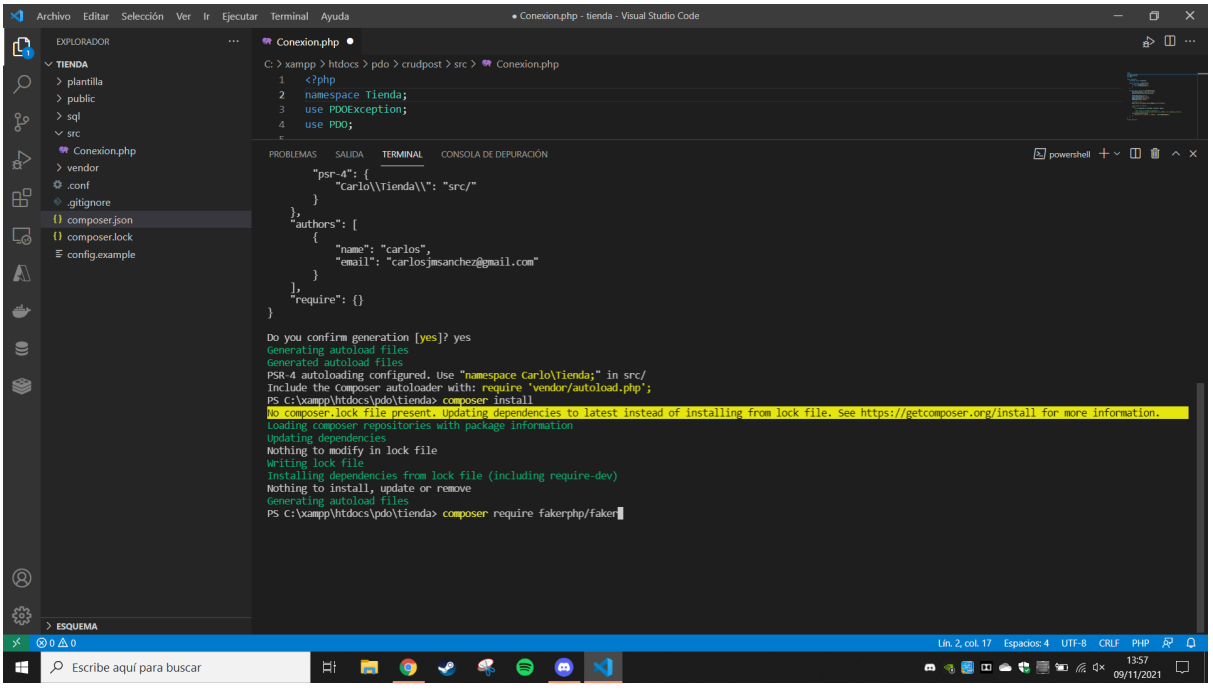
El proceso de creación de tablas



Una vez creadas las tablas nos conectamos a la base de datos en el visual studio

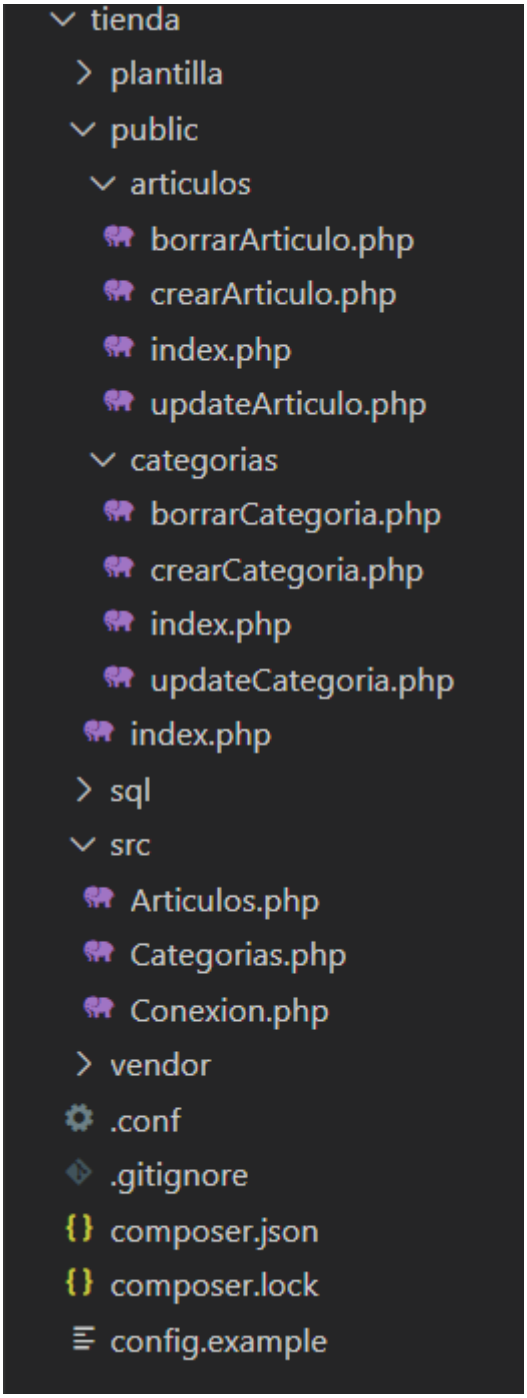


Ahora creamos los archivos necesarios tales como el gitignore, el .config e instalamos el composer y la librería faker



Una vez terminado el proyecto, procedo a comentar lo más característico.

Primero esta es la estructura de paquetes



En el index principal solo se crean aleatoriamente (en caso de que la tabla está vacía existan) datos para la tabla Categorias.

```php
<?php
    //INDEX PRINCIPAL

    //Aqui generamos con faker las categorias
    require dirname(__DIR__,1)."/vendor/autoload.php";
    use Tienda\Categorias;

    $hayCategorias=(new Categorias)->hayCategorias(); //True si hay categorias, false si no hay

    if($hayCategorias==false){
        (new Categorias)->insertarAleatorios(20);
    }
?>
```

Unas de las funciones que quiero destacar de la clase Categorías son estas:

Esta función nos devuelve solo el nombre de la categoría que hayamos pasado mediante su id

```php
public function devolverNombreCategoria($id){
    $q="select * from categorias where id=:id";
    $stmt=parent::$conexion->prepare($q);

    try{
        $stmt->execute([
