

**FOODY**



# ÍNDICE

- 1. Estructura y funcionamiento general de la web pag. 4-5
- 2. Contenido e imagenes de la web pag. 6-11
- 3. Funciones utilizadas pag. 12-23
- 4. Estructura de la base de datos pag. 24

Sergio Fernandez Nevado

Julian Duncan Gonzalez E.

Pedro del Olmo Jimenez

# ESTRUCTURA GENERAL

En todas las páginas tenemos una estructura general que comienza con lo siguiente:

- Se comprueba si hay una sesión activa, que se crea cuando se comprueba que las credenciales introducidas son correctas o cuando un usuario se registra correctamente en la aplicación. En caso de existir esta sesión en las páginas de index.php (el login) o register.php se redirijirá directamente a home.php, en caso de no existir esta sesión e intentar acceder a cualquiera de las demás páginas de la aplicación se le redirijirá al login.
- También tenemos algo común en estas páginas en las que el usuario debe haberse identificado previamente para acceder, se comprueba si su tiempo de inactividad es superior a los 5 minutos. Esta comprobación se hace mediante una cookie en la que guardamos la hora de la última actividad y la comparamos con la hora actual y en caso de haber sobrepasado los 5 minutos se cerrará la sesión.
- En caso de recibir un método POST, comprobará en todas las páginas que el token enviado mediante este método coincide con el token respectivo al usuario que ha realizado dicha petición, en caso contrario se cerrará la sesión y redirijirá al usuario al login.
- Las páginas también constan de un tema claro y un tema oscuro que se puede cambiar y quedará almacenado en una cookie, esto es común en cualquiera de los usuarios.
- En caso de producirse algún error al conectarse a la base de datos en alguna de las operaciones a realizar, se guardará esta información en error\_log.log

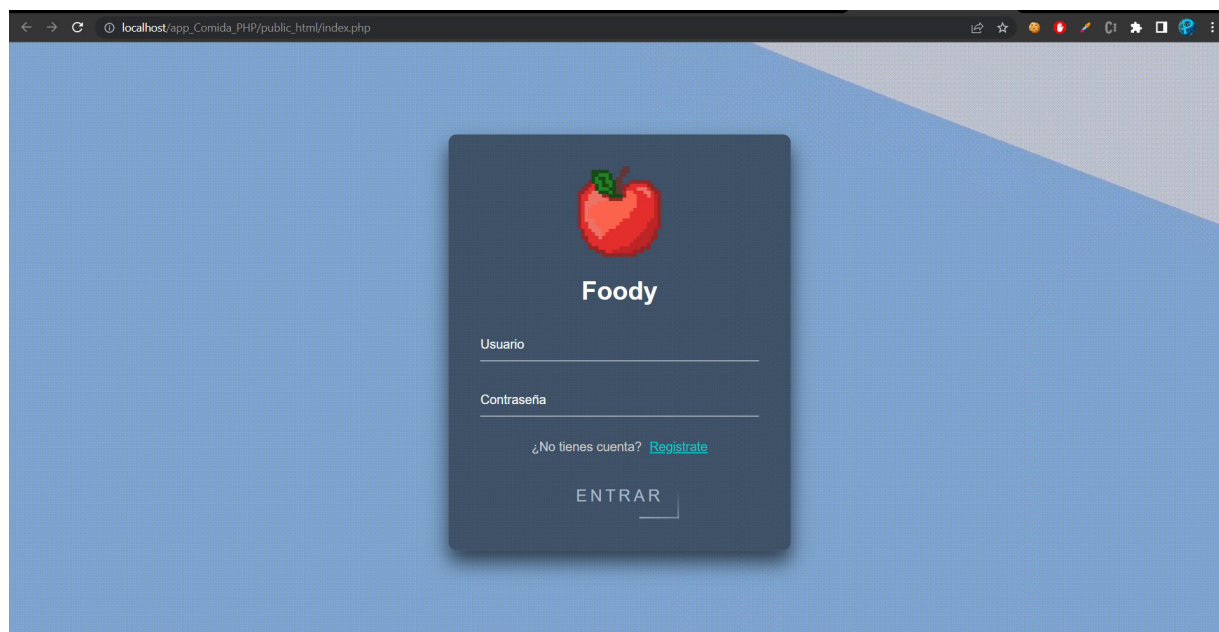
- Algo común también en todas las páginas y no menos importante es el include de página de funciones.php, donde se almacenan todas las funciones creadas para el funcionamiento de la página.
- El botón de buscar le permite a cualquier usuario poder usar el buscador para filtrar los productos que quiere ver, dentro de los productos disponibles de cada usuario.

## Index.php

Al entrar en el login comprueba si existe una sesión iniciada, y en caso de existir le redirige a home.php. Además, comprueba si existe la base de datos con el usuario y la contraseña que vamos a usar y en caso de no existir crea esta base de datos.

También comprueba si las credenciales introducidas en el login son correctas mediante una consulta a la base de datos. En esta comprobación la contraseña la ciframos para su comprobación mediante sha256 y estas contraseñas también se almacenan cifradas en la base de datos.

Si el usuario o la contraseña no son correctas lo muestro con un error y si son correctas guardo en la sesión el usuario que ha iniciado sesión, su rol, su token y también su hora de login como hora de última actividad en una cookie.



