2024

ESTRUCTURA CURSO
DESARROLLO WEB
JUNIOR

Ing. Álvaro Stid Bolaños
Vásquez

[DESARROLLO WEB JUNIOR]

Objetivo del Curso

Al finalizar este curso, el estudiante será capaz de crear una aplicación web responsive orientada a la publicación y gestión de productos de una tienda online, integrando frontend y backend con PHP y AJAX.

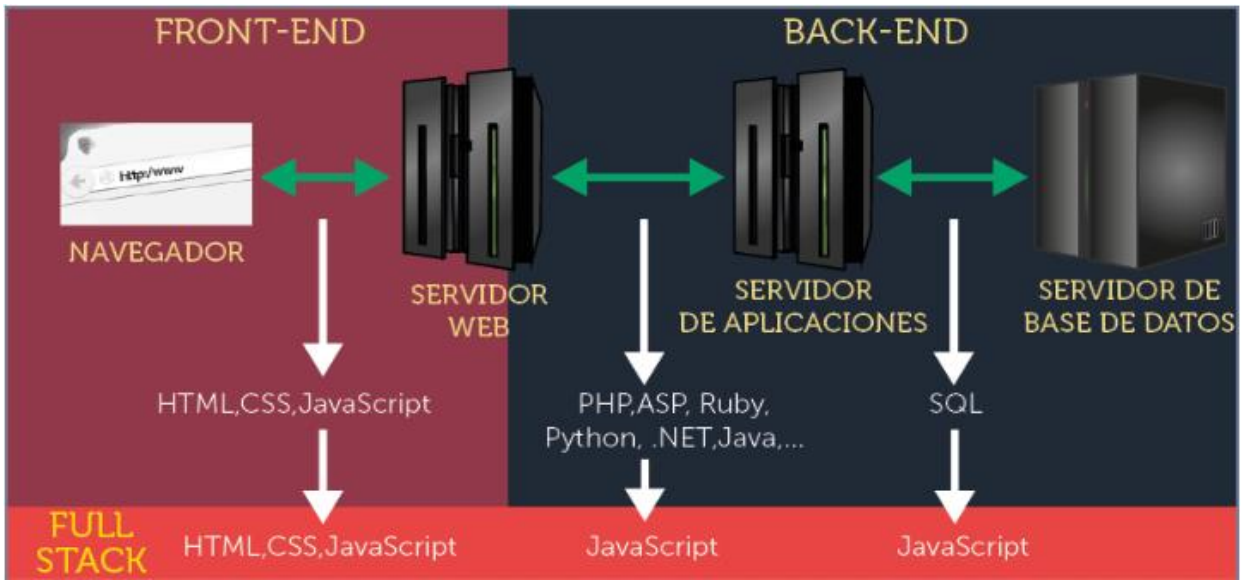
Plan del Curso

Duración total: 16 horas
Modalidad: Teórico-práctico
Nivel: Junior

Módulo 1: Introducción al Proyecto y Frontend (8 horas)

Sesión 1: Introducción y Configuración del Entorno (2 horas)

- **Teoría (30 min):**
  - Explicación del proyecto y su funcionalidad.
  - Introducción al stack de tecnologías: HTML, CSS, JavaScript, PHP, AJAX.



  - Estructura básica de un proyecto web.
- **Práctica (90 min):**
  - Configuración del entorno de desarrollo (Visual Studio Code, servidor local con XAMPP o WAMP).
  - Creación de la estructura de carpetas y archivos del proyecto.
  - Implementación de `index.html` con el diseño básico de la tienda online:
    - Header con logo y menú.
    - Contenedor para los productos.

Sesión 2: Diseño de la Interfaz con CSS (2 horas)

- **Teoría (20 min):**
  - Principios básicos de diseño responsivo (media queries y layout flexible).
  - Introducción a CSS Grid y Flexbox.
- **Práctica (100 min):**
  - Estilización del header y el menú de navegación.
  - Creación de tarjetas de productos en la galería (`#product-gallery`).
  - Implementación de diseño responsivo para diferentes tamaños de pantalla.

