



REPORTE FINAL – TAG SIZE

PROGRAMACIÓN WEB

**ADOLFO CORTÉS CORONA
MARIO DAEL SIERRA HERRERA
ROBERTO DANIEL RAMIREZ DIAZ DE LEON**

24/03/2025

Contenido

Introducción.....	4
Objetivo de la aplicación.....	4
Objetivos Específicos	4
Justificación.....	4
Alcance.....	4
Análisis del Sistema	5
Requerimientos Funcionales.....	5
Requerimientos No Funcionales	5
Módulos del Sistema.....	6
Modelo de Datos.....	6
Herramientas y Tecnologías Utilizadas.....	7
Diseño del Sistema e Interfaz Gráfica	7
Vista Principal (Inicio).....	7
Menu.js	9
Página catalogo:	10
Script.js.....	12
Página de acceso:.....	14
Pagina Registro:.....	16
Signup.php	19
Página Iniciar Sesión:	21
Vista de Acceso (Login)	24
Panel de Administración	25
Gestión de Productos (CRUD)	26
Lista de Productos	27
Agregar Nuevo Producto.....	28
Editar Productos.....	29
Estilo General	30
Desarrollo e Implementación del Sistema.....	30
Estructura General de Archivos.....	30

Conexión a Base de Datos	30
Gestión de Productos (CRUD)	31
Validaciones y Seguridad	36
Control de Sesiones.....	37
Carga y Visualización de Imágenes	37
Conclusiones:	39

Introducción

TagSize es una aplicación web y móvil desarrollada con tecnologías como PHP, HTML5, CSS3 y JavaScript, diseñada para facilitar la experiencia de compra en una zapatería. Permite a los clientes escanear códigos de barras físicos para consultar en tiempo real la disponibilidad de modelos de calzado por talla, color, marca y otros atributos. Este sistema reduce la necesidad de asistencia directa y mejora la eficiencia en puntos de venta físicos.

Objetivo de la aplicación

Diseñar e implementar una aplicación digital que permita a los clientes de una zapatería consultar fácilmente la disponibilidad de productos a través del escaneo de códigos de barras, optimizando el tiempo de compra y la organización del inventario.

Objetivos Específicos

- Desarrollar una interfaz web amigable para el cliente y el administrador.
- Crear una base de datos que almacene la información detallada de cada producto.
- Implementar una funcionalidad de escaneo de código de barras que permita filtrar productos por atributos específicos.
- Desarrollar un módulo de gestión para usuarios administradores con acceso a CRUD de productos.

Justificación

En tiendas físicas de calzado, es común que los clientes pierdan tiempo buscando su talla, color o disponibilidad de un modelo específico. TagSize busca resolver esta problemática mediante una solución intuitiva y rápida basada en el escaneo de códigos de barras. La aplicación se convierte en un asistente virtual que mejora la experiencia del cliente, reduce la carga de trabajo del personal y mejora el control de inventario.

Alcance

- Plataforma web funcional localmente (localhost) accesible mediante navegador.
- Gestión de usuarios y productos (alta, baja, edición).
- Escaneo de productos con lectura de código de barras.
- Control de stock, precios y descripción de cada producto.
- Aplicación orientada a su posterior implementación en dispositivos móviles (con lector de códigos integrado).

Análisis del Sistema

Requerimientos Funcionales

- El sistema permite a los clientes consultar la disponibilidad de calzado mediante escaneo de código de barras.
- El sistema muestra información detallada del producto: nombre, marca, precio, stock, descripción e imagen.
- Permite a los administradores:
 - Iniciar sesión en un panel de control.
 - Registrar nuevos productos con imagen, descripción y datos.
 - Editar productos existentes.
 - Eliminar productos de la base de datos.
 - Consultar una tabla completa de productos existentes.
- El sistema muestra una interfaz de inicio de sesión y una opción de registro de usuarios.
- Se implementa una navegación estructurada con secciones: Inicio, Catálogo, Contacto, Iniciar Sesión.

Requerimientos No Funcionales

- La interfaz debe ser intuitiva, responsiva y visualmente agradable.
- El sistema debe garantizar que los datos del producto se guarden y muestren correctamente.
- Las imágenes deben estar correctamente vinculadas y visibles desde la tabla de productos.
- El acceso al panel de administración debe requerir autenticación.

Módulos del Sistema

Módulo	Funcionalidad Principal
Inicio	Muestra un mensaje de bienvenida con la descripción del sistema.
Catálogo	(Próximo módulo o vista en desarrollo) Permitirá al cliente ver productos sin iniciar sesión.
Login / Registro	Permite el acceso al sistema como administrador.
Panel de Control	Vista del administrador con opciones para gestionar productos y usuarios.
Gestión de Productos	Listado CRUD de productos con tabla y botones de acción.
Gestión de Usuarios	(Vista futura o simulada) para administración de cuentas.

Modelo de Datos

A continuación, una posible estructura de la tabla productos:

Campo	Tipo de dato	Descripción
id	INT AUTO_INCREMENT	Identificador único del producto
codigo_barras	VARCHAR	Código escaneable del producto
nombre	VARCHAR	Nombre del modelo
marca	VARCHAR	Marca del calzado
precio	DECIMAL	Precio en moneda local
stock	INT	Existencias disponibles
descripcion	TEXT	Descripción breve del producto
imagen	VARCHAR	Ruta del archivo de imagen
fecha_creacion	DATE	Fecha en que fue agregado al sistema

Herramientas y Tecnologías Utilizadas

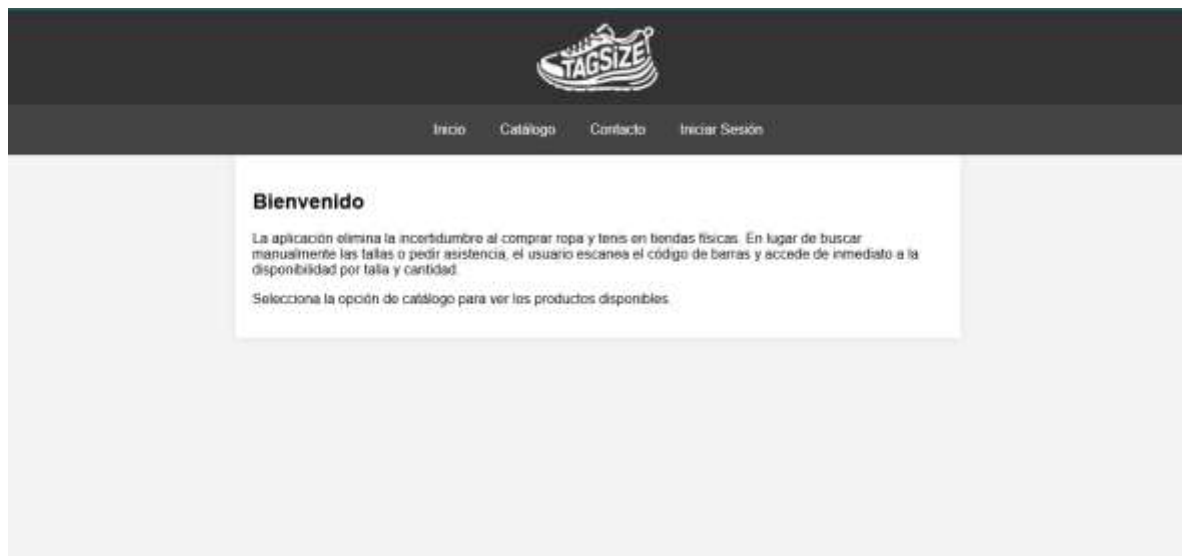
- **Lenguajes:** HTML5, CSS3, PHP, JavaScript
- **Base de datos:** MySQL o almacenamiento en JSON/PHP local (según implementación)
- **Entorno:** Localhost (XAMPP o similar)
- **IDE:** Visual Studio Code
- **Recursos Adicionales:** Bootstrap (para estilo y diseño responsive), íconos, Google Fonts, etc.

Diseño del Sistema e Interfaz Gráfica

Esta sección describe el diseño visual y funcional de las principales vistas que componen la aplicación. Se incluyen las interfaces más representativas del sistema tanto para el usuario como para el administrador.