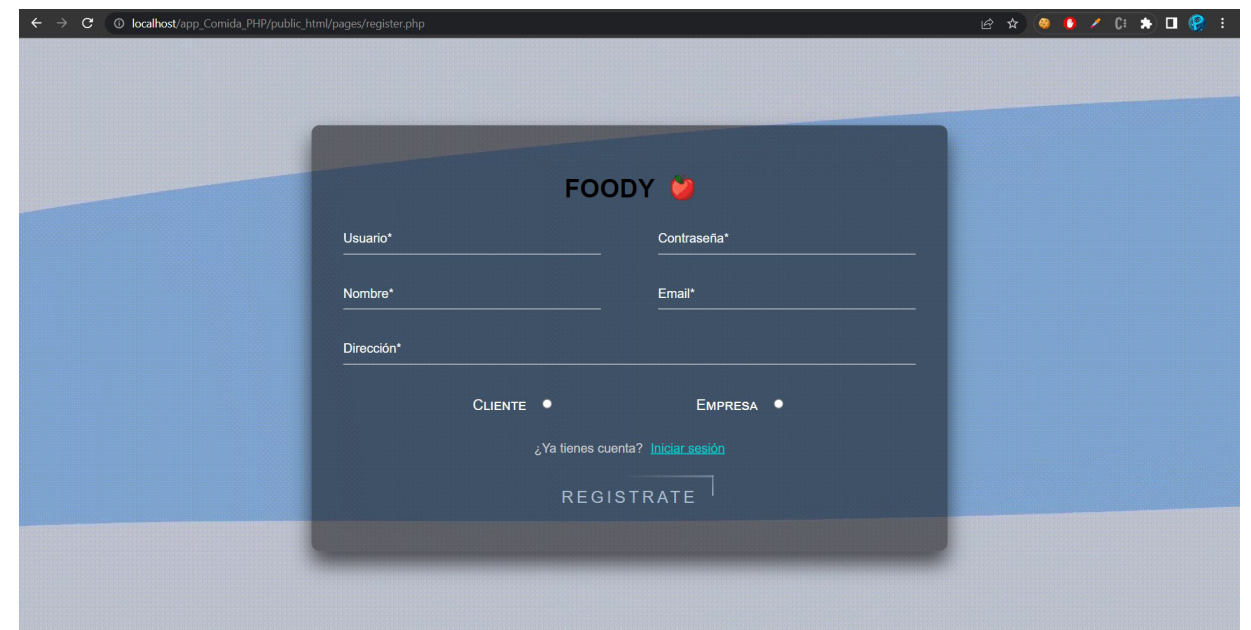
## Registro.php

La página de registro también comprueba si hay alguna sesión activa y si la base de datos existe. Si el registro es correcto creo la sesión con su usuario, su rol y su token.

También guardamos en una cookie la hora de registro como hora de última actividad al igual que en el registro, por lo que registrarse correctamente tendrá las mismas implicaciones que hacer login correctamente.

Además, también incluyo el nuevo registro del usuario creado en la base de datos si los datos son correctos. Para ello primero compruebo que todos los campos tengan contenido y guardo la contraseña cifrada con sha256.

En caso de que haya algún error por campos incompletos avisamos al usuario y no se realiza el registro, al igual que si el usuario introducido está repetido, que tampoco realizo el registro y aviso al usuario de que elija otro user.