Sesión 3: Dinamización de la Galería con JavaScript (2 horas)

- **Teoría (30 min):**
  - Introducción a DOM Manipulation.
  - Uso básico de `fetch` para consumir datos.
- **Práctica (90 min):**
  - Creación de un archivo JSON estático (`data/products.json`).
  - Implementación de un script en `js/script.js` para cargar dinámicamente los productos desde el archivo JSON.
  - Renderizado de productos en la galería con estructura de tarjetas.

Sesión 4: Mejorando la Interacción del Usuario (2 horas)

- **Teoría (30 min):**
  - Mejora de la experiencia del usuario (UX) con JavaScript.
  - Introducción a eventos en JavaScript.
- **Práctica (90 min):**
  - Implementación de eventos para acciones simples (ejemplo: destacar producto al pasar el cursor).
  - Optimización de estilos visuales (hover effects, transiciones).

Módulo 2: Backend y CRUD con PHP y AJAX (8 horas)

Sesión 5: Introducción a PHP y Backend (2 horas)

- **Teoría (45 min):**
  - Introducción a PHP como lenguaje backend.
  - Conceptos básicos: lectura y escritura de archivos, manejo de JSON.
- **Práctica (75 min):**
  - Creación del archivo `php/products.php` para servir datos del JSON.
  - Configuración del servidor local para ejecutar el backend.
  - Testeo básico del backend desde el navegador.

Sesión 6: Consumo de PHP con AJAX (2 horas)

- **Teoría (30 min):**
  - Introducción a AJAX.
  - Diferencias entre AJAX y `fetch`.
- **Práctica (90 min):**
  - Modificación de `script.js` para consumir datos del backend (`products.php`) usando `fetch`.
  - Implementación de manejo de errores en el consumo de datos.

Sesión 7: Implementación de Funciones CRUD (2 horas)

- **Teoría (45 min):**
  - Explicación del modelo CRUD (Create, Read, Update, Delete).
  - Métodos HTTP para CRUD: GET, POST, PUT, DELETE.
- **Práctica (75 min):**
  - Creación de un formulario de administrador (`admin.html`) para agregar nuevos productos.
  - Uso de PHP para manejar solicitudes de creación y escritura en el archivo JSON.
  - Pruebas iniciales de inserción de datos.

Sesión 8: Finalización del CRUD y Revisión del Proyecto (2 horas)

- **Teoría (30 min):**
  - Seguridad básica en PHP (validación de datos y protección contra inyecciones).
  - Mejores prácticas para manejar archivos JSON.
- **Práctica (90 min):**
  - Finalización de las funciones CRUD:
    - Actualización de productos existentes (PUT).
    - Eliminación de productos (DELETE).
  - Validación y pruebas completas del proyecto.
  - Resolución de errores y optimización.

Recursos del Curso

- **Software:** Visual Studio Code, XAMPP/WAMP.
- **Enlaces:**
  - Documentación de PHP
  - MDN Web Docs - JavaScript
  - CSS Grid Guide
- **Material adicional:** Ejemplos de código, ejercicios.

Resultado Final

Al finalizar el curso, el estudiante tendrá:

1. Un proyecto completo de tienda online con frontend responsivo y dinámico.
2. Backend funcional con un CRUD implementado usando PHP y AJAX.
3. Habilidad para trabajar con archivos JSON y manejar solicitudes asincrónicas.

PROYECTO TIENDA EN LINEA WEB

Estructura del Proyecto

```
kotlin
Copiar código
tienda/
├── index.html
├── admin.html
├── about.html
├── js/
│   └── script.js
├── css/
│   └── styles.css
├── php/
│   └── products.php
├── data/
│   └── products.json
├── images/
│   └── (aquí estarán las imágenes de los productos)
└── logo.png
```

