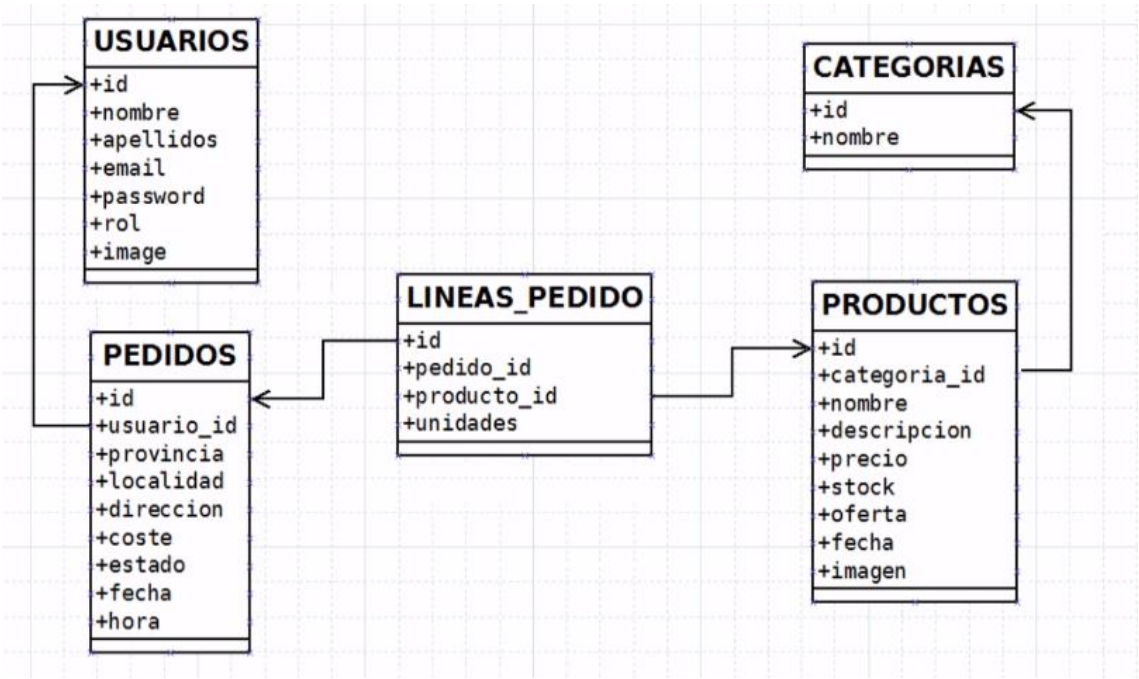
2º Proyecto : Carrito de la compra

Vamos a realizar una aplicación web para la gestión de la compra de productos con un carrito de la compra utilizando MVC y orientación a objetos

Los pasos para seguir son los siguientes

1. Diseño de la Base de datos usuarios
2. Maquetación
3. Estructura de MVC
4. Registro de usuarios y login de usuarios
5. Gestionar categorías de la tienda usuarios administradores (lista y crear)
6. Gestionar Productos de la tienda para usuarios administradores (listar, crear, subir imágenes, editar y borrar)
7. Mostrar productos página de inicio
8. Hacer carrito de la compra con sesiones
9. Mas funcionalidades del carrito (Subir nota)
10. Formulario para hacer el pedido (Subir nota)
11. Gestionar pedidos para administradores (Subir nota)