# Home.php

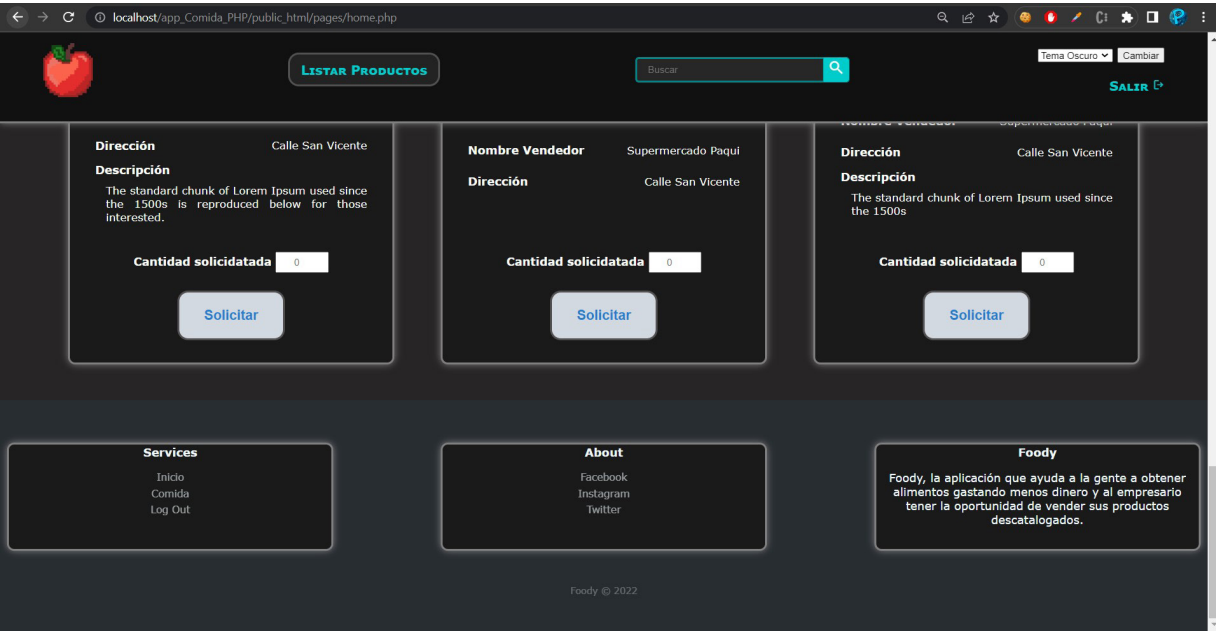
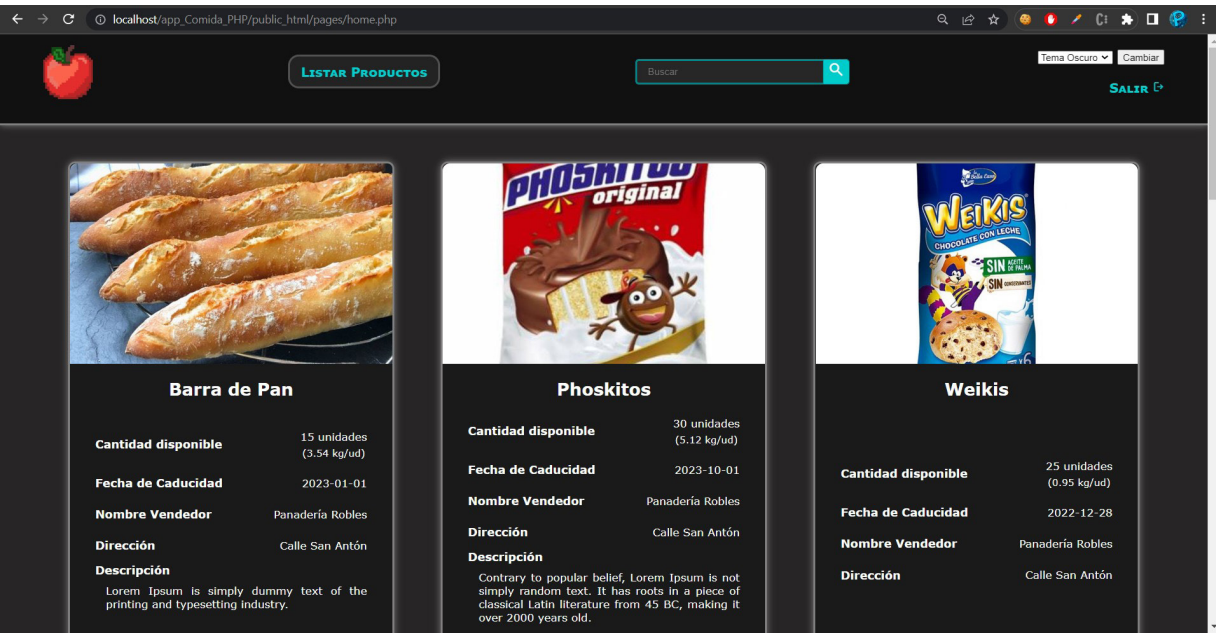
Es la página principal de la aplicación y no se podrá acceder a ella si el usuario no tiene una sesión iniciada.

La información y las funcionalidades en esta página dependerá del rol que tenga el usuario que ha iniciado sesión.

Si es un cliente (rol: user) podrá listar todos los productos con stock disponible y podrá solicitar una cantidad de unidades de estos productos. Al solicitar una cantidad se redirige a addPedido.php, donde se valida que el usuario haya introducido un número entero entre 0 y la cantidad disponible. En caso de ser así se generará un pedido del cliente a dicho producto y se restará la cantidad solicitada de el stock disponible del producto, en caso de no ser así se indicará al usuario el error.

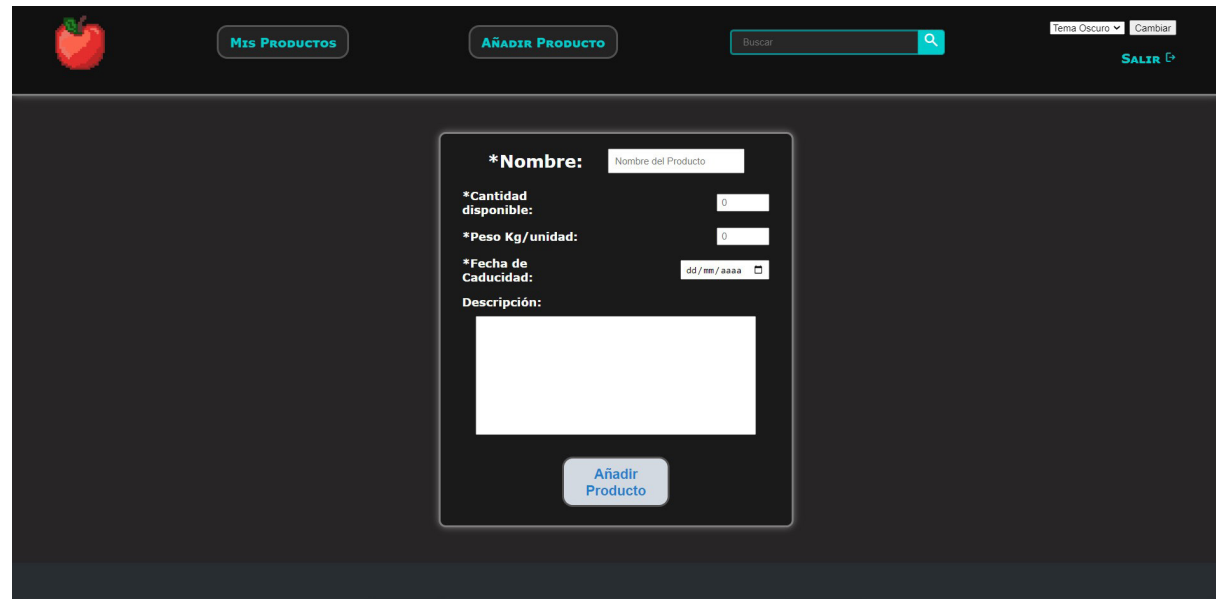
Si es una empresa (rol: empresa) podrá listar sus productos y borrar dichos productos que le pertenezcan. Si estos productos tienen algún pedido asociado se le avisara de que eliminar dicho producto conlleva eliminar también sus pedidos asociados (Estos pedidos eliminados se almacenarán en orders\_delete.log). Además, las empresas puede acceder a addItem.php y publicar un nuevo producto, que será añadido a la base de datos si los datos son correctos.