---

1. Archivo `index.html`

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Tienda Online</title>
    <link rel="stylesheet" href="css/styles.css">
</head>
<body>
    <header>
        <img src="logo.png" alt="Logo de la tienda" class="logo">
        <nav>
            <ul class="menu">
                <li><a href="index.html">Inicio</a></li>
                <li><a href="admin.html">Administrador</a></li>
                <li><a href="about.html">Acerca de</a></li>
            </ul>
        </nav>
    </header>
    <main>
        <h1>Productos Disponibles</h1>
        <section id="product-gallery"></section>
    </main>
    <footer>
        <p>© 2024 Tienda Online. Todos los derechos reservados.</p>
    </footer>
    <script src="js/script.js"></script>
</body>
</html>
```

---

2. Archivo `css/styles.css`

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
}

header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 10px 20px;
    background-color: #333;
    color: white;
}

header .logo {
    height: 50px;
}

.menu {
    list-style: none;
    display: flex;
    gap: 15px;
```

```
}

.menu li a {
    color: white;
    text-decoration: none;
}

h1 {
    text-align: center;
    margin: 20px 0;
}

#product-gallery {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
    gap: 15px;
    padding: 20px;
}

.product-card {
    border: 1px solid #ddd;
    padding: 15px;
    text-align: center;
    border-radius: 5px;
    background: #f9f9f9;
}

.product-card img {
    max-width: 100%;
    height: auto;
    border-radius: 5px;
}

footer {
    text-align: center;
    padding: 10px;
    background-color: #333;
    color: white;
}
```

---

3. Archivo `js/script.js`

```
document.addEventListener("DOMContentLoaded", function () {
    const gallery = document.getElementById("product-gallery");

    // Llama al archivo PHP para obtener los datos del JSON
    fetch("php/products.php")
        .then(response => response.json())
        .then(data => {
            data.forEach(product => {
                const card = document.createElement("div");
                card.className = "product-card";
                card.innerHTML = `
                    <img src="images/${product.imagen}" alt="${product.producto}">
                    <h3>${product.producto}</h3>
                    <p>${product.descripcion}</p>
                    <p><strong>Precio:</strong> $${product.precio}</p>
                `;
                gallery.appendChild(card);
            });
        })
```

```
        .catch(error => console.error("Error al cargar los productos:", error));
});
```

---

4. Archivo **`php/products.php`**

```
php
Copiar código
<?php
header('Content-Type: application/json');

// Ruta del archivo JSON
$jsonPath = '../data/products.json';

// Verifica si el archivo existe y envía los datos
if (file_exists($jsonPath)) {
    echo file_get_contents($jsonPath);
} else {
    echo json_encode([]);
}
?>
```

---

5. Archivo **`data/products.json`**

```
 [
    {
        "id": 1,
        "producto": "Camiseta Deportiva",
        "descripcion": "Camiseta de alta calidad para entrenamiento.",
        "imagen": "camiseta.jpg",
        "precio": 19.99
    },
    {
        "id": 2,
        "producto": "Zapatillas Running",
        "descripcion": "Zapatillas ligeras y cómodas para correr.",
        "imagen": "zapatillas.jpg",
        "precio": 49.99
    }
]
```

---

6. Archivos Adicionales

- **`admin.html`**: Página para gestionar productos (puedes ampliarla para agregar o editar productos).
- **`about.html`**: Página de información sobre la tienda.

TIENDA
QALLANA

ADICIONAR CARRITO DE COMPAS PROYECTO TIENDA EN LINEA WEB

1. **Archivo `index.html : Añadir logo de carrito de compras con la misma class del logo.`**
2. **Referencia:** crear <a href…. Apuntando a la pagina **carrito.html**
3. **Crear la página: carrito.html**

`carrito.html`

Añadiremos un botón para eliminar productos individualmente.

```
Copiar código
```

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Carrito de Compras</title>
    <link rel="stylesheet" href="css/styles.css">
</head>
<body>
    <header>
        <h1>Carrito de Compras</h1>
        <a href="index.html" class="volver-link">Volver a la Tienda</a>
    </header>
    <main>
        <section id="cart-items"></section>
        <div id="cart-summary">
            <h3>Total: $<span id="cart-total">0.00</span></h3>
            <button id="checkout-btn">Finalizar Compra</button>
        </div>
    </main>
    <script src="js/script.js"></script>
</body>
</html>
```

---

4. Actualización del archivo `js/script.js`

Añadimos las funciones para eliminar productos y finalizar la compra.