            ':id'=>$id
        ]);
    }catch(PDOException $ex){
        die("Error al devolver el nombre de la categoria: ".$ex->getMessage());
    }

    $nombre=$stmt->fetch(PDO::FETCH_OBJ)->nombre; //Como la consulta devuelve una fila, obtenemos el nombre de la misma y lo retornamos
    parent::$conexion=null;
    return $nombre;
}
```

La segunda que quiero comentar es esta:

Función que cambia la consulta en base a los datos que se les haya introducido.

```php
public function existeCategoria($nombre){
    if(isset($this->id)){
        //Si entra aqui quiere decir que hay un id en el objeto, por lo tanto estamos ante una secuencia de update
        //se modifica la consulta para que busque todos los ids de categorias menos el que tiene la categoria que queremos actualizar
        $q="select * from categorias where nombre=:n AND id!={$this->id}";
    }else{
        $q="select * from categorias where nombre=:n";
    }
    $stmt=parent::$conexion->prepare($q);

    try{
        $stmt->execute([
            ':n'=>$nombre
        ]);
    }catch(PDOException $ex){
        die("Error al comprobar si existe la categoria: ".$ex->getMessage());
    }
    parent::$conexion=null;

    return ($stmt->rowCount()==0); //Falso si existe esa categoria
}
```