Vista Principal (Inicio)

- Sección accesible desde home_page.html.
- Contiene un mensaje de bienvenida y explicación del objetivo de la app.
- Incluye barra de navegación con las opciones: Inicio, Catálogo, Contacto e Iniciar Sesión.
- Estética sobria con fondo gris oscuro en la barra superior y blanco en el contenido.



Dentro del código en la parte de “head” definimos que escale a las pantallas, de igual manera definimos un título para la página, ponemos el logo para que aparezca en la parte superior y de igual manera mandamos llamar nuestro “styles.css”.

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>TagSize - Inicio</title>
  <link rel="icon" type="image/png" href="../../public/img/logo-removebg-preview.png">
  <link rel="stylesheet" href="../../css/styles.css">
</head>
```

En la parte del “header” le ponemos un link a nuestro logo que se muestra en la parte superior para que al darle click nos regrese a la página de inicio, también ponemos nuestra imagen en la parte superior.

```
<header>
  <a href="../../html/home_page.html">
    
  </a>
</header>
```

Estas partes están dentro de un “container” que en la etiqueta “h2” contiene la palabra bienvenidos y la parte que está dentro de “p” es un párrafo con una breve descripción del servicio.

```
<div class="container">
  <h2>Bienvenido</h2>
  <p>Esta aplicación elimina la incertidumbre al comprar ropa y teñis en tiendas físicas. En lugar de buscar manualmente las tallas o pedir asistencia, el usuario esc
  </p>
  <p>selecciona la opción de catálogo para ver los productos disponibles.</p>
</div>
```

Después se manda llamar a “menú.js” que como su nombre lo indica funciona de un ligero menú, que utilizamos para desplazarnos entre vistas.

```
<script src="../../js/menu.js"></script>
```


Menu.js

Esto es lo que contiene el “menú.js” en donde se agrega un evento listener para el evento “DOMContentLoaded” es decir que para cuando todo el contenido del HTML ha sido cargado, después se declara una variable “menú” que contiene una plantilla de cadena de texto, el contenido de menú es crear enlaces para cuando le dan click a diferentes secciones de la página.

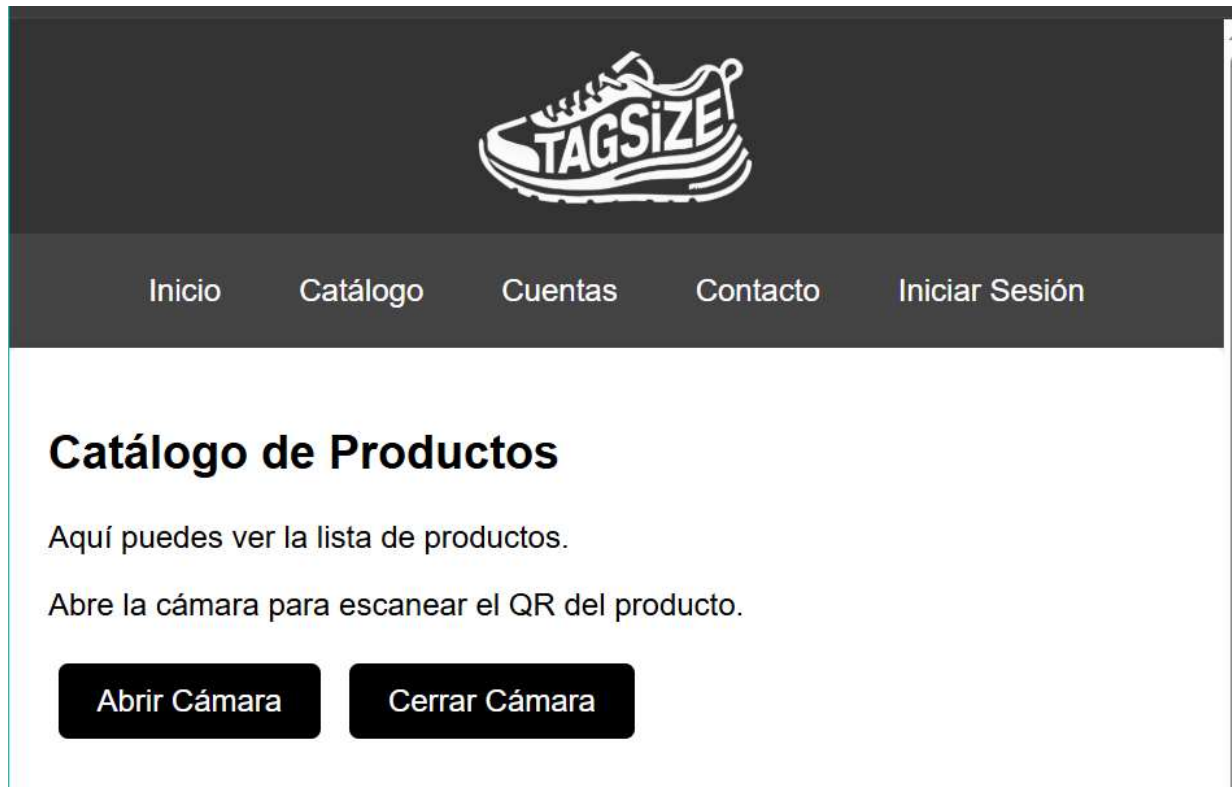
Al final del código lo que hace es buscar un elemento con el id “menú-container” en el HTML e inserta este código en el menú del HTML.

```
document.addEventListener("DOMContentLoaded", function() {  
  const menu = `  
    <nav>  
      <a href="./home_page.html" onclick="mostrarSeccion('inicio')">Inicio</a>  
      <a href="./catalogo.html" onclick="mostrarSeccion('catalogo')">Catálogo</a>  
      <a href="./cuentas.html" onclick="mostrarSeccion('cuentas')">Cuentas</a>  
      <a href="./contacto.html" onclick="mostrarSeccion('contacto')">Contacto</a>  
      <a href="./welcome.html">Iniciar Sesión</a>  
    </nav>  
  `;  
  document.getElementById("menu-container").innerHTML = menu;  
});
```

En este caso así queda nuestro menú que referencia a diferentes HTML.



Página catalogo:



Dentro del código tenemos este primer parte en el “head” en donde es lo mismo que el archivo que explicamos con anterioridad, todos los archivos como lo es contacto tienen la misma información en la parte del “head”.

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>TagSize - Catálogo</title>
  <link rel="icon" type="image/png" href="../../public/img/logo-removebg-preview.png">
  <link rel="stylesheet" href="../../css/styles.css">
</head>
```

De igual manera en el logo pusimos un link que nos reenvió a la pagina de inicio y de igual manera se pone el logo.

```
<header>
  <a href="../../html/home_page.html">
    
</header>
```

En la parte del “div” tenemos un “container” en donde primero en la etiqueta “h2” tenemos un título, después tenemos unos cuantos párrafos que los tenemos con la etiqueta “p”, de igual forma

ya por último tenemos dos botones uno con la función de abrir la cámara y otro con la función de cerrar la cámara.

```
<div class="container">
  <h2>Catálogo de Productos</h2>
  <p>Aquí puedes ver la lista de productos.</p>
  <p>Abre la cámara para escanear el QR del producto.</p>
  <button onclick="abrirCamara()">Abrir Cámara</button>
  <button onclick="cerrarCamara()">Cerrar Cámara</button>
  <video id="video" autoplay></video>
</div>
```

De igual forma como en el HTML anterior mandamos llamar al "menú.js" y en este caso un "script.js" para realizar las funciones de abrir la cámara.

```
<script src="..js/menu.js"></script>
<script src="..js/script.js"></script>
```

Script.js

Primero que nada, generamos una variable “videostream” que almacenara la transmisión de video, esta variable después nos ayuda a detener la transmisión.

Después tenemos la función “abrirCamara ()”, que realiza esta función:

- Se solicita el permiso al usuario para poder acceder a la cámara y se obtiene un stream de video.