```
Copiar código
```

```
document.addEventListener("DOMContentLoaded", function () {
    const cart = JSON.parse(localStorage.getItem("cart")) || [];

    function renderProducts() {
        fetch("php/products.php")
            .then((res) => res.json())
            .then((products) => {
                const gallery = document.getElementById("product-gallery");
                gallery.innerHTML = "";
```

```
                products.forEach((product) => {
                    const card = document.createElement("div");
                    card.className = "product-card";
                    card.innerHTML = `
                        <img src="images/${product.imagen}" alt="${product.producto}">
                        <h3>${product.producto}</h3>
                        <p>${product.descripcion}</p>
                        <p>Precio: $${product.precio}</p>
                        <input type="number" id="cant${product.id}" min="1" value="1">
                        <button data-id="${product.id}" data-price="${product.precio}"
data-name="${product.producto}">Añadir al carrito</button>
                    `;
                    gallery.appendChild(card);
                });
                attachAddToCartListeners();
            });
    }

    function attachAddToCartListeners() {
        document.querySelectorAll(".product-card button").forEach((button) => {
            button.addEventListener("click", function () {
                const id = this.dataset.id;
                const name = this.dataset.name;
                const price = parseFloat(this.dataset.price);
                const quantity = parseInt(document.getElementById(`cant${id}`).value) ||
1;

                const product = cart.find((item) => item.id === id);
                if (product) {
                    product.quantity += quantity;
                } else {
                    cart.push({ id, name, price, quantity });
                }

                localStorage.setItem("cart", JSON.stringify(cart));
                alert("Producto añadido al carrito");
            });
        });
    }

    function renderCart() {
        const cartItems = document.getElementById("cart-items");
        const cartTotal = document.getElementById("cart-total");

        cartItems.innerHTML = "";
        let total = 0;

        cart.forEach((item, index) => {
            total += item.price * item.quantity;
            const cartItem = document.createElement("div");
            cartItem.className = "cart-item";
            cartItem.innerHTML = `
                <h3>${item.name}</h3>
                <p>Cantidad: ${item.quantity}</p>
                <p>Precio total: $${(item.price * item.quantity).toFixed(2)}</p>
                <button class="remove-btn" data-index="${index}">Eliminar</button>
            `;
            cartItems.appendChild(cartItem);
        });

        cartTotal.textContent = total.toFixed(2);

        attachRemoveListeners();
```

```
    }

    function attachRemoveListeners() {
        document.querySelectorAll(".remove-btn").forEach((button) => {
            button.addEventListener("click", function () {
                const index = parseInt(this.dataset.index);
                cart.splice(index, 1);
                localStorage.setItem("cart", JSON.stringify(cart));
                renderCart();
            });
        });
    }

    function handleCheckout() {
        document.getElementById("checkout-btn").addEventListener("click", function () {
            if (cart.length === 0) {
                alert("El carrito está vacío");
                return;
            }

            fetch("php/carrito.php", {
                method: "POST",
                headers: { "Content-Type": "application/json" },
                body: JSON.stringify(cart),
            })
                .then((res) => res.json())
                .then((response) => {
                    if (response.success) {
                        alert("Compra finalizada con éxito");
                        localStorage.removeItem("cart");
                        location.reload();
                    } else {
                        alert("Hubo un problema al procesar la compra");
                    }
                });
        });
    }

    if (document.getElementById("product-gallery")) {
        renderProducts();
    }

    if (document.getElementById("cart-items")) {
        renderCart();
        handleCheckout();
    }
});
```

---

5. Nuevo archivo `php/carrito.php`

Este archivo simula el procesamiento de la compra.

