



IES ANA LUISA BENÍTEZ
Las Palmas de Gran Canaria

**2do Ciclo Formativo de Grado
Superior:
Desarrollo de Aplicaciones Web**

**Asignatura:
Desarrollo Web en Entorno Servidor
(DSW)**

**Unidad 6 (Recuperación):
Proyecto “**Valoración con Estrellas**”**

**Richard Francisco Vaca Garcia
09/02/2025**

Indice

| | |
|-------------------------------------|---|
| 1. views/layout/footer.php..... | 3 |
| 2. views/layout/header.php..... | 3 |
| 3. views/login/index.php..... | 3 |
| 4. views/productos/borrar.php..... | 4 |
| 5. views/productos/crear.php..... | 4 |
| 6. views/productos/detalle.php..... | 4 |
| 7. views/productos/listado.php..... | 5 |
| 8. views/productos/update.php..... | 5 |
| 9. public/js/app.js..... | 6 |

Desarrollo Web en Entorno Servidor (DSW): UT6 – Proyecto “Valoración con Estrellas”

1. views/layout/footer.php

Este archivo representa el pie de página y la inclusión de scripts necesarios para tu aplicación.

- Muestra un elemento `<footer>` con un mensaje de copyright dinámico (utilizando `date("Y")` para el año).
- Incluye la librería Axios a través de una CDN, para que puedas realizar peticiones AJAX (GET, POST, etc.) de manera sencilla en el lado del cliente.
- Por último, carga el fichero `app.js`, que contiene la lógica de JavaScript de tu aplicación (por ejemplo, el proceso de login).

En resumen, **footer.php** cierra el HTML iniciado en el header, agrega un mensaje al final de la página y prepara los scripts para que el proyecto funcione.

2. views/layout/header.php

Este archivo prepara la **estructura inicial** y la apariencia base que se verá en todas las vistas que lo incluyan.

- Declara el DOCTYPE, el idioma del documento, y metaetiquetas para definir la codificación de caracteres y la responsividad en dispositivos móviles.
- Inyecta el título del proyecto en la pestaña del navegador (por ejemplo, “RFVG: Valoración con Estrellas”).
- Enlaza la hoja de estilos principal `estilos.css` para dar formato a la página, y también carga una librería de iconos (ej. Font Awesome) que se usan para mostrar estrellas o cualquier otro ícono.
- Contiene un encabezado con un título y un pequeño menú de navegación (por ejemplo, enlaces para “Productos” y “Cerrar sesión”).
- Comienza la etiqueta `<main>` para alojar el contenido propio de cada vista.

En resumen, **header.php** agrupa toda la configuración visual y estructural que comparten las distintas páginas del proyecto.

3. views/login/index.php

Esta vista contiene la **interfaz de inicio de sesión**.

- Incluye el header al comienzo para mantener la coherencia visual y estructural.

- Dentro de un contenedor (por ejemplo, `login-container`), presenta un formulario que pide usuario y contraseña.
- El formulario incluye un `id` que, desde JavaScript (`app.js`), se utiliza para gestionar el envío de datos y validar la respuesta del servidor.
- Hay también un apartado para mostrar mensajes de error o éxito (por ejemplo, si el usuario introdujo credenciales erróneas).
- Se utiliza un pequeño tooltip que puede mostrar ejemplos de usuarios y contraseñas de prueba.
- Por último, cierra el contenedor de login y carga el footer.

En resumen, **index.php** del login gestiona la parte visual para que el usuario se autentique y, si todo sale bien, se redirija a la zona de productos.

The screenshot shows a web browser window with the URL `localhost:3000/public/index.php?controller=Login&action=login`. The page has a teal background and features the title **Proyecto UT6: Valoración con Estrellas**. Navigation links for **Productos** and **Cerrar sesión** are visible. A dark grey tooltip displays test credentials: **Usuario1: admin, 1234** and **Usuario2: ana, 4321**. The main heading is **Iniciar Sesión**, followed by a login form with fields for **Usuario:** (containing 'admin') and **Contraseña:** (masked with dots). A blue **Entrar** button is at the bottom of the form. The footer text reads: **© 2025 Richard Francisco Vaca Garcia, 2do CFGS DAW**.

4. views/productos/borrar.php

Esta vista despliega una **pantalla de confirmación** antes de eliminar un producto.

- Muestra un mensaje preguntándole al usuario si está seguro de la eliminación (indicando el nombre del producto).
- Presenta un formulario con un campo oculto que contiene el `id` del producto.

- Tiene un botón “Eliminar” que confirma la acción y envía la solicitud al controlador correspondiente (por ejemplo, `ProductosController`).
- Hay un enlace de “Cancelar” para que, si el usuario se arrepiente, regrese al listado de productos.
- Incluye la cabecera y el pie de página, como en el resto de vistas.

En resumen, **borrar.php** te permite, de forma segura, confirmar la eliminación y así evitar borrar un producto por error.

5. `views/productos/crear.php`

Esta vista presenta el **formulario para crear un producto nuevo**.