1. Diseño de la Base de datos usuarios



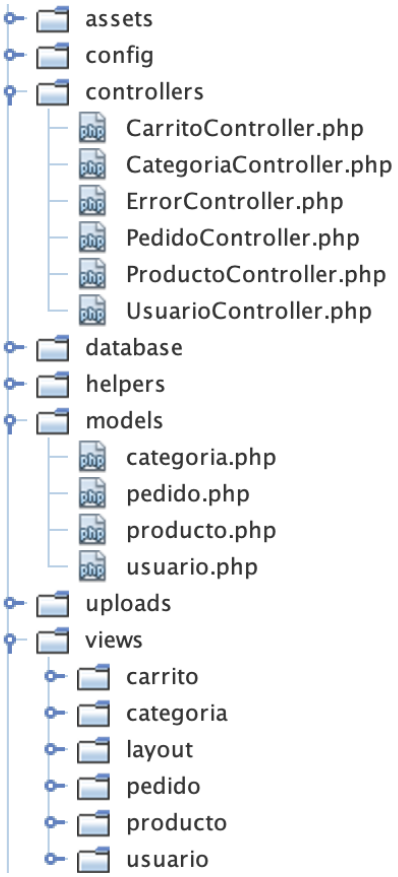
2. Maquetación

Realiza una maquetación adecuada utilizando css y separa el código de el header, nav, footer en **views/layout.** Las imágenes y css de tu tienda se guardarán en **assets** y las imágenes subidas por el usuario de los productos en **uploads.**

Partiremos de 2 usuarios creados en la base de datos unos con rol **admin** y otro con rol **user.**

3. Estructura de MVC

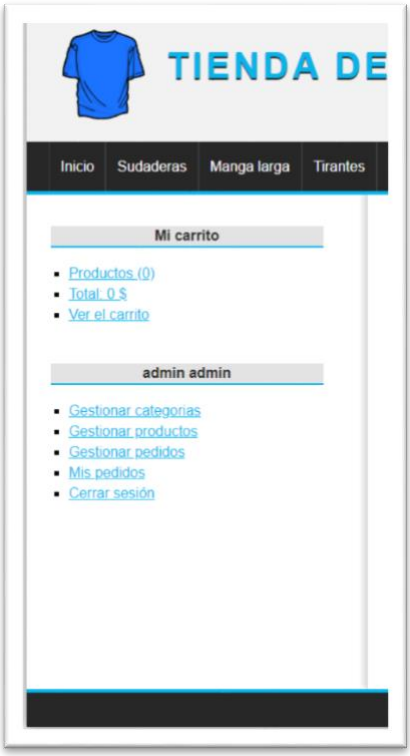
Partiremos de la estructura de la pizzeria para empezar a realizar nuestro proyecto



4. Registro de usuarios y login de usuarios

El enlace de Registrate aquí (solamente registra usuarios tipo user) lo dejaremos para el final del proyecto (Subir nota)**. Los usuarios administradores se introducen manualmente en la Base de datos**

Los usuarios tendrán 2 roles diferentes:

admin	user
	

- Tendrás que crear 2 variables de sesión $_SESSION[‘identidad’] para guardar el objeto con todos los datos del usuario y otra $_SESSION[‘admin’] donde asignaremos el valor **true** si es administrador.

Modelo: usuario

```
public function login(){
    $usuario=false;
    $sql = "SELECT * FROM usuarios WHERE email='{$this->getEmail()}' AND password='{$this->getPasswor
    $save = $this->conn->query($sql);
    if($save && ($save->rowcount()== 1)){
        $usuario = $save->fetchObject();

    }

    return $usuario;
}
```

Controlador: usuario@controler

```php
public function login(){
    if (isset($_POST['submit'])){
        $email = isset($_POST['email-login']) ? $_POST['email-login'] : false;
        $password = isset($_POST['password-login']) ? $_POST['password-login'] : false;
        if($email && $password){
            $usuario = new UsuarioModel();
            $usuario->setEmail($email);
            $usuario->setPassword($password);
            $miusuario = $usuario->login();

            if($miusuario){
                $_SESSION['identidad'] = $miusuario;
                if($miusuario->rol == 'admin'){
                    $_SESSION['admin'] = true;
                }
            }else{
                $_SESSION['error_login'] = "El usuario o contraseá es incorrecto";
            }
        }

    }
        header('Location:index.php');

}
```

- Realiza en todos los formularios un control de errores por HTML (email, numero, etc) y un control básico de errores en php donde indique solamente que tiene que volver a rellenar el formulario sin indicar el campo.