```php
<?php
header('Content-Type: application/json');

$requestBody = file_get_contents('php://input');
$cart = json_decode($requestBody, true);

if (!empty($cart)) {
```

```
    // Simula un procesamiento exitoso
    echo json_encode(["success" => true, "message" => "Compra realizada correctamente."]);
} else {
    echo json_encode(["success" => false, "message" => "El carrito está vacío."]);
}
?>
```

---

6. Actualizar en el CSS (`css/styles.css`)

```
body {
 font-family: Arial, sans-serif;
 margin: 0;
 padding: 0;
}
header {
 display: flex;
 justify-content: space-between;
 align-items: center;
 padding: 10px 20px;
 background-color: #000000;
 color: white;
}
header .logo {
    /*width: 150px; /* Ajusta el tamaño deseado */
    height: 150px; /* Igual al ancho para mantener el círculo */
    border-radius: 30%; /* Convierte en círculo */
    object-fit: cover; /* Asegura que la imagen llene el contenedor */
    background-color: transparent; /* Fondo transparente */
    border: 2px solid #143f0b; /* Opcional: borde alrededor de la imagen */
}
.menu {
 list-style: none;
 display: flex;
 gap: 15px;
}
.menu li a {
 color: white;
 text-decoration: none;
}
h1 {
 text-align: center;
 margin: 20px 0;
}
#product-gallery {
 display: grid;
 grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
 gap: 15px;
 padding: 20px;
```

```
}
/*.product-card {
 border: 1px solid #ddd;
 padding: 15px;
 text-align: center;
 border-radius: 5px;
 background: #f9f9f9;
}
.product-card img {
 max-width: 100%;
 height: auto;
 border-radius: 5px;
}*/
footer {
 text-align: center;
 padding: 10px;
 background-color: #333;
 color: white;
}

/***************CARRO******************/
.boton-carrito {
    background-color: #4CAF50; /* Verde */
    color: white;
    padding: 5px 20px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
    margin: 4px 2px;
    cursor: pointer;
    border-radius: 4px;
  }

  .numero-max-3 {
    width: 40px; /* Ajusta este valor según el ancho deseado */
    text-align: center;
    font-size: 16px;
    border: 1px solid #ccc;
    border-radius: 4px;
    padding: 5px;

    /* Estilo al obtener el foco */
    &:focus {
      outline: none;
      border-color: #007bff; /* Azul */
      box-shadow: 0px 0px 5px rgba(0, 123, 255, 0.5);
    }
  }
```

```css
main {
    padding: 20px;
}

.cart-link {
    text-decoration: none;
    color: #333;
    font-size: 16px;
    display: flex;
    align-items: center;
  }

#product-gallery, #cart-items {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
    gap: 20px;
}

.product-card, .cart-item {
    border: 1px solid #ddd;
    padding: 15px;
    text-align: center;
    border-radius: 5px;
}

.product-card img, .cart-item img {
    max-width: 100%;
    height: 150px;
    border-radius: 5px;
}

/******* carrito compra  ****/

.remove-btn {
    background-color: red;
    color: white;
    border: none;
    padding: 5px 10px;
    cursor: pointer;
    border-radius: 5px;
    margin-top: 5px;
}

#cart-summary {
    margin-top: 20px;
    text-align: center;
}

#checkout-btn {
    background-color: green;
```

```
    color: white;
    border: none;
    padding: 10px 20px;
    cursor: pointer;
    border-radius: 5px;
    font-size: 16px;
}

.carrito-flotante {
    position: fixed; /* Lo fija en la pantalla */
    top: 0px; /* Distancia desde la parte superior */
    left: 50%; /* Posiciona al 50% del ancho */
    transform: translateX(-50%); /* Centra horizontalmente */
    background-color: #000000; /* Color de fondo (verde) */
    color: white;
    padding: 5px 20px;
    text-align: center;
    text-decoration: none;
    font-size: 16px;
    border-radius: 120px;
  }
```

Flujo de Uso

1. **Eliminar productos:** En el carrito, cada producto tiene un botón para eliminarlo.
2. **Finalizar compra:** Al presionar "Finalizar Compra", se envía el contenido del carrito a `php/carrito.php`.
3. **Confirmación:** Si la compra es exitosa, el carrito se vacía y se recarga la página.

AÑADIR EL CRUD DE PRODUCTOS AL PROYECTO TIENDA EN LINEA WEB

a continuación, se genera la carpeta `admin` con las funcionalidades:

1. **Página de login** para administradores (datos en `users.json`).
2. **CRUD para productos** basado en el archivo `../data/products.json`, con subida de imágenes a la carpeta `images`.

Estructura del Proyecto