Si se trata de un administrado (rol: admin) el usuario podrá ver todos los productos de la base de datos, tengan o no stock, y podrá eliminarlos si lo desea. Al igual que para las empresas, también se le indicará si el producto tiene pedidos activos y en caso de ser así se eliminarán dichos pedidos asociados al producto y se guardará esta información en orders\_delete.log.



## addItem.php

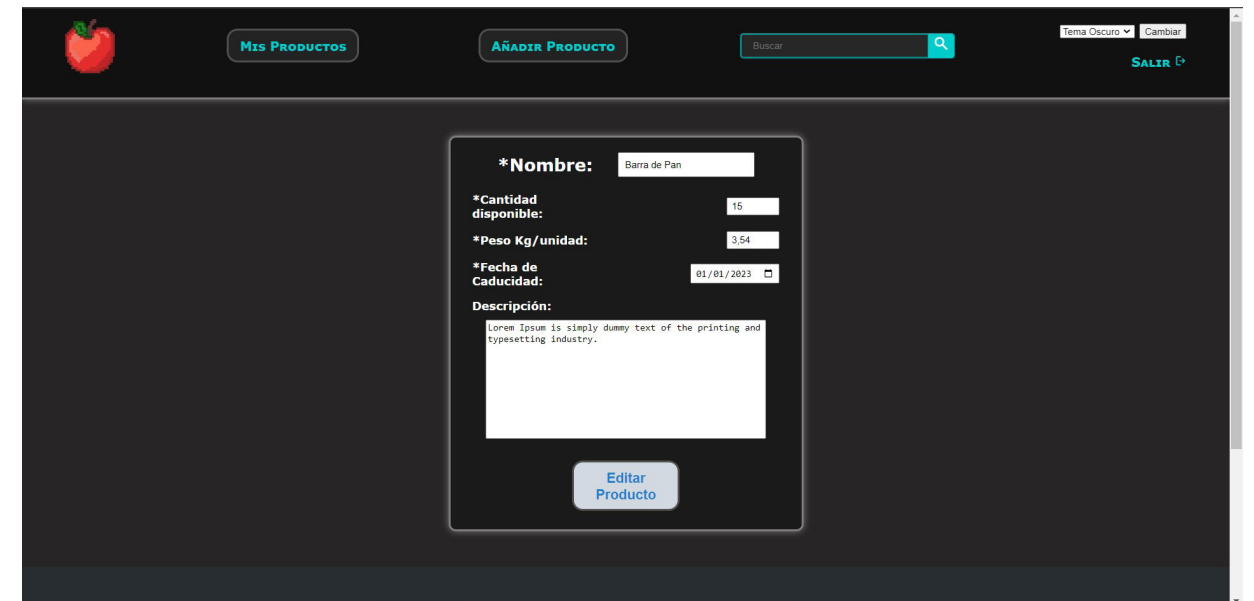
Es página sirver para añadir un producto nuevo a la empresa, lo unico que tiene es un formulario con un boton que genera un insert que mete en la base de datos. Valida que la fecha introducida en el producto no puede ser inferior a la fecha actual.



The screenshot shows a web application interface for adding a new product. The header includes a logo of a red apple, navigation buttons for 'Mis Productos' and 'AÑADIR PRODUCTO', a search bar, and a dark theme toggle. The main content area features a form with the following fields: a required name field (\*Nombre:), a quantity field (\*Cantidad disponible:), a weight field (\*Peso Kg/unidad:), a date field (\*Fecha de Caducidad:), and a description field (Descripción:). A blue 'Añadir Producto' button is located at the bottom of the form.

## editItem.php

La funcionalidad de esta pagina sirver para modificar uno de los productos creados, tiene el mismo formulario que la pagina de addItem.php



The screenshot shows a web application interface for editing an existing product. The header is identical to the addItem.php page. The main content area features a form with the following fields: a required name field (\*Nombre:), a quantity field (\*Cantidad disponible:), a weight field (\*Peso Kg/unidad:), a date field (\*Fecha de Caducidad:), and a description field (Descripción:). A blue 'Editar Producto' button is located at the bottom of the form.

Hemos validado el código html y css de todas las páginas.

# FUNCIONES

A continuación veremos todas las funciones creadas para el funcionamiento de la pagina:

- **Función `filtrarInput()`:**

La funcion **`filtrarInput()`** se utiliza para filtrar un input mediante `htmlspecialchars()`.

Tendremos las variables **`$input`** que sera el nombre de la variable input que vamos a filtrar, la variable **`$method`** que servira para indicar el tipo de método utilizado ("GET" o "POST") y la variable que se crea dentro del **`if`** llamada **`$variableFiltrada`** que sera lo que la función devuelva, la función lo que realiza es comparar el método que se manda y filtrar el input, comprobando primero si la variable esta inicializada o no, en caso de que no este inicializada devolvera null.

```
function filtrarInput($input, $metodo) {
    if ($metodo === "POST") { //Si el método es POST
        $variableFiltrada = isset(filter_input_array(INPUT_POST)[$input]) ? htmlspecialchars(filter_input_array(INPUT_POST)[$input]) : null;
    } else if ($metodo === "GET") { //Si el método es GET
        $variableFiltrada = isset(filter_input_array(INPUT_GET)[$input]) ? htmlspecialchars(filter_input_array(INPUT_GET)[$input]) : null;
    }
    return $variableFiltrada;
}
```

A la hora de utilizar la función pondremos el nombre de la funcion y seguidamente las dos variables que le queremos pasar.

```
filtrarInput("token", "POST");
```

- **Función `filtrarArrayInput()`**

La función **`filtrarArrayInput()`** sirve para filtrar un input POST tipo array. Filtra hasta dos niveles de array anidados con `htmlspecialchars()` y si la variable **`$clavesAComprobar`** esta vacía (1 nivel).