Si el acceso es exitoso:

- `let video = document.getElementById("video");`
 - Se obtiene el elemento “video” de la pagina donde se mostrará el video en vivo.
- `video.srcObject = stream;`
 - Asigna el stream de la cámara como fuente del video permitiendo que este se visualice.
- `videoStream = stream;`
 - Guarda el stream en la variable global “videoStream” para después detenerla.
- Si ocurre un error
 - Muestra un mensaje en la consola y de igual manera una alerta de que no se pudo acceder a la cámara.

```
let videoStream;

function abrirCamara() {
  navigator.mediaDevices.getUserMedia({ video: true })
    .then(function(stream) {
      let video = document.getElementById("video");
      video.srcObject = stream;
      videoStream = stream;
    })
    .catch(function(error) {
      console.error("Error al acceder a la cámara: ", error);
      alert("No se pudo acceder a la cámara.");
    });
}
```

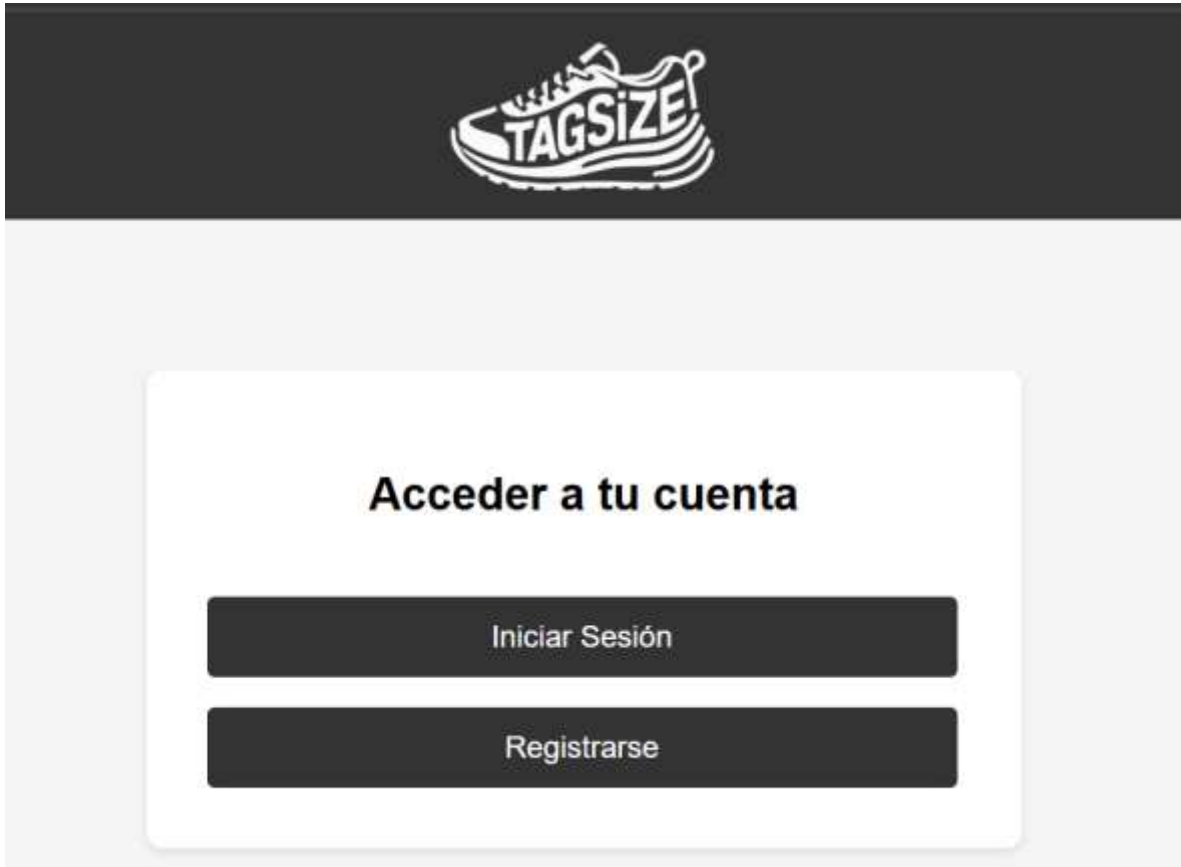
Ahora tenemos la función de “cerrarCamara ()”:

- Primero verificamos si la cámara esta activa.

- Si esta activa cierra la transmisión.
- Detiene todas las pistas de video.
- Después limpia la referencia del video.
- Ya por último hace que la variable "videoStream" =null para liberar recursos.

```
function cerrarCamara() {  
  if (videoStream) {  
    let tracks = videoStream.getTracks();  
    tracks.forEach(track => track.stop());  
    document.getElementById("video").srcObject = null;  
    videoStream = null;  
  }  
}
```

Página de acceso:



En esta nueva vista si cambia un poco el “head”, de igual manera le damos un titulo a nuestra pestaña, después se coloca lo que es el icono, de igual manera mandamos llamar “styles_welcome.css” si logramos observar para esta pestaña si cambia lo que es nuestro CSS debido a que en este se tiene un nuevo formato.

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8" />
    <title>Acceso - TagSize</title>
    <link rel="icon" type="image/jpeg" href="../../../public/img/logo-removebg-preview.png">
    <link rel="stylesheet" href="../../../css/styles_welcome.css">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
```

En la parte del “header” de igual manera se coloca lo que es nuestro logo y asi mismo se le coloca un link por si se da click sobre este redireccionándonos a la página de bienvenida.

```
<header>
  <a href="../../../html/home_page.html">
    
  </a>
</header>
```

Ahora tenemos un “login-container” en donde tenemos la etiqueta “h2” donde tenemos un título, después tenemos 2 botones los cuales nos redirigen a diferentes páginas, la primera de ellas nos reenvía a la parte de iniciar sesión y en el segundo botón este nos envía a registrarnos.