```
kotlin
Copiar código
tienda/
├── index.html
├── images/
│   └── (imágenes de productos)
├── data/
│   ├── products.json
│   └── users.json
├── admin/
│   ├── login.html
│   ├── dashboard.html
│   ├── css/
│   │   └── admin.css
│   ├── js/
│   │   └── admin.js
│   ├── php/
│   │   ├── login.php
│   │   ├── products.php
│   │   └── upload_image.php
│   └── logout.php
```

---

1. Archivo `admin/login.html`

Página de login para administradores.

```
html
Copiar código
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Administrador - Login</title>
    <link rel="stylesheet" href="css/admin.css">
</head>
<body>
    <div class="login-container">
        <h2>Iniciar Sesión</h2>
        <form id="login-form">
            <input type="text" id="username" placeholder="Usuario" required>
            <input type="password" id="password" placeholder="Contraseña" required>
```

```
            <button type="submit">Ingresar</button>
        </form>
        <div id="login-error"></div>
    </div>
    <script src="js/admin.js"></script>
</body>
</html>
```

---

2. Archivo **`admin/dashboard.html`**

Panel administrativo con CRUD.

```
html
Copiar código
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Administrador - Dashboard</title>
    <link rel="stylesheet" href="css/admin.css">
</head>
<body>
    <header>
        <h1>Panel de Administración</h1>
        <a href="logout.php" class="logout-link">Cerrar Sesión</a>
    </header>
    <main>
        <h2>Gestión de Productos</h2>
        <form id="product-form">
            <input type="hidden" id="product-id">
            <input type="text" id="product-name" placeholder="Nombre del Producto"
required>
            <textarea id="product-description" placeholder="Descripción"
required></textarea>
            <input type="number" id="product-price" placeholder="Precio" step="0.01"
required>
            <input type="file" id="product-image">
            <button type="submit">Guardar Producto</button>
        </form>
        <h3>Productos Existentes</h3>
        <table id="product-table">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Nombre</th>
                    <th>Descripción</th>
                    <th>Precio</th>
                    <th>Imagen</th>
                    <th>Acciones</th>
                </tr>
            </thead>
            <tbody></tbody>
        </table>
    </main>
    <script src="js/admin.js"></script>
</body>
</html>
```

---

3. Archivo `admin/css/admin.css`

Estilo para las páginas de administración.

```
css
Copiar código
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f9;
}

header {
    background-color: #333;
    color: white;
    padding: 10px 20px;
    display: flex;
    justify-content: space-between;
}

header h1 {
    margin: 0;
}

header a {
    color: white;
    text-decoration: none;
}

.login-container, main {
    max-width: 600px;
    margin: 50px auto;
    background: white;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

form {
    display: flex;
    flex-direction: column;
    gap: 10px;
}

table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
}

table th, table td {
    padding: 10px;
    text-align: left;
    border: 1px solid #ddd;
}

button {
    padding: 10px;
    background-color: #333;
    color: white;
    border: none;
```

```
    border-radius: 5px;
    cursor: pointer;
}

button:hover {
    background-color: #555;
}
```

---

4. Archivo **admin/js/admin.js**

Gestión del login y CRUD.