Ahora lo más característico del index.php de Artículos es esto

```php
<?php
    //INDEX DE ARTICULOS
    session_start(); //Se usarán al momento de mostrar mensajes

    require dirname(__DIR__,2)."/vendor/autoload.php";
    use Tienda\{Articulos, Categorias}; //Usaremos Categorias para traer el nombre del id de la categoria

    /* AQUI GENERAREMOS LOS ARTICULOS */
    // SE HACE DE ESTA FORMA PORQUE ANTES DE HABER ARTICULOS DEBE DE HABER CATEGORIAS
    if((new Articulos)->hayArticulos()==false){
        //SI entra quiere decir que no hay articulos, los generamos
        (new Articulos)->generarAleatorios(30);
    }

    $listadoArticulos=(new Articulos)->readAll(); //Trae todos los articulos
?>
```

En este index es donde se generan aleatoriamente los datos de artículos. Esto es debido a que no puede haber artículos si no hay categorías. Se podría haber puesto también en el index principal, pero para dejarlo más organizado he decidido ponerlo aquí.

A la hora de crear un artículo, tendremos un selector donde podremos elegir cualquiera de las categorías ya existentes. Para ello primero obtenemos todas las categorías, las ordenamos como si fueran arrays y después sacamos el valor del nombre de cada categoría e internamente se hará uso del id de la categoría.

```php
<div class="mb-3">
    <label for="a" class="form-label">Indica el nombre de la categoria</label>
    <select class="form-select" name="categoria_id" id="a">
        <?php
            $categorias=(new Categorias)->readAll(); //Nos traemos todas las categorias

            //Como readAll devuelve el saco $stmt, lo tratamos como un array con fetchAll y lo vamos recorriendo
            foreach($categorias->fetchAll(PDO::FETCH_OBJ) as $item){
                echo "<option value='{$item->id}'>{$item->nombre}</option>";
            }
        ?>
    </select>
```

Estas son las distintas comprobaciones de los artículos

```php
function hayError($name, $precio){
    global $error;
    //Comprobamos si hay cadenas vacias
    if(strlen($name)==0){
        $error=true;
        $_SESSION['error_nombre']="El nombre no puede estar vacio";
    }

    if(strlen($precio)==0){
        $error=true;
        $_SESSION['error_precio']="El precio no puede estar vacia";
    }

    //Comprobamos que el precio está entre los valores admitidos

    if($precio<0){
        $error=true;
        $_SESSION['error_precio']="El precio no puede ser menor que 0";
    }

    if($precio>999.99){
        $error=true;
        $_SESSION['error_precio']="El precio no puede ser mayor que 999.99";
    }
    //--------
}
```

Se comprueba que las cajas de texto no estén vacías y que el precio está en los valores correctos.

A la hora de editar un artículo, por defecto se selecciona la categoría que tiene mediante la siguiente forma

```php
<div class="mb-3">
    <label for="a" class="form-label">Indica el nombre de la categoria</label>
    <select class="form-select" name="categoria_id" id="a">
        <?php
            $categorias=(new Categorias)->readAll(); //Nos traemos todas las categorias

            //Como readAll devuelve el saco $stmt, lo tratamos como un array con fetchAll y lo vamos recorriendo
            foreach($categorias->fetchAll(PDO::FETCH_OBJ) as $item){
                if($item->id == $id){
                    echo "<option selected value='{$item->id}'>{$item->nombre}</option>";
                }
                echo "<option value='{$item->id}'>{$item->nombre}</option>";
            }
        ?>
    </select>
</div>
```

Por último mostraré las comprobaciones de las Categorías.

```php
function hayError($name, $description){
    global $id;
    global $error;
    //Comprobamos si hay cadenas vacias
    if(strlen($name)==0){
        $error=true;
        $_SESSION['error_nombre']="El nombre no puede estar vacio";
    }

    if(strlen($description)==0){
        $error=true;
        $_SESSION['error_descripcion']="La descripcion no puede estar vacia";
    }

    //Comprobamos que el nombre no exista ya en la base de datos comprobando tambien el id
    if(!(new Categorias)->setId($id)->existeCategoria($name)){
       $error=true;
       $_SESSION['error_nombre']="La categoria ya existe";
    }
}
```

Se comprueban que las cadenas no estén vacías y que en este caso la categoría no exista ya en la base de datos