```
<div class="login-container">
  <h2>Acceder a tu cuenta</h2>
  <a href="login.html" class="btn-auth">Iniciar Sesión</a>
  <a href="signup.html" class="btn-auth">Registrarse</a>
</div>
```

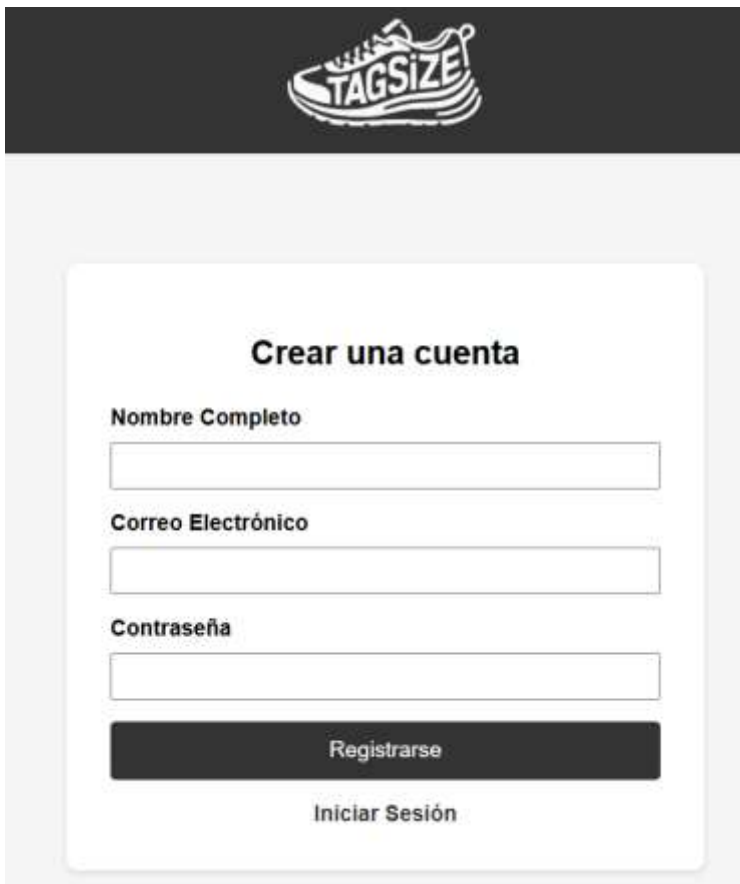
Pagina Registro:

Como ya se mencionó antes al darle click al botón “Registrarse” nos reenvía a una nueva página.



The screenshot shows the login page for TAGSIZE. At the top is a dark header with the TAGSIZE logo, which is a sneaker with the brand name on it. Below the header is a light gray background. In the center is a white rounded rectangle containing the text "Acceder a tu cuenta". Below this text are two dark buttons: "Iniciar Sesión" and "Registrarse".

En nuestra vista signup.html, tenemos un formulario para registrar solo el personal de la tienda, los clientes no será necesario que se registren en nuestra plataforma.



The screenshot shows the signup page for TAGSIZE. At the top is a dark header with the TAGSIZE logo. Below the header is a light gray background. In the center is a white rounded rectangle containing the text "Crear una cuenta". Below this text are three input fields: "Nombre Completo", "Correo Electrónico", and "Contraseña". Below the input fields is a dark button labeled "Registrarse". At the bottom of the white rectangle is a link labeled "Iniciar Sesión".

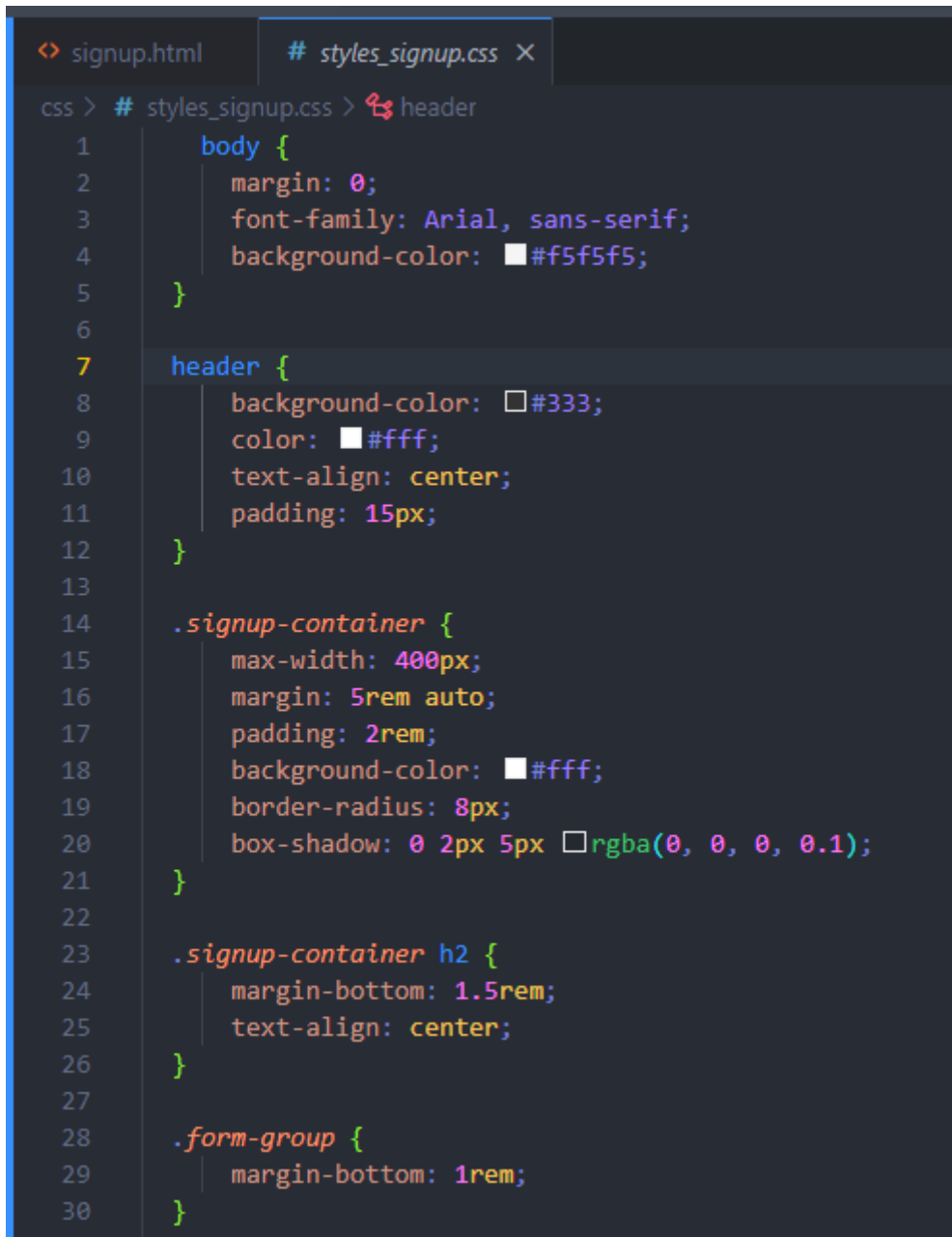
Para la parte inicial de nuestra vista “signup”, pusimos el título de la pestaña como “Registro – TagSize”.

Importamos el logo de nuestra empresa y los estilos que conlleva nuestro signup.html:

Signup.css:

Para el body, usamos un fondo blanco y la fuente Arial

Para el encabezado en donde se encuentra la barra de navegación, configuramos color negro, y el texto centrado, que, en nuestro caso, centrará la imagen

A screenshot of a code editor with two tabs: 'signup.html' and '# styles_signup.css'. The active tab is '# styles_signup.css', which contains CSS code for styling a signup page. The code is as follows:

```
css > # styles_signup.css > header
1   body {
2     margin: 0;
3     font-family: Arial, sans-serif;
4     background-color: #f5f5f5;
5   }
6
7   header {
8     background-color: #333;
9     color: #fff;
10    text-align: center;
11    padding: 15px;
12  }
13
14  .signup-container {
15    max-width: 400px;
16    margin: 5rem auto;
17    padding: 2rem;
18    background-color: #fff;
19    border-radius: 8px;
20    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
21  }
22
23  .signup-container h2 {
24    margin-bottom: 1.5rem;
25    text-align: center;
26  }
27
28  .form-group {
29    margin-bottom: 1rem;
30  }
```

```

<? signup.html X
html > <? signup.html
1  <!DOCTYPE html>
2  <html Lang="es">
3  <head>
4      <meta charset="UTF-8">
5      <title>Registro - TagSize</title>
6      <link rel="icon" type="image/jpeg" href="../public/img/logo-removebg-preview.png">
7      <link rel="stylesheet" href="../css/styles_signup.css">
8  </head>
9  <body>
10     <header>
11         <a href="home_page.html">
12             
13         </a>
14     </header>

```

En nuestra vista signup.html, ponemos un div que engloba todo el formulario, en donde la acción del formulario apunta a nuestro controlador signup, que es un archivo php, en donde valida que los inputs no estén vacíos y tengan el formato adecuado, utilizamos el método HTTP post, debido a que se creara un nuevo usuario en la base de datos

```

<div class="signup-container">
    <h2>Crear una cuenta</h2>
    <form action="http://localhost/TAGSIZE/http/Controllers/signup.php" method="POST">
        <div class="form-group">
            <label for="nombre">Nombre Completo</label>
            <input type="text" id="nombre" name="nombre">
        </div>
        <div class="form-group">
            <label for="email">Correo Electrónico</label>
            <input type="email" id="email" name="email">
        </div>
        <div class="form-group">
            <label for="password">Contraseña</label>
            <input type="password" id="password" name="password">
        </div>
        <button type="submit" class="btn-submit">Registrarse</button>
    </form>
    <a href="login.html" class="btn-auth">Iniciar Sesión</a>
</div>
</body>
</html>

```

Signup.php

Iniciamos una sesión en nuestra base de datos

Revisa que el método POST se haya solicitado correctamente, y que los inputs nombre, email, password no estén vacíos y que si se haya ingresado algo

Si todo está correct, realiza la consulta en la base de datos para verificar que el registro no esté ya en la base de datos y sea un registro nuevo. Si ese registro no existe en la base de datos, realiza el insert a la tabla usuarios