Tendremos las variables **`$arrayInputName`** que sera el nombre de la variable array a filtrar, **`$clavesAComprobar`** el cual sera el array con las claves de los campos que se quieren comprobar si están vacíos, **`$errorInputVacio`** referencia a un booleano que será false si alguna de las claves a comprobar esta vacía y la variable **`$arrayInputs`** que devolvera el array POST filtrado y puede cambiar el valor del parámetro que pasemos como **`$errorInputVacio`**.

Lo primero que realiza la función es comprobar que **`$arrayInputName`** este inicializado en caso de estarlo lo filtrara y entrara en la siguiente condicion que esta filtrara con **`htmlspecialchars()`** todos los campos del array y seguidamente comprobaremos si existen todos los campos necesarios y si estan vacios, en caso de que no exista o si exista y este vacío **`$errorInputVacio`** cambiara a true y devolvera un error.

```
function filtrarArrayInput($arrayInputName, $clavesAComprobar, $errorInputVacio) {
    $arrayInputs = isset(filter_input_array(INPUT_POST)[$arrayInputName]) ? filter_input_array(INPUT_POST)[$arrayInputName] : null;
    if (isset($arrayInputs)) { //Si el array existe
        //Filtro con htmlspecialchars todos los campos del array
        foreach ($arrayInputs as $value) {
            $value = htmlspecialchars($value);
        }
        //Compruebo si los campos necesarios existen y si están vacios
        foreach ($clavesAComprobar as $inputs) {
            if (!isset($arrayInputs[$inputs]) || (isset($arrayInputs[$inputs]) && trim($arrayInputs[$inputs]) == "")) { //Si no existe o si existe y está vacío
                $errorInputVacio = true; //Cambio el valor del error a true
            }
        }
    }
    return $arrayInputs;
}
```

A la hora de usar la funcion la utilizaremos de esta manera, pondremos el nombre de la funcion y entre parentesis le daremos el primer parametro que sera el nombre del array al filtrar, el segundo parametro sera el array de campos a comprobar y por ultimo la variable de error.

```
filtrarArrayInput("updateItem", ["nombre", "stock", "kg_ud", "fechaCaducidad"], $errorEditarProducto);
```

# FUNCIONES

- **Función imprimirOptions()**

La función **imprimirOptions()** sirve para mostrar en código html los options de un select.

Esta función contiene dos variables la cual la primera es **\$array** que sera el array con todos los options que se desean mostrar en el select y la otra variable sera **\$select** que sera la opción que aparecerá seleccionada de todos los options.

La función crea todos los option que le pasemos en el parametro **\$array** y dara el atributo select a la option indicada en la variable **\$select**.

```
function imprimirOptions($array, $select) { //Función para imprimir select y seleccionar una si ya la había elegido previamente
    foreach ($array as $value) {
        if ($value == $select) { //Si había seleccionado antes una opción
            echo "<option value = '$value' selected>$value</option>";
        } else { //Si no ha seleccionado nada
            echo "<option value = '$value'>$value</option>";
        }
    }
}
```

- **Función checkBD()**

La función **checkBD()** comprueba la conexión (true) o no (false) de la base de datos indicada en los parametros.

Esta función contiene las variables **\$conexionDB** que sera la cadena de conexión con la BD, **\$user** sera el usuario de la BD y **\$pass** que sera la contraseña de la BD. Esta funcion devolvera un boolean si la conexión se realiza devolver true y en caso de que falle devolveria false, ya que entraria en la excepción del **try catch**.

La función realiza un **try catch** en el cual en el **try** tendremos la conexion a la BD y en el catch tendremos la excepción que se daria en caso de error la cual guardaria ese error con la fecha en el archivo de **error\_log.log**.

```
function checkBD($conexionDB, $user, $pass, $rutaLog = "../logs/error_log.log") {
    try {
        $bd = new PDO($conexionDB, $user, $pass);
        //Se cierra la conexión
        $bd = null;
    } catch (Exception $ex) {
        error_log(date("F j, Y, g:i a") . " - Error con la base de datos: " . $ex->getMessage() . "\n", 3, $rutaLog);
        return false;
    }
    return true;
}
```

A la hora de utilizar la función la utilizaremos de la siguiente manera llamando a la función y entre los parentesis le pasaremos la conexión de la BD, el usuario y la contraseña.

```
checkBD("mysql:dbname=appcomida;host=127.0.0.1", "root", "");
```

- **Función createBD()**

La función **createBD()** sirve como su propio nombre indica para crear la base de datos.

Esta función contiene las variables **\$query** que sera el script MySQL a ejecutar y las mismas variables que hemos visto en la función anterior que son **\$conexionDB**, **\$user** y **\$pass**.