- Incluye la cabecera al principio.
- Tiene un título descriptivo (por ejemplo, “Crear Producto”).
- Muestra los campos necesarios: nombre, descripción y precio, todos obligatorios.
- El método de envío es POST y apunta al controlador de productos con la acción “crear”.
- Al final se incluye el footer para cerrar la estructura.

En resumen, **crear.php** da la interfaz para que el usuario introduzca un producto en la base de datos.

6. `views/productos/detalle.php`

Esta vista muestra los **detalles completos de un producto**.

- Presenta un título “Detalles del Producto” y luego la información concreta: nombre, descripción y precio.
- Ofrece enlaces para editar o eliminar el producto (básicamente, botones que apuntan a las acciones “update” y “borrar”).
- Antes de eliminar, lanza una pequeña confirmación en el navegador (un `confirm()`) para asegurarse de que el usuario no borre algo por accidente.
- Termina con la inclusión del footer.

En resumen, **detalle.php** sirve para ver en pantalla la información exacta de un producto y acceder de forma rápida a editarlo o borrarlo.

7. `views/productos/listado.php`

Es la **página principal** de productos.

- Incluye el header, da la bienvenida al usuario (por ejemplo, mostrando su nombre a partir de la sesión).
- Muestra un título “Listado de Productos”.
- Incluye una tabla que recorre todos los productos. Por cada uno, enseña su ID, nombre, descripción, precio y valoración.

- Para la valoración, implementa una función que genera estrellas completas, medias o vacías según la puntuación. Además, se muestra la puntuación numérica (por ejemplo, “(3.5/5)”).
- Para cada producto, hay enlaces que llevan a “Ver” (detalle), “Editar” (update) y “Borrar” (borrar).
- Ofrece también un enlace “Crear nuevo producto” que lleva a la vista donde se introduce un producto por primera vez.
- Cierra con el footer.

En resumen, **listado.php** es la vista de inventario o directorio de productos, donde el usuario puede ver, gestionar y navegar hacia las demás acciones (detallar, crear, editar o eliminar).



8. views/productos/update.php

Esta vista muestra el **formulario para actualizar un producto** existente.

- Comienza con la cabecera y un título (“Actualizar Producto”).
- Presenta un formulario con campos para el nombre, la descripción y el precio. Dichos campos se rellenan inicialmente con los valores actuales del producto (para que el usuario vea qué datos tiene y pueda modificarlos).
- Un campo oculto (id) almacena la clave primaria del producto, para que el controlador sepa qué producto debe actualizar al enviar el formulario.
- Al final, se incluye el footer.

En resumen, **update.php** sirve para editar datos de un producto que ya existe, permitiendo cambiar cualquiera de sus atributos (nombre, descripción, precio) y guardar esos cambios.

9. `public/js/app.js`

Este archivo contiene la **lógica de JavaScript** que se ejecuta en el navegador, y actualmente se centra en manejar el **formulario de inicio de sesión**. A grandes rasgos:

1. Detección del DOM cargado

Se utiliza un `document.addEventListener("DOMContentLoaded", ...)` para asegurarse de que el contenido del HTML ya esté disponible antes de manipular elementos o formularios.

2. Selección de elementos clave

- Se busca en el DOM el elemento con `id="loginForm"` para controlarlo (si existe).
- Se obtiene también el contenedor `mensajeResultado`, donde se mostrarán mensajes de error o éxito.

3. Manejo del evento `submit`

- Si se detecta la presencia del formulario de login, se añade un “listener” que, al pulsar el botón de Enviar, previene el envío tradicional (`event.preventDefault()`) y, en lugar de ello, realiza acciones personalizadas.
- Se crea un objeto `FormData` para recopilar los campos del formulario (`usuario` y `password`).

4. Validaciones en el lado del cliente

- Antes de enviar la petición al servidor, se verifica que usuario y contraseña no estén vacíos. Si faltan, se coloca un mensaje de error en color rojo y se detiene el proceso.

5. Petición AJAX con Axios

- Si las validaciones pasan, se hace una petición POST a la URL indicada (el `controller=Login` y `action=login`).
- En el `then(...)` se procesa la respuesta:
 - Si el backend responde con `data.ok === true`, significa que las credenciales son correctas. Se muestra un mensaje en verde y se redirige a la página correspondiente (`data.redirectUrl`), si existe.
 - Si `data.ok === false`, se muestran los errores de credenciales en rojo. Además, si llega un texto “debugOutput”, se abre una ventana emergente mostrando detalles técnicos (simulando un “debug”).
- En el `catch(...)`, se maneja cualquier error de conexión o excepción de Axios, avisando al usuario de que hubo un problema.

6. Responsabilidad principal

- Proporcionar una experiencia de login interactiva y clara.
- Validar campos de forma básica (vacíos o no).

- Enviar credenciales con Axios y procesar la respuesta del servidor (login exitoso o fallido).

En resumen, este archivo **conecta la parte visual del formulario de login con la lógica** para autenticar al usuario, avisarle de posibles errores y, en caso de éxito, llevarlo a la pantalla de productos.