```
<> signup.html M  signup.php X
http > Controllers > signup.php
1  <?php
2  session_start();
3  include "db_connection.php"; // Conexión a la base de datos
4
5  if ($_SERVER["REQUEST_METHOD"] === "POST") {
6      $nombre = trim($_POST["nombre"]);
7      $email = trim($_POST["email"]);
8      $password = trim($_POST["password"]);
9
10     // Validación de campos vacíos
11     $errores = [];
12     if (empty($nombre)) {
13         $errores[] = "El nombre es obligatorio.";
14     }
15     if (empty($email) || !filter_var($email, FILTER_VALIDATE_EMAIL)) {
16         $errores[] = "El correo electrónico no es válido.";
17     }
18     if (empty($password) || strlen($password) < 6) {
19         $errores[] = "La contraseña debe tener al menos 6 caracteres.";
20     }
21     if (!empty($errores)) {
22         $_SESSION["errores"] = $errores;
23         header("Location: ../Views/error.php");
24         exit();
25     }
26
27     // Verificar si el correo ya existe en la base de datos
28     $stmt = $conn->prepare("SELECT id FROM usuarios WHERE email = ?");
29     $stmt->bind_param("s", $email);
30     $stmt->execute();
31     $resultado = $stmt->get_result();
32
```

```
if ($resultado->num_rows > 0) {
    $_SESSION["errores"] = ["El correo ya está registrado."];
    header("Location: ../Views/error.php");
    exit();
}

// Insertar nuevo usuario sin encriptar la contraseña
$tipo_usuario = 'E'; // Tipo de usuario por defecto

$stmt = $conn->prepare("INSERT INTO usuarios (nombre, email, password, tipo_usuario) VALUES (?, ?, ?, ?)");
$stmt->bind_param("ssss", $nombre, $email, $password, $tipo_usuario);

if ($stmt->execute()) {
    $_SESSION["mensaje_exito"] = "Usuario registrado correctamente.";
    header("Location: ../Views/success.php");
    exit();
} else {
    $_SESSION["errores"] = ["Error al registrar el usuario."];
    header("Location: ../Views/error.php");
    exit();
}
}

$conn->close();
?>
```

Página Iniciar Sesión:

Esta es nuestra página para iniciar sesión, a donde direcciona la página de acceso e igualmente la página de Registrarse después de crear un usuario.

Regístrate aquí'." data-bbox="135 165 859 698"/>

Tagsize

Iniciar Sesión

Correo electrónico

Contraseña

Entrar

¿No tienes cuenta? [Regístrate aquí](#)

Verificamos si el formulario se envió usando el método POST. Esto es una buena práctica para asegurarnos de que los datos del formulario se procesen solo cuando se envían correctamente.

Con el trim() eliminamos los espacios en blanco al inicio y final de los valores enviados desde el formulario y con \$_POST["email"] y \$_POST["password"] recuperamos los valores enviados desde el formulario de inicio de sesión.

```

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $email = trim($_POST["email"]);
    $password = trim($_POST["password"]);

    if (empty($email) || empty($password)) {
        echo "✗ Todos los campos son obligatorios.";
        exit;
    }
}

```

Verificamos si el correo electrónico o la contraseña están vacíos. Si alguno de los campos está vacío, muestra un mensaje de error y detiene la ejecución del script con exit.

```

if (empty($email) || empty($password)) {
    echo "✗ Todos los campos son obligatorios.";
    exit;
}

```

Usamos \$sql para definir una consulta SQL para buscar un usuario en la tabla usuarios usando su correo electrónico y con \$resultado = \$stmt->get_result() obtenemos el resultado de la consulta.

También preparamos la consulta para evitar inyecciones SQL con \$stmt->prepare(\$sql):

```

$sql = "SELECT id_usuarios, nombre_usuario, email_usuario, password_usuario,
tipo_usuario FROM usuarios WHERE email_usuario = ?";
$stmt = $conn->prepare($sql);
$stmt->bind_param("s", $email);
$stmt->execute();
$resultado = $stmt->get_result();

```

Verificamos si se encontró exactamente un usuario con el correo electrónico proporcionado y convertimos el resultado en un array asociativo para acceder a los datos del usuario

```

if ($resultado->num_rows == 1) {
    $usuario = $resultado->fetch_assoc();
}

```

Comparamos la contraseña proporcionada por el usuario con la contraseña almacenada en la base de datos, aunque no es lo más ideal, ya que de preferencia deberíamos de hacerlo mediante password_hash() y verificar con password_verify().

Una vez que verificamos la contraseña, almacenamos los datos del usuario en la sesión para usarlos en las demás páginas y redirigimos al usuario al dashboard después de un inicio de sesión exitoso.

```
if ($password == $usuario["password_usuario"]) {  
    // Iniciar sesión y almacenar datos del usuario  
    $_SESSION["usuario_id"] = $usuario["id_usuarios"];  
    $_SESSION["usuario_nombre"] = $usuario["nombre_usuario"];  
    $_SESSION["usuario_tipo"] = $usuario["tipo_usuario"];  
  
    // Redirigir al dashboard  
    header("Location: ../../html/dashboard.html");  
    exit;  
} else {  
    echo "❌ Contraseña incorrecta.";  
}  
} else {  
    echo "❌ Usuario no encontrado.";  
}
```

❌ Usuario no encontrado.

❌ Contraseña incorrecta.

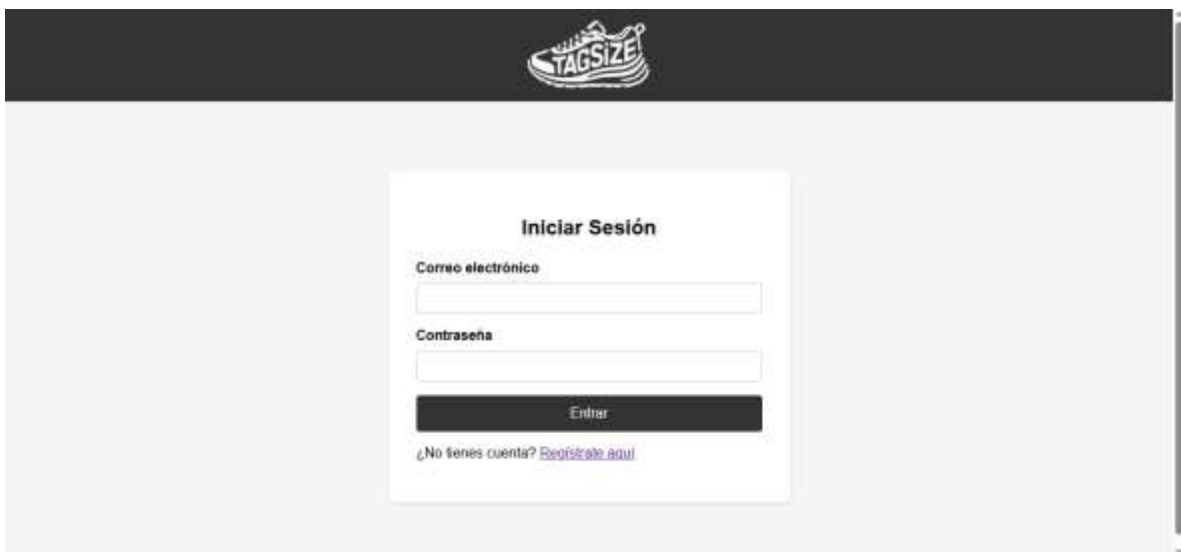
Estos son algunos de los usuarios que tenemos dados de alta, y con cualquiera de ellos podemos acceder correctamente

123 • `select * from usuarios;`

	id_usuarios	nombre_usuario	email_usuario	password_usuario	fecha_registro_usuario	tipo_usuario
▶	1	Mario Sierra	mario@example.com	pass1234	2025-03-05	A
	2	Luis Pérez	luis@example.com	luispass	2025-03-05	E
	3	Ana Gómez	ana@example.com	anasegura	2025-03-05	A
	4	Carlos Ruiz	carlos@example.com	carlospass	2025-03-05	E
	5	Elena Torres	elena@example.com	torrespas	2025-03-05	A
	6	MarioD	smariodael@gmail.com	17101710	NULL	E
*	NULL	NULL	NULL	NULL	NULL	NULL

Vista de Acceso (Login)

- Presenta un contenedor central con el título “**Acceder a tu cuenta**”.
- Dos botones:
 - Iniciar Sesión: Redirige al formulario para iniciar sesión.
 - Registrarse: Permite crear una cuenta nueva.
- Interfaz simple, centrada, con fondo claro y botones oscuros.





The screenshot shows a registration form titled "Crear una cuenta" (Create an account) on the TAGSIZE website. The form is centered on a light gray background. It contains three input fields: "Nombre Completo" (Full Name), "Correo Electrónico" (Email), and "Contraseña" (Password). Below these fields is a dark gray button labeled "Registrarse" (Register). At the bottom of the form, there is a link that says "Iniciar Sesión" (Log In).

Panel de Administración

- Muestra el mensaje: "Inicio de sesión como: [Nombre del usuario]" y un botón para cerrar sesión.
- Opciones:
 - Gestión de Productos: Accede al CRUD completo.
 - Gestión de Usuarios: Vista pendiente o en desarrollo.
- Diseño limpio y directo, adaptado a tareas administrativas.



The screenshot shows the "Panel de Control" (Control Panel) in the administration area. The top header is dark gray and contains the text "Bienvenido" (Welcome) on the left, "Inicio de sesión como: Mario Sierra" (Logged in as: Mario Sierra) in the center, and a red button labeled "Cerrar sesión" (Log out) on the right. The main content area is light gray and features a white box titled "Panel de Control". Inside this box are two dark gray buttons: "Gestión de Productos" (Product Management) and "Gestión de Usuarios" (User Management).

Gestión de Productos (CRUD)

Presenta una **tabla de productos** con columnas:

- ID, Código de Barras, Nombre, Marca, Precio, Stock, Descripción, Imagen, Fecha de creación y Acciones.

Botones de acción:

- Editar: Permite modificar datos del producto.
- Eliminar: Borra el producto de la base de datos.

Botón superior: Agregar Producto (formulario emergente o vista alterna).

Estilo visual:

- Botones verdes (Agregar), azules (Editar) y rojos (Eliminar).
- Visualización clara de imágenes miniatura.

Lista de Productos

Lista de Productos

[Agregar Producto](#) [Regresar](#)

ID	Código de Barras	Nombre	Marca	Precio	Stock	Descripción	Imagen	Fecha de Creación	Acciones
6	2006	Nike Blazer	Nike	\$1,200.00	20	Tennis nike de bota		2025-05-22	Editar Eliminar
7	2007	addias campus	Adidas	\$1,700.00	10	Tenis de gamuza		2025-05-22	Editar Eliminar
8	2008	nike air force 1	nike	\$2,000.00	23	nike air force 1		2025-05-22	Editar Eliminar