5. Categorías de la tienda

Es recomendable realizar en **helpers/utils** un método dentro de la clase utils que nos identifique como administrador o usuario y si no que nos reenvíe pagina principal. El método identificar podrá tener el valor administrador o usuario.

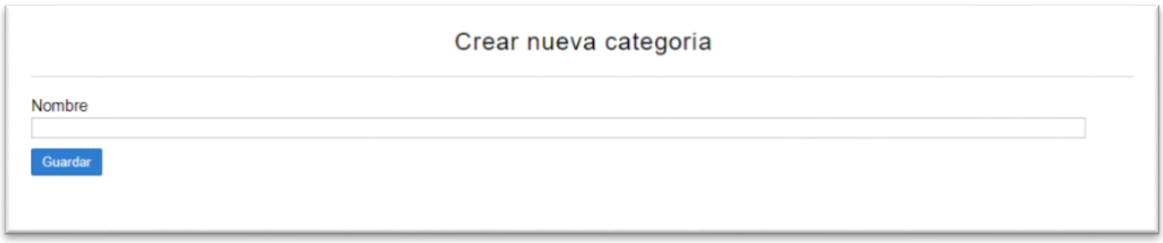
```
class Utils{

    public static function identificar($user){
            if(!isset($_SESSION[$user])){
                    header("Location:"index.php”);
            }else{
                    return true;
            }
    }}
```

Controllers/CategoriaController

Importa las librerías al principio de la clase
require_once 'models/categoria.php';

- **Index():** Obtener todas las categorías y mostrarlas con la vista
- **Crear():** Mostrar la vista del formulario
- **Save():** Comprueba que existe $_POST[‘nombre’] y crea un objeto categoría con el nombre y utiliza el método del modelo correpondiente para guárdalo de la BD.

Todos los métodos llevarán al principio la línea **Utils::identificar(“administrador”)**



Muestra en el menú todas las categorías que tengamos en el sistema.

6. Productos de la tienda

Controllers/ProductoController

Todos los métodos llevarán al principio la línea **Utils::identificar("administrador")**

- **Gestion():** Muestra todos los productos a través de una vista con sus botones de editar y eliminar.
- **Crear():** Mostrará el formulario para crear el producto. Esa misma vista del formulario también te puede servir para modificar el producto seleccionado la **action y value** de forma correcta. En modificar muestra la imagen del producto.
- **Save():** En este método recogemos los datos del formulario de la forma correcta con sus trim y realizaremos un control de errores simple (un if que incluya todos los campo sin distinguir cual es el erroneo) por si el campo está vacio y se almacenará en una variable de sesión si es correcto o si ha fallado.
  También tendrás que guardar la imagen en la carpeta **uploads/img y** comprobar si te han introducido un id por la url entonces tendrás que modificar y si no crearlo.
- **Editar():** Comprobar si existe la variable $_GET['id'], crear el objeto del producto con esa id y sacar los datos para introducirlo en la vista de crear. En la vista se comprobará si el objeto esta creado para cambiar el action y los value
- **Eliminar():** Elimina un producto por medio una id introducido por la url

7. Mostrar productos

Productos en la página de inicio

Importante: Recuerda que si no existe la imagen del producto muestre alguna imagen por defecto que tengas **/assets/img**

Controllers/ProductoController

- **Index():** Elegir al azar la cantidad de productos que le indiques para la pantalla principal y muéstralos con su vista correspondiente. **Este será el método por defecto de nuestro portal.**

  Dentro de **Models/producto** incluye este método para el index()

```php
public function getRandom($limit){
        $productos = $this->conexion->query("SELECT * FROM productos ORDER BY RAND() LIMIT $limit");
        return $productos;
}
```

Listado de productos en una categoría del menú

Controllers/CategoriaController

Añade junto a la librería de categoría la librería de producto **require_once 'models/producto.php';**