```
javascript
Copiar código
// Manejo de Login
if (document.getElementById("login-form")) {
    document.getElementById("login-form").addEventListener("submit", function (e) {
        e.preventDefault();
        const username = document.getElementById("username").value;
        const password = document.getElementById("password").value;

        fetch("php/login.php", {
            method: "POST",
            headers: { "Content-Type": "application/json" },
            body: JSON.stringify({ username, password }),
        })
            .then((res) => res.json())
            .then((data) => {
                if (data.success) {
                    window.location.href = "dashboard.html";
                } else {
                    document.getElementById("login-error").innerText = data.message;
                }
            });
    });
}

// Manejo de CRUD
if (document.getElementById("product-form")) {
    const fetchProducts = () => {
        fetch("php/products.php?action=read")
            .then((res) => res.json())
            .then((products) => {
                const tbody = document.querySelector("#product-table tbody");
                tbody.innerHTML = "";
                products.forEach((product) => {
                    tbody.innerHTML += `
                        <tr>
                            <td>${product.id}</td>
                            <td>${product.producto}</td>
                            <td>${product.descripcion}</td>
                            <td>$${product.precio.toFixed(2)}</td>
                            <td><img src="../images/${product.imagen}"
alt="${product.producto}" width="50"></td>
                            <td>
                                <button
onclick="editProduct(${product.id})">Editar</button>
                                <button
onclick="deleteProduct(${product.id})">Eliminar</button>
                            </td>
```

```
                    </tr>
                `;
            });
        });
    };

    document.getElementById("product-form").addEventListener("submit", function (e) {
        e.preventDefault();
        const id = document.getElementById("product-id").value;
        const name = document.getElementById("product-name").value;
        const description = document.getElementById("product-description").value;
        const price = document.getElementById("product-price").value;
        const image = document.getElementById("product-image").files[0];

        const formData = new FormData();
        formData.append("id", id);
        formData.append("name", name);
        formData.append("description", description);
        formData.append("price", price);
        if (image) formData.append("image", image);

        fetch("php/products.php?action=" + (id ? "update" : "create"), {
            method: "POST",
            body: formData,
        }).then(() => {
            fetchProducts();
            document.getElementById("product-form").reset();
        });
    });

    fetchProducts();
}

function editProduct(id) {
    fetch("php/products.php?action=read&id=" + id)
        .then((res) => res.json())
        .then((product) => {
            document.getElementById("product-id").value = product.id;
            document.getElementById("product-name").value = product.producto;
            document.getElementById("product-description").value = product.descripcion;
            document.getElementById("product-price").value = product.precio;
        });
}

function deleteProduct(id) {
    if (confirm("¿Deseas eliminar este producto?")) {
        fetch("php/products.php?action=delete&id=" + id, { method: "POST" }).then(() =>
fetchProducts());
    }
}
```

Aquí tienes los archivos necesarios para el backend en **PHP** que gestiona el sistema de autenticación de administradores y el CRUD para los productos.

Este archivo verifica las credenciales del administrador usando `users.json`.

1. Archivo **`admin/php/login.php`**

Código actualizado para compatibilidad con PHP 5.4:

```
php
Copiar código

<?php
header('Content-Type: application/json');

// Leer el cuerpo de la solicitud
$requestBody = file_get_contents('php://input');
$data = json_decode($requestBody, true);

// Validar campos
if (!isset($data['username']) || !isset($data['password'])) {
    echo json_encode(array('success' => false, 'message' => 'Faltan datos.'));
    exit;
}

$username = $data['username'];
$password = $data['password'];

// Leer usuarios desde users.json
$usersFile = '../../data/users.json';
if (!file_exists($usersFile)) {
    echo json_encode(array('success' => false, 'message' => 'Archivo de usuarios no
encontrado.'));
    exit;
}

$users = json_decode(file_get_contents($usersFile), true);
$validUser = false;

// Validar credenciales
foreach ($users as $user) {
    if ($user['username'] === $username && $user['password'] === $password) {
        $validUser = true;
        break;
    }
}

if ($validUser) {
    session_start();
    $_SESSION['logged_in'] = true;
    echo json_encode(array('success' => true));
} else {
    echo json_encode(array('success' => false, 'message' => 'Usuario o contraseña
incorrectos.'));
}
?>
```

---

2. Archivo `admin/php/products.php`

Código actualizado para el CRUD:

```
php
Copiar código
<?php
header('Content-Type: application/json');
$productsFile = '../../data/products.json';

if (!file_exists($productsFile)) {
    echo json_encode(array('success' => false, 'message' => 'Archivo de productos no
encontrado.'));
    exit;
}

$action = isset($_GET['action']) ? $_GET['action'] : null;

// Leer productos
$products = json_decode(file_get_contents($productsFile), true);

// Acciones CRUD
switch ($action) {
    case 'read':
        if (isset($_GET['id'])) {
            $id = intval($_GET['id']);
            $product = null;
            foreach ($products as $p) {
                if ($p['id'] === $id) {
                    $product = $p;
                    break;
                }
            }
            echo json_encode($product);
        } else {
            echo json_encode($products);
        }
        break;

    case 'create':
        $lastProduct = end($products);
        $id = $lastProduct ? $lastProduct['id'] + 1 : 1;
        $name = isset($_POST['name']) ? $_POST['name'] : null;
        $description = isset($_POST['description']) ? $_POST['description'] : null;
        $price = isset($_POST['price']) ? floatval($_POST['price']) : 0;
        $image = handleImageUpload();

        $newProduct = array(
            'id' => $id,
            'producto' => $name,
            'descripcion' => $description,
            'imagen' => $image,
            'precio' => $price,
        );

        $products[] = $newProduct;
        saveProducts($products);
        echo json_encode(array('success' => true));
        break;

    case 'update':
        $id = intval($_POST['id']);
```

```
        $name = isset($_POST['name']) ? $_POST['name'] : null;
        $description = isset($_POST['description']) ? $_POST['description'] : null;
        $price = isset($_POST['price']) ? floatval($_POST['price']) : 0;
        $image = handleImageUpload();

        foreach ($products as &$product) {
            if ($product['id'] === $id) {
                $product['producto'] = $name;
                $product['descripcion'] = $description;
                $product['precio'] = $price;
                if ($image) {
                    $product['imagen'] = $image;
                }
                break;
            }
        }
        saveProducts($products);
        echo json_encode(array('success' => true));
        break;

    case 'delete':
        $id = intval($_GET['id']);
        $newProducts = array();
        foreach ($products as $product) {
            if ($product['id'] !== $id) {
                $newProducts[] = $product;
            }
        }
        $products = $newProducts;
        saveProducts($products);
        echo json_encode(array('success' => true));
        break;

    default:
        echo json_encode(array('success' => false, 'message' => 'Acción no válida.'));
        break;
}

// Función para guardar productos en el archivo JSON
function saveProducts($products) {
    global $productsFile;
    file_put_contents($productsFile, json_encode($products, JSON_PRETTY_PRINT));
}

// Función para manejar la carga de imágenes
function handleImageUpload() {
    if (isset($_FILES['image']) && $_FILES['image']['error'] === UPLOAD_ERR_OK) {
        $uploadDir = '../../images/';
        $fileName = basename($_FILES['image']['name']);
        $filePath = $uploadDir . $fileName;

        if (move_uploaded_file($_FILES['image']['tmp_name'], $filePath)) {
            return $fileName;
        }
    }
    return null;
}
?>
```

3. Archivo `admin/logout.php`

Cierra la sesión del administrador.

```
php
Copiar código
<?php
session_start();
session_destroy();
header('Location: login.html');
exit;
?>
```

4. Archivo `data/users.json`

Este archivo contiene los datos de los usuarios administradores.

```
json
Copiar código
[
    {
        "username": "admin",
        "password": "12345"
    },
    {
        "username": "manager",
        "password": "admin123"
    }
]
```

Cambios realizados para compatibilidad con PHP 5.4:

1. **Eliminación de funciones anónimas:**
   - Reemplacé funciones como `fn($p)` con bucles tradicionales (`foreach`).
2. **Compatibilidad con manejo de arrays:**
   - Reemplacé `array_filter` con bucles manuales para filtrar y buscar elementos.
3. **Compatibilidad con JSON:**
   - Usé arreglos de estilo antiguo `array()` en lugar de `[]` (PHP 5.4 soporta ambos, pero es más común usar `array()` en esa versión).