En esta función también se realizara un **try catch** en la parte de **try** tendremos de nuevo la conexión a la BD y la siguiente intrucción sera ejecutar el script con la funcion de php **query()** en caso de excepción mandara un error.

```
function createBD($query, $conexionDB, $user, $pass, $rutaLog = "../logs/error_log.log") {
    try {
        $bd = new PDO($conexionDB, $user, $pass);
        //Ejecutamos la query
        $bd->query($query);
        //Se cierra la conexión
        $bd = null;
    } catch (Exception $ex) {
        error_log(date() . " - Error con la base de datos: " . $ex->getMessage() . "\n", 3, $rutaLog);
    }
}
```



# FUNCIONES

A la hora de utilizar la función la aplicaremos de la siguiente manera, el primer parametro sera el script de MySQL, el segundo parametro sera la cadena de conexión de la base de datos, el tercero y cuarto serán uno el usuario y otro la contraseña de la base de datos.

```
createBD($queryBD, "mysql:host=127.0.0.1", "root", "");
```

- **Función checkUser()**

La función **checkUser()** se utiliza para comprobar si los parámetros introducidos existen en la base de datos (si el login es correcto).

La función contiene las variables **\$conexionDB**, **\$user** y **\$pass** ya que necesitaremos conectarnos a la base de datos, **\$userLogin** que sera el input\_login del usuario, **\$passLogin** que sera el input\_login de la contraseña y la variable **array** que devolvera un array del idUsuario y su rol en caso de que sea correcto, en caso contrario devuelve un array vacío.

La función realiza un **try catch** en el cual en el **try** tendremos la conexión a la BD, **\$loginSQL** que sera la consulta que realizaremos a la base de datos a través de la función **prepare()**, también se creara una variable **\$login** que se utilizara para saber si existe un registro en la tabla usuarios en el que el user y pass son las introducidas en el login, en caso de estar en la base de datos recorreremos el array creado anteriormente y guardaremos en nuestra variable **\$userChecked** el userId y el rol, esto sera lo que nos devuelva la función en caso de estar en la base de datos en caso de que no estuviera entraria en el **catch** y saltaria la excepción con el error.

```
function checkUser($conexionDB, $user, $pass, $userLogin, $passLogin, $rutaLog = "../logs/error_log.log") {
    $userChecked = [];
    try {
        $bd = new PDO($conexionDB, $user, $pass);
        $loginSQL = "SELECT userId, pass, rol FROM usuarios WHERE userId = :userId AND pass = :pass";
        $preparada_user = $bd->prepare($loginSQL);
        $preparada_user->execute(array(":userId" => $userLogin, ":pass" => $passLogin));
        $login = ($preparada_user->rowCount() == 0) ? false : true;
        if ($login) { //Si las credenciales son correctas
            foreach ($preparada_user as $row) { //Guardamos el usuario y su rol
                $userChecked[0] = $row['userId'];
                $userChecked[1] = $row['rol'];
            }
        }
        //Se cierra la conexión
        $bd = null;
    } catch (Exception $ex) {
        error_log(date("F j, Y, g:i a") . " - Error con la base de datos: " . $ex->getMessage() . "\n", 3, $rutaLog);
    }
    return $userChecked;
}
```

A la hora de utilizar la función la ejecutaremos de la siguiente manera, primero pondremos las variables que necesitamos para conectarnos a la base de datos y luego las variables user y pass de los login.

```
checkUser("mysql:dbname=appcomida;host=127.0.0.1", "root", "", $datosRegistro["userId"], $datosRegistro["pass"]);
```

- **Función insertInBD()**

La función **insertInBd()** se utiliza para insertar un array (con valores numéricos o string) en la BD.

La función contiene las variables **\$conexionDB**, **\$user** y **\$pass** ya que necesitaremos conectarnos a la base de datos, **\$table** que sera la tabla de la BD a la que vamos a hacer el insert, **\$arrayUser** sera el array con los datos a insertar en **\$table**.

La función introduce un nuevo registro, en la \$table que le pasemos como parametro, los valores a introducir se los pasaremos en un array en el que sus claves, tendran el mismo nombre que los campos de la tabla en la que vamos a hacer la inserción.

# FUNCIONES

```
function insertInBD($conexionDB, $userDB, $passDB, $table, $arrayInsert, $rutaLog = "../../../logs/error_log.log") {
    $errorAddUser = false;
    try {
        //Hacemos la conexión a la BD
        $bd = new PDO($conexionDB, $userDB, $passDB);
        //Concatenamos todas las variables del array a dos strings (uno con las claves y otro con los valores) para poder utilizar la función con cualquier inserción a la BD con strings o int
        $values = "";
        $keys = "";
        foreach ($arrayInsert as $key => $value) { //Concatenamos el nombre de los campos (su clave) y los valores de dichas variables
            if (is_string($value)) {
                $values .= "'$value'";
            } else if (is_numeric($value)) {
                $values .= "$value";
            }
            $keys .= "$key";
            if ($key !== array_key_last($arrayInsert)) { //Si no es el último valor del array pongo la coma
                $values .= ", ";
                $keys .= ", ";
            }
        }
        //Query MySQL de inserción: introduciremos la tabla de la que se trata ($table), los campos ($keys) y los valores a insertar ($values)
        $queryInsert = "INSERT INTO $table ($keys) values ($values)";
        //Ejecutamos la query
        $bd->query($queryInsert);
        //Se cierra la conexión
        $bd = null;
    } catch (Exception $ex) {
        if (str_contains($ex->getMessage(), "1062")) { //Si salta el mensaje de clave primaria repetida
            $errorAddUser = true;
        }
        error_log(date("F j, Y, g:i a") . " - Error con la base de datos: " . $ex->getMessage() . "\n", 3, $rutaLog);
    }
    return $errorAddUser;
}
```

A la hora de utilizar la función la utilizaremos de la siguiente manera pondremos los parametros para poder ejecutar la base de datos, tambien le indicaremos la tabla donde se va a hacer el insert y el array a insertar.

```
insertInBD("mysql:dbname=appcomida;host=127.0.0.1", "root", "", "pedidos", $pedido);
```

- **Función deleteInBD()**

La función deleteInBD() sirve para borrar de la base de datos el producto que queramos borrar.