- **Ver():** A través de la id obtenemos los datos de la categoría por un lado y por medio de un método de producto obtenemos todos losproductos de una categoría (Tendrás que utilizar un select con INNER JOIN de las tablas producto y categorías para obtener todos los datos)

Detalle del producto (Subir nota)

Controllers/ProductoController

- **Ver():** Muestra los datos de un producto a través del id pasado por la URL. Recuerda que en la vista tienes que controlar que si el producto no tiene imagen pon una por defecto

8. Carrito de la compra (Sesiones)

Para llevar todos los productos del carrito tendremos una variable de $_SESSION[‘carrito’] que será un array bidimensional donde en cada posición guardemos un array asociativo con los datos del producto (id_producto, precio,unidades=1, producto)

CarritoController

- **Index():** Comprueba si existe la variable $_SESSION[‘carrito’] y si es igual o mayor 1 entonces guardaremos $carrito el valor en una variable $carrito y si no la variable $carrito=array(); después llamaremos a la vista que imprima la variable $carrito.
- **Add():** Cuando pulsemos el **botón de comprar** tendremos que guardar los datos del producto (buscaremos los datos a través de un método de Producto)a través de su id en la variable de sesión en la última posición

  $_SESSION['carrito'][]= array("id_producto" => $producto->id,…).

  Después iremos a nuestro método index para cargar la vista

  Cuando el usuario pulse el botón de comprar de nuevo en un producto tendrá que subir las unidades y no insertarlo de nuevo en la sesión.Colocaremos este código antes de guardarlo en la sesión

```
if(isset($_SESSION['carrito'])){
                $counter = true;
                foreach($_SESSION['carrito'] as $indice => $elemento){
                        if($elemento['id_producto'] == $producto_id){
                                $_SESSION['carrito'][$indice]['unidades']++;
                                $counter=false;}}}
```

  Para introducir un articulo nuevo tendremos que comprobar que $counter sea igual a true o que no exista.

Estadísticas de Mi carrito

Utiliza una función externa en **helpers/utils.php**. La función devolverá un array asociativo (numProducto y totalPrecio)

```
Class Utils {……
Public static function DatosCarrito(){}
……................}
```

Para llamar a la función **<?php $datos = Utils::DatosCarrito(); ?>**

Mostrar el carrito

Dentro de la vista /**views/carrito/index.php realizar un recorrido por el array y saca el objeto completo**

```
foreach($carrito as $indice => $elemento){
        $producto = $elemento['producto'];
……….
}
```

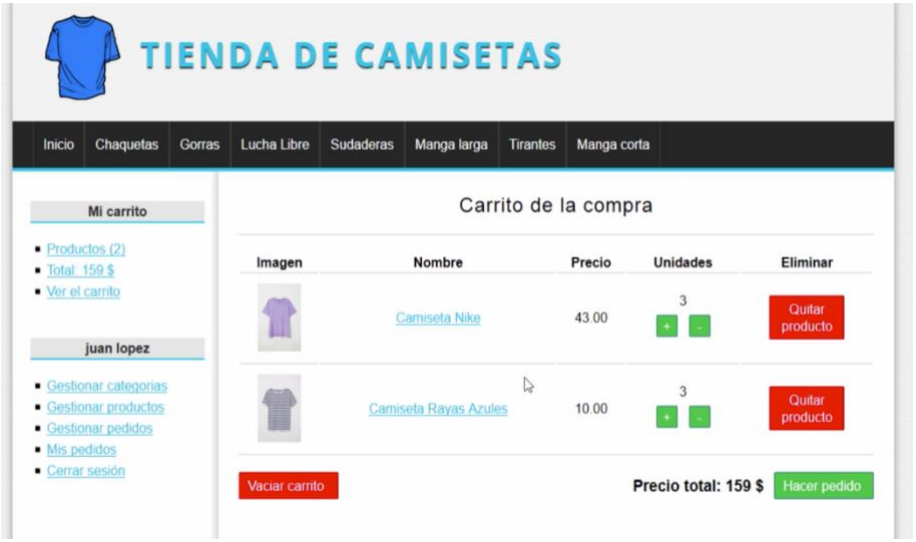
Si el producto no tiene una imagen (null) muestra una por defecto que tengas en la carpeta **/assets/img**

```
<?php if ($producto->imagen != null){ ?>
                <img src="uploads/images/<?= $producto->imagen ?>"
class="img_carrito" />
<?php }else{ ?>
        <img src="assets/img/camiseta.png" class="img_carrito" />
                        <?php }?>
```