```
<table>
<thead>
<tr>
<th>ID</th>
<th>Código de Barras</th>
<th>Nombre</th>
<th>Marca</th>
<th>Precio</th>
<th>Stock</th>
<th>Descripción</th>
<th>Imagen</th>
<th>Fecha de Creación</th>
<th>Acciones</th>
</tr>
</thead>
<tbody>
<?php while ($row = $result->fetch_assoc()): ?>
<tr>
<td><?php echo $row['id_producto']; ?></td>
<td><?php echo $row['codigo_barras']; ?></td>
<td><?php htmlspecialchars($row['nombre_producto']); ?></td>
<td><?php htmlspecialchars($row['marca_producto']); ?></td>
<td><?php number_format($row['precio_producto'], 2) ?></td>
<td><?php echo $row['stock_del_producto']; ?></td>
<td><?php htmlspecialchars($row['descripcion_producto']); ?></td>
<td>
<img alt="Imagen de producto" data-bbox="605 218 638 235" />
<img alt="Imagen de producto" data-bbox="601 271 641 286" />
<img alt="Imagen de producto" data-bbox="603 312 639 329" />
</td>
<td><?php echo $row['fecha_creacion_producto']; ?></td>
<td>
<button class="btn-editar" onclick="abrirModalEditar(
<?php echo $row['id_producto']; ?>,
'<?php htmlspecialchars($row['codigo_barras'], ENT_QUOTES) ?>',
'<?php htmlspecialchars($row['nombre_producto'], ENT_QUOTES) ?>',
'<?php htmlspecialchars($row['marca_producto'], ENT_QUOTES) ?>',
'<?php echo $row['precio_producto']; ?>',
'<?php echo $row['stock_del_producto']; ?>',
'<?php htmlspecialchars($row['descripcion_producto'], ENT_QUOTES) ?>'
)>Editar</button>

<form action="eliminar_producto.php" method="post" onsubmit="return confirm('¿Estás seguro de que deseas eliminar este producto?');" style="display:inline;"
<input type="hidden" name="id" value="<?php echo $row['id_producto']; ?>" />
<button type="submit" class="btn-eliminar">Eliminar</button>
</form>
</td>
</tr>
</tbody>
</table>
```

Muestra una tabla de productos en una página web. Cada fila de la tabla representa un producto almacenado en una base de datos.

Agregar Nuevo Producto

Agregar Nuevo Producto

Seleccionar Imagen

Ningún archivo seleccionado

Guardar

```

<!-- MODAL AGREGAR -->
<div id="modalAgregar" class="modal">
  <div class="modal-contenido">
    <span class="cerrar" onclick="cerrarModal()">&times;</span>
    <h2>Agregar Nuevo Producto</h2>
    <form action="agregar_producto.php" method="post" enctype="multipart/form-data">
      <input type="text" name="codigo_barras" placeholder="Código de barras" required>
      <input type="text" name="nombre_producto" placeholder="Nombre" required>
      <input type="text" name="marca_producto" placeholder="Marca" required>
      <input type="number" step="0.01" name="precio_producto" placeholder="Precio" required>
      <input type="number" name="stock_del_producto" placeholder="Stock" required>
      <textarea name="descripcion_producto" placeholder="Descripción" required></textarea>

      <!-- Personalización input file -->
      <div class="file-input-wrapper">
        <label for="imagen_producto" class="file-label">Seleccionar Imagen</label>
        <input type="file" name="imagen_producto" id="imagen_producto" accept="image/*" required>
        <span class="file-name" id="file-name">Ningún archivo seleccionado</span>
      </div>

      <button type="submit">Guardar</button>
    </form>
  </div>
</div>

```

Este modal sirve para:

- Mostrar un formulario para ingresar un nuevo producto.
- Incluir la subida de una imagen.
- Enviarlos a `agregar_producto.php` para su procesamiento.

Editar Productos

Editar Producto

2006

Nike Blazer

Nike

1200

20

Tennis nike de bota.

Seleccionar Imagen (opcional)

Ningún archivo seleccionado

Guardar Cambios

```
<!-- MODAL EDITAR -->
<div id="modalEditar" class="modal">
  <div class="modal-contenido">
    <span class="cerrar" onclick="cerrarModalEditar()">&times;</span>
    <h2>Editar Producto</h2>
    <form action="editar_producto.php" method="post" enctype="multipart/form-data">
      <input type="hidden" name="id_productos" id="editar-id">
      <input type="text" name="codigo_barras" id="editar-codigo" placeholder="Código de barras" required>
      <input type="text" name="nombre_producto" id="editar-nombre" placeholder="Nombre" required>
      <input type="text" name="marca_producto" id="editar-marca" placeholder="Marca" required>
      <input type="number" step="0.01" name="precio_producto" id="editar-precio" placeholder="Precio" required>
      <input type="number" name="stock_del_producto" id="editar-stock" placeholder="Stock" required>
      <textarea name="descripcion_producto" id="editar-descripcion" placeholder="Descripción" required></textarea>

      <!-- Personalización input file -->
      <div class="file-input-wrapper">
        <label for="editar-imagen" class="file-label">Seleccionar Imagen (opcional)</label>
        <input type="file" name="imagen_producto" id="editar-imagen" accept="image/jpeg,image/png,image/webp">
        <span class="file-name" id="file-name-edit">Ningún archivo seleccionado</span>
      </div>

      <button type="submit">Guardar Cambios</button>
    </form>
  </div>
</div>
```

Este modal:

- Permite editar los datos de un producto existente.
- Incluye la opción de cambiar la imagen.
- Usa JavaScript para rellenar el formulario antes de mostrarlo.
- Envía los cambios a editar_producto.php.

Estilo General

- Predominio de colores grises oscuros para navegación y botones.
- Textos contrastantes en blanco y negro.
- Interfaz responsive y centralizada, adecuada para pantallas medianas y grandes.
- Uso de íconos para mejorar la usabilidad y estética (por ejemplo, en el panel de control).

Desarrollo e Implementación del Sistema

Esta sección describe el funcionamiento interno del sistema, la lógica de programación, el flujo de datos y cómo se integran los distintos módulos utilizando PHP, HTML, CSS y posiblemente JavaScript.

Estructura General de Archivos

El proyecto está organizado de forma modular, con carpetas principales como:

```
/tagsize/  
|  
├─ /html/           ← Archivos HTML como `home_page.html`  
├─ /php/            ← Archivos PHP para lógica, conexión, CRUD  
├─ /css/            ← Hojas de estilo personalizadas  
├─ /img/            ← Imágenes de productos y recursos gráficos  
└─ /js/             ← Archivos JavaScript si aplica
```

Conexión a Base de Datos

Se realiza con PHP, generalmente en un archivo dedicado

```
db_connection.php X
http > Controllers > db_connection.php
1  <?php
2  $host = "localhost"; // Servidor de MySQL
3  $user = "root";      // Usuario de MySQL (cambia si usas otro)
4  $password = "root";  // Contraseña de MySQL (déjala vacía si no configuraste una)
5  $dbname = "tennis2"; // Nombre de la base de datos
6
7  // Crear la conexión
8  $conn = new mysqli($host, $user, $password, $dbname);
9
10 // Verificar si la conexión fue exitosa
11 if ($conn->connect_error) {
12     die("Error de conexión: " . $conn->connect_error);
13 }
14 ?>
```

- Se utiliza MySQL como sistema gestor.
- Las consultas SQL están distribuidas en funciones para insertar, actualizar, listar y eliminar productos.

Gestión de Productos (CRUD)

Agregar Producto: Formulario que envía los datos al servidor con POST, almacenando en la tabla productos.

```
agregar_producto.php M X
http > Controllers > agregar_producto.php
1  <?php
2  require_once 'db_connection.php';
3
4  // Recoge los datos del formulario
5  $codigo_barras = $_POST['codigo_barras'] ?? '';
6  $nombre = $_POST['nombre_producto'] ?? '';
7  $marca = $_POST['marca_producto'] ?? '';
8  $precio = $_POST['precio_producto'] ?? 0;
9  $stock = $_POST['stock_del_producto'] ?? 0;
10 $descripcion = $_POST['descripcion_producto'] ?? '';
11 $fecha = date('Y-m-d H:i:s');
12
13 // Validaciones básicas
14 if (empty($codigo_barras) || empty($nombre) || empty($marca) || empty($precio) || empty($stock) || empty($descripcion)) {
15     die("X Todos los campos son obligatorios.");
16 }
17
18 // Validar subida de imagen
19 if ($_FILES['imagen_producto']['error'] !== UPLOAD_ERR_OK) {
20     die("X Error al subir la imagen. Código: " . $_FILES['imagen_producto']['error']);
21 }
```

```

44 // Guardar imagen con nombre unico
45 $nombre_original = basename($_FILES['imagen_producto']['name']);
46 $nombre_unico = uniqid() . "_" . preg_replace("/[^\a-zA-Z0-9.\-_]/", "_", $nombre_original);
47 $ruta_imagen = $directorio_subida . $nombre_unico;
48
49 if (!move_uploaded_file($imagen_temp, $ruta_imagen)) {
50     die("❌ Error al mover la imagen al directorio destino.");
51 }
52
53 // Insertar en la base de datos
54 $sql = "INSERT INTO productos (
55     codigo_barras, nombre_producto, marca_producto,
56     precio_producto, stock_del_producto, descripcion_producto,
57     imagen_producto, fecha_creacion_producto
58 ) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
59
60 $stmt = $conn->prepare($sql);
61 $stmt->bind_param("ssdssss", $codigo_barras, $nombre, $marca, $precio, $stock, $descripcion, $ruta_imagen, $fecha);
62
63 if ($stmt->execute()) {
64     header("Location: ver_productos.php?mensaje=exito");
65     exit(); // <- Muy importante para detener el script.
66 } else {
67     header("Location: ver_productos.php?mensaje=error");
68     exit();
69 }
70
71 $stmt->close();
72 $conn->close();
73 ?>

```

Listar Productos: Se realiza una consulta SQL `SELECT * FROM productos`, cuyos resultados se imprimen en una tabla HTML.

```

ver_productos.php X
http > Controllers > ver_productos.php
1  <?php
2  require_once 'db_connection.php';
3
4  $query = "SELECT * FROM productos";
5  $result = $conn->query($query);
6  ?>
7

```



```

74 <tbody>
75 <?php while ($row = $result->fetch_assoc()): ?>
76 <tr>
77 <td><?=$row['id_productos'] ?></td>
78 <td><?=$row['codigo_barras'] ?></td>
79 <td><?=$row['nombre_producto'] ?></td>
80 <td><?=$row['marca_producto'] ?></td>
81 <td><?=$row['precio_producto'] ?></td>
82 <td><?=$row['stock_producto'] ?></td>
83 <td><?=$row['descripcion_producto'] ?></td>
84 <td>
85 <?php if (empty($row['imagen_producto'])): ?>
86 
87 <?php else: ?>
88 Sin imagen
89 <?php endif: ?>
90 </td>
91 <td><?=$row['fecha_creacion_producto'] ?></td>
92 <td>
93 <button class="btn-editar" onclick="abrirModalEditar(
94 <?=$row['id_productos'] ?>,
95 '<?=$row['codigo_barras'] ?>',
96 '<?=$row['nombre_producto'] ?>',
97 '<?=$row['marca_producto'] ?>',
98 <?=$row['precio_producto'] ?>,
99 <?=$row['stock_producto'] ?>,
100 '<?=$row['descripcion_producto'] ?>'
101 )">Editar</button>

```

Editar Producto: Se recupera el id del producto, se precarga en un formulario editable, y se actualizan los valores con un UPDATE.

```

<td>
<button class="btn-editar" onclick="abrirModalEditar(
<?=$row['id_productos'] ?>,
'<?=$row['codigo_barras'] ?>',
'<?=$row['nombre_producto'] ?>',
'<?=$row['marca_producto'] ?>',
<?=$row['precio_producto'] ?>,
<?=$row['stock_producto'] ?>,
'<?=$row['descripcion_producto'] ?>'
)">Editar</button>

<form action="eliminar_producto.php" method="post" onsubmit="return confirm('¿Estás seguro de que desea
<input type="hidden" name="id" value="<?=$row['id_productos'] ?>")>
<button type="submit" class="btn-eliminar">Eliminar</button>
</form>

</td>

```

```

<script>
function abrirModalEditar(id, codigo, nombre, marca, precio, stock, descripcion) {
document.getElementById("editar-id").value = id;
document.getElementById("editar-codigo").value = codigo;
document.getElementById("editar-nombre").value = nombre;
document.getElementById("editar-marca").value = marca;
document.getElementById("editar-precio").value = precio;
document.getElementById("editar-stock").value = stock;
document.getElementById("editar-descripcion").value = descripcion;

document.getElementById("modalEditar").style.display = "block";
}

```

```

editar_producto.php X
http > Controllers > editar_producto.php
1  <?php
2  require_once 'db_connection.php';
3
4  // Verifica que todos los datos hayan sido enviados
5  if (
6      isset($_POST['id_productos'], $_POST['codigo_barras'], $_POST['nombre_producto'],
7          $_POST['marca_producto'], $_POST['precio_producto'],
8          $_POST['stock_del_producto'], $_POST['descripcion_producto'])
9  ) {
10     $id = $_POST['id_productos'];
11     $codigo = $_POST['codigo_barras'];
12     $nombre = $_POST['nombre_producto'];
13     $marca = $_POST['marca_producto'];
14     $precio = $_POST['precio_producto'];
15     $stock = $_POST['stock_del_producto'];
16     $descripcion = $_POST['descripcion_producto'];
17
18     $imagenNueva = false;
19     $ruta_imagen = '';
20
21     if ($imagenNueva) {
22         $sql = "UPDATE productos
23             SET codigo_barras = ?, nombre_producto = ?, marca_producto = ?,
24                 precio_producto = ?, stock_del_producto = ?, descripcion_producto = ?,
25                 imagen_producto = ?
26             WHERE id_productos = ?";
27         $stmt = $conn->prepare($sql);
28         $stmt->bind_param("ssssdissi", $codigo, $nombre, $marca, $precio, $stock, $descripcion, $ruta_imagen, $id);
29     } else {
30         $sql = "UPDATE productos
31             SET codigo_barras = ?, nombre_producto = ?, marca_producto = ?,
32                 precio_producto = ?, stock_del_producto = ?, descripcion_producto = ?
33             WHERE id_productos = ?";
34         $stmt = $conn->prepare($sql);
35         $stmt->bind_param("sssdiss", $codigo, $nombre, $marca, $precio, $stock, $descripcion, $id);
36     }
37
38     if ($stmt->execute()) {
39         header("Location: ver_productos.php?mensaje-editado");
40         exit();
41     } else {
42         echo "X Error al actualizar el producto: " . $stmt->error;
43     }
44
45     $stmt->close();
46 } else {
47     echo "X Faltan datos del formulario.";
48 }
49
50 $conn->close();
51 }

```

Eliminar Producto: Se ejecuta un DELETE FROM productos WHERE id = .