La función contiene las variables **\$conexionDB**, **\$user** y **\$pass** ya que necesitaremos conectarnos a la base de datos, **\$table** que sera la tabla de la que haremos la consulta, **\$pk** que sera el campo y **\$pkDelete** sera el campo que queremos borrar.

La función realiza un **try catch** en el cual en el **try** tendremos la conexion a la BD, y ejecutara la consulta de borrado del campo que hallamos seleccionado en caso de error entrara en la excepción.

```
function deleteInBD($conexionDB, $userDB, $passDB, $table, $pk, $pkDelete, $rutaLog = "../../../logs/error_log.log") {
    try {
        //Hacemos la conexión a la BD
        $bd = new PDO($conexionDB, $userDB, $passDB);
        //Query MySQL de borrado: introduciremos la tabla de la que se trata ($table), el campo de la PK ($pk) y el valor de dicho campo ($pkDelete)
        $queryInsert = "DELETE FROM $table WHERE $pk = $pkDelete";
        //Ejecutamos la query
        $bd->query($queryInsert);
        //Se cierra la conexión
        $bd = null;
    } catch (Exception $ex) {
        error_log(date("F j, Y, g:i a") . " - Error con la base de datos: " . $ex->getMessage() . "\n", 3, $rutaLog);
    }
}
```

A la hora de utilizar la funcion la ejecutaremos de la siguiente manera, primero pondremos las variables que necesitamos para conectarnos a la base de datos y luego pondremos la tabla, el id del producto que queramos borrar.

```
insertInBD("mysql:dbname=appcomida;host=127.0.0.1", "root", "", "pedidos", $pedido);
```

- **Función updateInBD()**

La función updateInBD() sirve para actualizar un producto de la base de datos.

La función contiene las variables **\$conexionDB**, **\$user** y **\$pass** ya que necesitaremos conectarnos a la base de datos, **\$table** que sera la tabla de la que haremos la consulta, **\$pk** que sera el campo y **\$pkUpdate** sera el valor que queremos actualizar.

```
function updateInBD($conexionDB, $userDB, $passDB, $table, $pk, $pkUpdate, $fieldName, $newValue, $rutaLog = "../../../logs/error_log.log") {
    try {
        //Hacemos la conexión a la BD
        $bd = new PDO($conexionDB, $userDB, $passDB);
        //Query MySQL de actualización:
        $queryInsert = "UPDATE $table SET $fieldName = $newValue WHERE $pk = $pkUpdate";
        //Ejecutamos la query
        $bd->query($queryInsert);
        //Se cierra la conexión
        $bd = null;
    } catch (Exception $ex) {
        error_log(date("F j, Y, g:i a") . " - Error con la base de datos: " . $ex->getMessage() . "\n", 3, $rutaLog);
    }
}
```

A la hora de utilizar la funcion ingresaremos lo primero los parametros de la base de datos, lo segundo la tabla, lo tercero es el campo de la clave primaria, el cuarto el valor de la clave, el quinto el nombre del campo que se quiere actualizar y el ultimo el valor del campo a actualizar.

# FUNCIONES

```
updateInBD("mysql:dbname=appcomida;host=127.0.0.1", "root", "", "productos", "idProducto", $pedido["idProducto"], "stock", $stockDisponible);
```

## • Función logOutInactivity()

La función **logOutInactivity()** sirve para desloguear al usuario si supera el el tiempo de inactividad.

La función contiene las variables **\$now** que sera la variable que se utiliza para guardar la fecha actual de la comprobación, **\$lastActivity** sera la variable que guarda la ultima actividad de la hora y **\$secondsAllowed** sera la variable que guarda los segundos permitidos de inactividad.

Lo función lo que hace es comparar la fecha actual con la ultima fecha de la actividad en caso de ser mayo el tiempo de inactividad que el permitido devolviera true y cerrara la sesión.

```
function logOutInactivity($now, $lastActivity, $secondsAllowed) {  
    if ((strtotime($now) - strtotime($lastActivity)) > $secondsAllowed) { //Si el tiempo de inactividad es mayor al permitido  
        return true;  
    } else { //Si no ha excedido el tiempo de inactividad  
        return false;  
    }  
}
```

A la hora de utilizar la consulta lo primero que haremos es ingresar la fecha actual, lo segundo sera poner la variable de la ultima vez que tenemos guardada y por ultimo le indicamos el tiempo que queremos que cuando se sobrepase se desloge la sesión.

```
logOutInactivity(date("Y-n-j H:i:s"), $horaUltimaActividad, 300)
```

## • Función selectQuery()

La función selectQuery() devuelve el objeto de la query que se la pasa como parámetro.

La función contiene las variables **\$conexionDB**, **\$user** y **\$pass** ya que necesitaremos conectarnos a la base de datos, **\$query** sera la variable que llevara la consulta que se desea realizar.