9. Más funcionalidades de carrito de la compra (Subir nota)

En este apartado vamos añadir a nuestro carrito

- Vaciar carrito
- Quitar producto
- Subir/bajar unidades



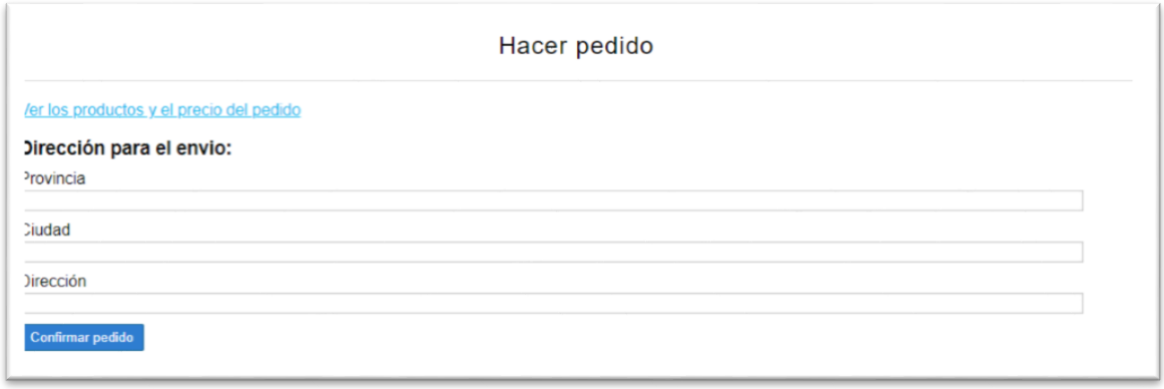
Es necesario implementar un método en el **Controllers/CarritoController** para cada una de las funciones

- **Delete_all():** Libere la variable de sesión del carrito (unset) y te redirija al index (listado del carrito)
- **Delete_prod():** Recogeremos el índice del array del producto que queremos borrar y liberamos dentro de **unset ($_SESSION[‘carrito’][$index]);**
- **Up():** Recogemos el índice del array y aumentamos en una unidad de dicho producto
- **Down():**Recogemos el índice del array y disminuimos en una unidad de dicho producto

10. Pedidos de la tienda (Subir nota)

Controllers/PedidoController

- **Hacer():** En este método mostraremos el formulario para realizarla dirección de envío cuando pulse el botón del carrito hacer pedido. El enlace (ver los productos y el precio del pedido) que aparce arriba en el formulario volverá a mostrar los productos del carrito.



Cuando confirmes el pedido introduciremos un método

- **ADD():** Asegúrate que el usuario esté identificado e introduce los datos del formulario en la tabla pedidos, el id_usuario puedes obtenerlo de la cariable de sesión y el coste de la función de **Utils::DatosCarrito();** Realiza un control de errores simple. Además de insertar en la tabla pedido tendrás que usar otro método para insertar una línea por cada producto en la tabla **líneas_pedidos**
- **Confirmado():** Muestra la información que muestra la fotografía

En Gestionar pedidos (administrador) y Mis pedidos (Usuario) en principio mostraremos lo mismo aunque el administrador tendrá un enlace extra en gestionar para cambiar el estado del pedido

Mis pedidos

Nº Pedido	Coste	Fecha	Estado
1	360.00 $	2019-12-03	Pendiente