UPTrep
UNIVERSIDAD POLITECNICA DE TLAXCALA REGION PONIENTE

Universidad Politécnica de Tlaxcala Región Poniente

Ingeniería en Sistemas Computacionales

Administración a Base de Datos

Alumnos:

2SIC008 Isaac Brandon Martinez Ramirez

22SIC016 Gabriel Perez Tzompa

Docente:

Ing. Vanesa Tenopala Zavala

Reporte de Proyecto Final

Ciclo Cuatrimestral Mayo – Agosto

Fecha: 14 de agosto 20024

Introducción.

En este reporte se hablada sobre el proceso de creación de una base de datos y una pagina web sobre la venta de productos relacionados con las tecnologías y “gaming” con el nombre de “**WestTronics**” y como fue el proceso de la elaboración de cada paso en este pequeño proyecto.

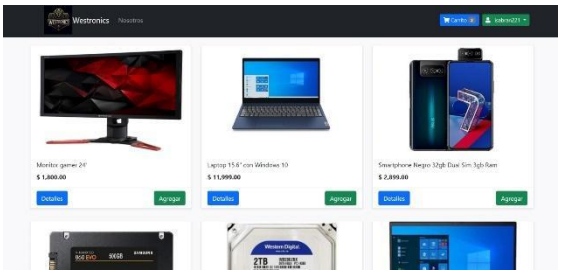
1. Diseño responsivo

Desarrollar un proyecto de tienda en línea, enfocándose en el diseño responsivo del sitio utilizando Bootstrap. Después de integrar un botón de PayPal para realizar pagos, se centra en crear una estructura sencilla y adaptable que funcione correctamente en dispositivos de diferentes tamaños.

Contenidos Principales

1. **Preparación del Proyecto:**
    - Uso de Visual Studio Code como editor principal.
    - Estructuración de carpetas para css, js, e imágenes.
    - Creación de un archivo index.php como punto de partida del proyecto.
2. **Implementación de Bootstrap:**
    - Uso de la versión 5.1 de Bootstrap, incluyendo CSS y JavaScript a través de CDNs para optimizar la carga del sitio.
    - Selección de una plantilla predefinida ("Álbum") para visualizar productos.
3. **Creación de la Barra de Navegación:**
    - Personalización de la barra de navegación para incluir elementos como el título de la tienda y enlaces de menú.
    - Eliminación de elementos innecesarios y ajustes para mejorar la visualización en diferentes dispositivos.
4. **Diseño de las Tarjetas de Producto:**
    - Uso de cards para mostrar los productos, incluyendo título, imagen, precio, y botones de acción ("Detalles" y "Agregar al carrito").
    - Creación de una estructura responsiva que se adapta a diferentes tamaños de pantalla.
5. **Optimización del Sitio:**
    - Añadir un archivo estilos.css para personalizar el padding de la página y mejorar la disposición del contenido.
    - Prueba de la responsividad del diseño, asegurando que los productos se muestren

adecuadamente en dispositivos móviles.



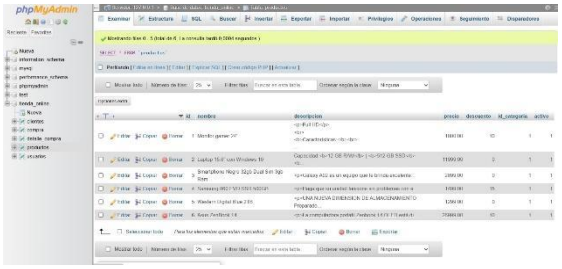
2. Conexión a BD y Tablas Productos.

Se realiza la conexión de una tienda online a una base de datos utilizando PHP y MySQL. El objetivo principal es crear una tabla que almacene los productos y luego visualizarlos de manera dinámica en el sitio web.