La función ejecuta una consulta que nos devolviera un objeto con todas los resultados de es consulta.

```
function selectQuery($conexionDB, $userDB, $passDB, $query, &$amp;resultados, $rutaLog = "../logs/error_log.log") {  
    $select = null;  
    try {  
        $bd = new PDO($conexionDB, $userDB, $passDB);  
        $select = $bd->query($query);  
        if ($select->rowCount() === 0) {  
            $resultados = false;  
        }  
        //Se cierra la conexión  
        $bd = null;  
    } catch (Exception $ex) {  
        error_log(date("F j, Y, g:i a") . " - Error con la base de datos: " . $ex->getMessage() . "\n", 3, $rutaLog);  
    }  
    return $select;  
}
```

A la hora de utilizarla lo primero sera poner los parametros de la base de datos, la consulta que queramos realizar y un boolean que sera false en caso de que no se encuentre ningun resultado para esa consulta.

```
selectQuery("mysql:dbname=appcomida;host=127.0.0.1", "root", "", $query, $resultados);
```



# FUNCIONES

- **Función listarProductos()**

La función recorre el objeto con todos los resultados de una consulta que se ha hecho anteriormente y recorre todos los campos de cada registro y muestra por pantalla la información de cada producto.

Dependiendo del rol del usuario se le muestran unos botones y funcionalidades diferentes.

```
function listarProductos($items, $rol, $token) {
    foreach ($items as $item) { //Recorro todos los productos
        echo "<div class='item'>";
        echo "<h2 class='item_title'>" . $item['Nombre del Producto'] . "</h2>";
        echo "<ul>";
        foreach ($item as $key => $value) { //Recorro cada campo de cada producto
            if (is_string($key) && $value != "" && !str_contains($key, "key") && $key != "Nombre del Producto") { //Si las claves no son string, el campo está vacío, o se trata de algún identificador, no lo muestro
                if ($key != "Peso") {
                    echo "<li class='item_li'>";
                    echo "<h3 class='li_title'>$key</h3><p class='li_text'>$value";
                    if ($key == "Cantidad disponible") { //Si es la cantidad disponible indico que se trata de Kg
                        echo " unidades<span class='item_peso'> (" . $item['Peso'] . " kg/ud)</span>";
                    }
                    echo "</p></li>";
                }
            }
        }
    }

    if ($rol == "cliente") {
        echo "<form class='item_li' method='POST' action='./addPedido.php'>";
        echo "<label class='li_title'>Cantidad solicitada</label> <input type='number' name='cantidadPedido' step='.01' placeholder='0' class='item_input' min='0' max='' . $item['Cantidad disponible'] . ''>";
        . "<input type='hidden' name='idProducto' value=' . $item['keyProducto'] . '><input type='hidden' name='idEmpresa' value=' . $item['keyEmpresa'] . '>";
        . "<input type='hidden' name='token' value='' . $token . ''>";
        . "<button type='submit' class='item_btn'>Solicitar</button>";
        echo "</form>";
    } else {
        //Compruebo si el producto tiene algún pedido y si es el caso lo incluyo en un input hidden:
        $query = "SELECT * FROM pedidos WHERE idProducto = " . $item['keyProducto'] . " ";
        $resultados = true;
        selectQuery("mysql:dbname=appcomida;host=127.0.0.1", "root", "", $query, $resultados);
        echo "<form class='item_li' method='POST' action='./removePedido.php'>";
        if ($resultados) { //si el producto tiene pedidos en la BD
            echo "<input type='hidden' name='pedidos' value='true'>";
            echo "<p class='item_avisoProducto'>Producto con pedidos activos (si se elimina también se eliminarán sus pedidos asociados)</p>";
        }
        echo "<input type='hidden' name='idProducto' value=' . $item['keyProducto'] . '><input type='hidden' name='idEmpresa' value=' . $item['keyEmpresa'] . '>";
        . "<input type='hidden' name='token' value='' . $token . ''>";
        . "<button type='submit' class='item_btn'>Eliminar</button>";
        echo "</form>";
    }

    echo "</div>";
}
}
```

A la hora de utilizar la función le pasaremos los parametros de la consulta que queremos realizar y el token de la sesión.

```
listarProductos($productos, $rol, $tokenSession);
```

- **Función checkStock()**

La función checkStock sirve para comprobar que cuando se vaya a solicitar una cantidad de cualquier producto la cantidad que se solicite sea menor a la cantidad de stock.

Las variables a utilizar serán lo primero los parametros de la base de datos, **\$idProducto** el id del producto que se a solicitado y **\$cantidadPedido** la cantidad solicitada.

```
function checkStock($conexionDB, $user, $pass, $idProducto, $cantidadPedido, $rutaLog = "../logs/error_log.log") {
    $stock = -1;
    try {
        $bdd = new PDO($conexionDB, $user, $pass);
        $stockSQL = "SELECT * FROM productos WHERE idProducto = $idProducto AND stock >= :cantidadSolicitada";
        $preparada_stock = $bdd->prepare($stockSQL);
        $preparada_stock->execute(array(":cantidadSolicitada" => $cantidadPedido));
        $stockSuficiente = ($preparada_stock->rowCount() == 0) ? false : true;
        if ($stockSuficiente) { //Si hay stock suficiente compruebo si ha solicitado todo el stock restante o no
            foreach ($preparada_stock as $row) { //Comprobamos si el stock es todo el disponible
                $stock = $row['stock'] - $cantidadPedido;
            }
        }
        //Se cierra la conexión
        $bdd = null;
    } catch (Exception $ex) {
        error_log(date("F j, Y, g:i a") . " - Error con la base de datos: " . $ex->getMessage() . "\n", 3, $rutaLog);
    }
    return $stock;
}
```

A la hora de utilizar la función le pasamos los parametros para entrar en la base de datos, el valor de la cantidad y el valor del stock.

```
checkStock("mysql:dbname=appcomida;host=127.0.0.1", "root", "", $pedido["idProducto"], $pedido["cantidad"])
```



# ESTRUCTURA BD