```

<form action="eliminar_producto.php" method="post" onsubmit="return confirm('¿Estás seguro de que deseas eli
    <input type="hidden" name="id" value="<?= $row['id_productos'] ?>">
    <button type="submit" class="btn-eliminar">Eliminar</button>
</form>

```

eliminar_producto.php ×

http > Controllers > eliminar_producto.php

```
1  <?php
2  require_once 'db_connection.php'; // Conecta con tu base de datos
3
4  if ($_SERVER['REQUEST_METHOD'] === 'POST') {
5      $id = $_POST['id'] ?? null;
6
7      if ($id === null || !is_numeric($id)) {
8          die("ID inválido.");
9      }
10
11     $sql = "DELETE FROM productos WHERE id_productos = ?";
12     $stmt = $conn->prepare($sql);
13     $stmt->bind_param("i", $id);
14
15     if ($stmt->execute()) {
16         header("Location: ver_productos.php?mensaje=eliminado");
17         exit();
18     } else {
19         echo "✗ Error al eliminar el producto.";
20     }
21
22     $stmt->close();
23     $conn->close();
24 } else {
25     die("Método no permitido.");
26 }
27
```

Validaciones y Seguridad

- Los formularios usan required en campos HTML para evitar datos vacíos.

```
<div class="login-container">
  <h2>Iniciar Sesión</h2>
  <form id="loginForm">
    <div class="form-group">
      <label for="email">Correo electrónico</label>
      <input type="email" id="email" name="email" required>
    </div>
    <div class="form-group">
      <label for="password">Contraseña</label>
      <input type="password" id="password" name="password" required>
    </div>
    <button type="submit" class="btn-submit">Entrar</button>
  </form>
  <p>¿No tienes cuenta? <a href="signup.html">Regístrate aquí</a></p>
  <div id="loginMessage" style="color: red; margin-top: 10px;"></div>
</div>
```

- En el lado del servidor, se recomienda usar:
 - mysqli_real_escape_string() o prepared statements para evitar inyecciones SQL.
 - Restricción de acceso al panel mediante sesiones (session_start() y \$_SESSION).

```
if ($resultado->num_rows == 1) {
    $usuario = $resultado->fetch_assoc();

    if ($password == $usuario["password_usuario"]) {
        $_SESSION["usuario_id"] = $usuario["id_usuarios"];
        $_SESSION["usuario_nombre"] = $usuario["nombre_usuario"];
        $_SESSION["usuario_tipo"] = $usuario["tipo_usuario"];

        //echo json_encode(["success" => true]);
        printf(json_encode(["success" => true]));
    } else {
        printf(json_encode(["success" => false, "message" => "✗ Contraseña incorrecta."]));
    }
} else {
    printf(json_encode(["success" => false, "message" => "✗ Usuario no encontrado."]));
}
```

Control de Sesiones

- Cuando un usuario inicia sesión correctamente:

```
session_start();
$_SESSION['usuario'] = $nombre_usuario;
```

- En páginas protegidas:

```
session_start();
if (!isset($_SESSION['usuario'])) {
    header("Location: login.php");
}
```

- Para cerrar sesión:

```
session_destroy();
header("Location: home_page.html");
```

Carga y Visualización de Imágenes

- Las imágenes se almacenan en una carpeta (/img/productos/) y se referencian por su ruta en la base de datos.
- Visualización en la tabla:

```
<td>
  <?php if (!empty($row['imagen_producto'])): ?>
    
  <?php else: ?>
    Sin imagen
  <?php endif; ?>
</td>
```

```
// Crear carpeta si no existe
$directorio_subida = "uploads/";
if (!is_dir($directorio_subida)) {
    mkdir($directorio_subida, 0755, true);
}

// Guardar imagen con nombre único
$nombre_original = basename($_FILES['imagen_producto']['name']);
$nombre_unico = uniqid() . "_" . preg_replace("/^[a-zA-Z0-9.\-_]/", "_", $nombre_original);
$ruta_imagen = $directorio_subida . $nombre_unico;

if (!move_uploaded_file($imagen_temp, $ruta_imagen)) {
    die("✗ Error al mover la imagen al directorio destino.");
}
```

```

// Validar si se cargó una imagen
if (isset($_FILES['imagen_producto']) && $_FILES['imagen_producto']['error'] === UPLOAD_ERR_OK) {
    $imagen_temp = $_FILES['imagen_producto']['tmp_name'];
    $tipo_imagen = mime_content_type($imagen_temp);
    $tipos_permitidos = ['image/jpeg', 'image/png', 'image/webp'];

    if (!in_array($tipo_imagen, $tipos_permitidos)) {
        die("✗ Solo se permiten imágenes JPG, PNG o WEBP.");
    }

    if ($_FILES['imagen_producto']['size'] > 2 * 1024 * 1024) {
        die("✗ La imagen excede los 2MB permitidos.");
    }

    $directorio = "uploads/";
    if (!is_dir($directorio)) {
        mkdir($directorio, 0755, true);
    }

    $nombre_original = basename($_FILES['imagen_producto']['name']);
    $nombre_unico = uniqid() . "_" . preg_replace("/[^a-zA-Z0-9.\-_]/", "_", $nombre_original);
    $ruta_imagen = $directorio . $nombre_unico;

    if (!move_uploaded_file($imagen_temp, $ruta_imagen)) {
        die("✗ Error al mover la imagen.");
    }

    $imagenNueva = true;
}

```

Conclusiones:

Integración de funcionalidades clave en un sitio web

Durante el desarrollo del proyecto, logramos integrar múltiples funcionalidades esenciales para la gestión de un catálogo de productos en línea. Implementamos un sistema de autenticación de usuarios, permitiendo el registro e inicio de sesión a través de una base de datos. Además, desarrollamos la capacidad de agregar, eliminar, visualizar y actualizar productos, asegurando que los usuarios pudieran gestionar la información de manera intuitiva y eficiente. Este proceso nos permitió comprender la importancia de estructurar adecuadamente un sitio web y mantener una arquitectura bien organizada para facilitar la escalabilidad y el mantenimiento del proyecto.

Conexión entre el entorno digital y físico mediante el escaneo de códigos de barras

Una de las características más innovadoras del proyecto fue la integración de un sistema de escaneo de códigos de barras a través de la cámara del dispositivo del usuario. Esto permitió la consulta en tiempo real de productos dentro de una tienda física, brindando información detallada sobre precio, stock, tallas y colores disponibles. La implementación de esta funcionalidad nos permitió aprender sobre la interacción entre tecnologías web y dispositivos de hardware, así como sobre el uso de bibliotecas para el procesamiento de imágenes y la detección de códigos de barras en un entorno de navegador.

Gestión de bases de datos y seguridad en la autenticación de usuarios

En este proyecto, trabajamos con bases de datos para almacenar información de usuarios y productos, lo que nos permitió reforzar nuestras habilidades en modelado de datos y consultas SQL. También comprendimos la importancia de implementar medidas de seguridad para proteger la información, como el uso de cifrado de contraseñas y la validación de datos en el proceso de autenticación. Este aprendizaje resultó clave para mejorar la robustez del sistema y garantizar que los datos de los usuarios y la información de los productos se manejen de forma segura y eficiente.