Pasos principales:

1. **Creación de la base de datos:**
   - Se utiliza **phpMyAdmin** para crear una base de datos llamada tienda_online con codificación utf8_spanish.
   - Se crea una tabla llamada productos con seis columnas: id, nombre, descripción, precio, categoría y activo.
2. **Definición de las columnas:**
   - id: entero auto incremental.
   - nombre: tipo VARCHAR, con un tamaño sugerido de 200 caracteres.
   - descripción: tipo TEXT para contenido más largo.
   - precio: tipo DECIMAL con dos decimales.
   - categoría: entero que referencia una categoría.
   - activo: indica si el producto está activo.
3. **Conexión a la base de datos:**
   - Se crea un archivo database.php en una carpeta config, donde se define una clase Database para manejar la conexión a la base de datos.
   - Se configuran los atributos de conexión: hostname, dbname, username, password, y charset.
   - La clase Database utiliza PDO para la conexión y manejo de excepciones.
4. **Consulta y visualización dinámica:**
   - En el archivo index.php, se incluye la conexión a la base de datos y se realiza una consulta para seleccionar productos activos.
   - Se utiliza un bucle foreach para recorrer los resultados y mostrar los productos de forma dinámica.
   - La imagen del producto se carga desde una carpeta, y si no existe, se muestra una imagen predeterminada.
   - Se formatea el precio con dos decimales y separadores de miles.
5. **Pruebas y validaciones:**
   - Se insertan productos de prueba en la tabla.
   - Se valida que los productos se muestren correctamente en la tienda y que las imágenes se carguen según la ruta configurada.

- Se verifica que los productos inactivos no se muestren.



3. Detalles de Productos

Crearemos la página para visualizar los detalles completos del producto.

1. Creación de la Página de Detalles

- **Duplicación de Plantilla**:
  - Copiar index.php a detalles.php.
  - Ajustar el código para mostrar detalles específicos del producto.

2. Enlaces Seguros

- **Generación de Token**:
  - Crear config.php para manejar constantes como el token de cifrado.
  - Implementar cifrado en URLs para evitar alteraciones.

3. Validación de Datos

- **Validación de URL**:
  - Verificar si los parámetros id y token están definidos.
  - Comparar el token recibido con el generado para asegurar la integridad.

4. Consulta de Producto

- **Consultas SQL**:
  - Verificar existencia del producto en la base de datos.
  - Consultar información detallada del producto (nombre, descripción, precio).

5. Diseño de la Página de Detalles

- **Estructura HTML**:
  - Crear diseño con Bootstrap: fila, columnas, imagen principal, descripción, precio.
- **Botones**:
  - Añadir botones para "Comprar ahora" y "Agregar al carrito".

6. Imágenes del Producto

- **Carga Dinámica**:
  - Mostrar imágenes del producto en un carrusel.
  - Implementar un directorio para manejar múltiples imágenes.

7. Descuentos

- **Implementación**:
  - Añadir columna de descuento en la base de datos.
  - Calcular y mostrar precio con descuento si aplica.

8. Errores y Validaciones Finales

- **Manejo de Errores**:
  - Validar y mostrar mensajes de error si la URL está alterada o si faltan datos.
- **Corrección de Errores**:
  - Solucionar problemas con la visualización de precios y descuentos.

4. Agregar al Carrito de Compra

Objetivo: Implementar la funcionalidad para agregar productos al carrito de compras y actualizarlo en tiempo real.

Detalles del Producto

- **Descripciones:** Añadidas desde la base de datos usando HTML para mejorar la visualización.
- **Validación de Imágenes:** Se añadió una validación para productos sin imágenes y evitar errores.

Implementación del Carrito de Compras

- **Uso de Sesiones:**
  - Se usa PHP para manejar sesiones y almacenar el carrito.
  - No es necesario registrarse para agregar productos; solo se requiere al momento del pago.
- **Actualización del Carrito:**
  - **JavaScript y AJAX:** Se utiliza fetch para enviar datos al servidor sin recargar la página.
  - **Función producto():** Envía el ID del producto y un token mediante POST.

Manejo del Carrito en el Servidor

Objetivo: Mostrar los productos del carrito de compras, incluyendo cantidades, subtotales, y opciones para eliminar productos y realizar el pago.

Preparación del Proyecto

- **Editor:** Visual Studio Code
- **Archivo Nuevo:** checkout.php creado a partir de una copia del index.php.
- **Archivo carrito.php:**
  - Valida si se reciben el ID y el token.
  - Almacena y actualiza el carrito en la sesión PHP.
  - Incrementa la cantidad de productos si ya existe en el carrito.
- **Respuestas:**
  - Devuelve el número total de productos en el carrito.
  - Muestra la cantidad actualizada en la interfaz.

Pruebas y Ajustes

- **Interfaz:**
  - Se muestra el número de productos en el carrito en tiempo real.
  - Se realizan pruebas con varios productos para verificar el funcionamiento.
- **Eliminar Carrito:**
  - Se agrega una función para eliminar todos los productos del carrito mediante session_destroy()

5. Mostrar Producto del Carrito

  - Se eliminan elementos innecesarios para enfocarse en el carrito de compras.

Consulta de Productos en el Carrito

- **Validación de Sesión:** Se verifica si existe una sesión de carrito y se extraen los productos.
- **SQL:** Se realiza una consulta para obtener detalles de los productos,

incluyendo descuentos y cantidades.

Desarrollo de la Interfaz

- **HTML Estructura:**
  - Se crea una tabla con encabezados para Producto, Precio, Cantidad, Subtotal, y Eliminar.
  - En el cuerpo de la tabla, se usan foreach para listar los productos del carrito.
- **Validaciones:**
  - Si el carrito está vacío, se muestra un mensaje "Lista Vacía".
  - Si hay productos, se calcula y muestra el subtotal y el total.

Implementación Dinámica

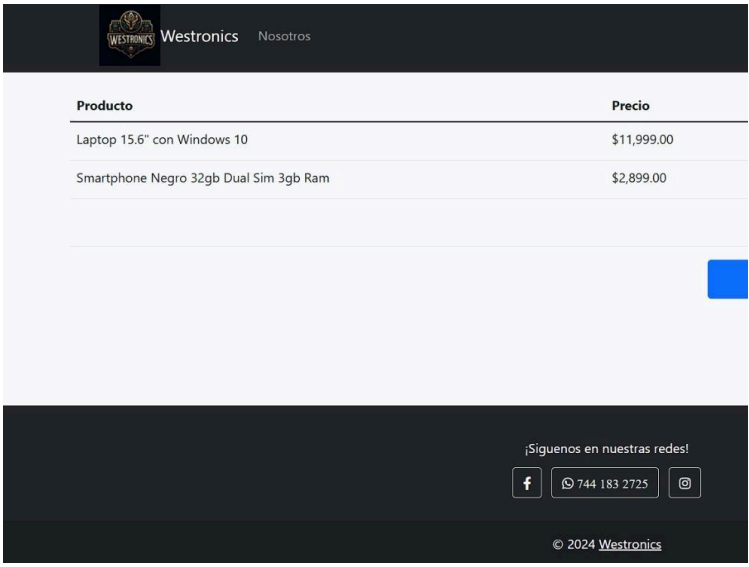
- **Inputs de Cantidad:**
  - Se usa un campo input para ajustar la cantidad de productos, con validaciones HTML5.
  - El subtotal se calcula dinámicamente y se muestra en la tabla.
- **Subtotal y Total:**
  - Se calcula el subtotal multiplicando la cantidad por el precio con descuento.
  - Se acumula el total en una variable y se muestra al final de la tabla.

Botones de Acción

- **Eliminar Producto:**
  - Se agrega un botón para eliminar productos, que abre un modal para confirmar la acción.
- **Realizar Pago:**
  - Se añade un botón "Realizar Pago" fuera de la tabla, con estilo y diseño responsivo.

Problemas y Soluciones

- **Actualización de Cantidades:**
  - Se corrige un error en la actualización de cantidades en el carrito.
- **Visualización y Diseño:**
  - Se ajusta el diseño para que sea responsive y se visualice correctamente en diferentes dispositivos.



6. Modificar Productos del Carrito

Modificaciones al carrito de compras, incluyendo la capacidad de modificar cantidades y eliminar productos.

Implementación de Modificaciones

1. **Actualizar Cantidades:**
   - **Script:** checkout.php.

- **Funcionalidad:** Se añade una función actualizaCantidad para modificar la cantidad de productos en el carrito. Se actualiza el subtotal y total de la compra automáticamente.
- **Envío de Datos:** Se utiliza JavaScript para enviar datos al script actualizar_carrito.php mediante POST.
- **Validaciones:** Se realizan validaciones para asegurar que la cantidad sea mayor a 0 y numérica.

2. **Cálculo de Totales:**
   - **Subtotal y Total:** Se recalculan en el frontend utilizando JavaScript.
   - **Formato de Moneda:** Se usa Intl.NumberFormat para dar formato al total.
3. **Eliminar Productos:**
   - **Implementación de Modal:** Se utiliza Bootstrap 5 para implementar un modal de confirmación.
   - **Funcionalidad del Modal:** Permite confirmar la eliminación del producto seleccionado.
   - **JavaScript:** Se añade lógica para capturar el ID del producto a eliminar y se llama a la función eliminar en actualizar_carrito.php.
   - **Acción de Eliminación:** Se elimina el producto del carrito y se actualiza la vista.

Pruebas y Depuración

- **Pruebas de Funcionalidad:** Se realizan pruebas para asegurar que la actualización de cantidades y la eliminación de productos funcionen correctamente.
- **Depuración:** Se utiliza la consola y herramientas de desarrollo para depurar errores en el código.

7. Email con Detalle de Compra

- Enviar correos electrónicos con detalles de compra usando PHP y la biblioteca PHPMailer.

Pasos Principales

1. **Instalación de PHPMailer**
   - **Métodos de Instalación:**
     - **Composer:** Usar composer require phpmailer/phpmailer .
     - **Manual:** Descargar desde GitHub y agregar las carpetas src y languages al proyecto.
   - **Configuración en Visual Studio Code:**
     - Crear archivo enviar_email.php en la carpeta clases.
2. **Configuración del Script de Envío de Correo**
   - **Incluir Bibliotecas:**
     - Ajustar rutas y namespaces según la estructura del proyecto.
   - **Ejemplo Básico:**
     - Configurar SMTP, autenticación, y

dirección de correo del remitente.

- Definir el contenido del correo en formato HTML con título, cuerpo y formato UTF-8.

- **Manejo de Errores:**
  - Capturar y mostrar errores en caso de fallos en el envío.

3. **Integración con el Proyecto**
   - **Incluir Script en captura.php:**
     - Incluir enviar_email.php para enviar el correo después de procesar la compra.
   - **Ajuste en pago.php:**
     - Redireccionar a una página de confirmación (completado.php) después de realizar el pago.
4. **Pruebas y Debugging**
   - **Verificación:**
     - Asegurarse de que el correo se envía correctamente revisando la consola y la red.
   - **Solución de Problemas:**
     - Verificar contraseñas y configuraciones SMTP.
5. **Modificaciones Finales**
   - **Actualización de completado.php:**
     - Agregar lógica para mostrar detalles de la compra y errores en caso de fallos.
   - **Mostrar Detalles de Compra:**
     - Mostrar información como folio, fecha, total y detalles de los productos comprados en formato HTML.
6. **Conclusión**
   - **Verificación Final:**
     - Realizar pruebas completas para asegurar el funcionamiento correcto del sistema de envío de correos.

8. Registro y Validación de Formulario de Cliente

HTML: Define la estructura del formulario, con campos como nombre, apellido, correo electrónico, contraseña y confirmación de contraseña.

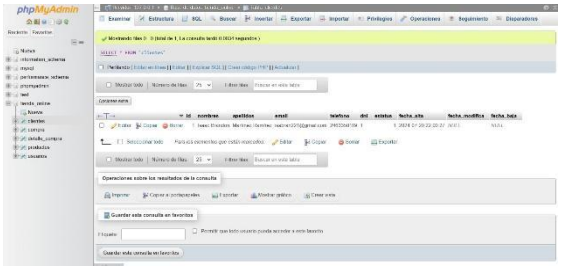
Bootstrap: Estiliza el formulario para que sea visualmente atractivo y fácil de usar.

JavaScript: Añade una validación básica para asegurarse de que las contraseñas coincidan antes de enviar el formulario.

mplementar validaciones del lado del servidor (backend) en un formulario de registro de usuarios para una tienda en línea. Aquí se destacan los puntos más importantes

1. Importancia de las Validaciones en Backend

- Se recomienda realizar validaciones tanto del lado del cliente (HTML y JavaScript) como del lado del servidor (PHP) para asegurar la integridad de los datos.
- Las validaciones en el cliente pueden ser fácilmente omitidas por un usuario con conocimientos técnicos, por lo que es crucial reforzar con validaciones en el servidor.



2. Validaciones Implementadas

- **Validación de Campos Vacíos:** Se crea una función esNulo que recibe un arreglo de parámetros y verifica si alguno de ellos está vacío. Se usa la función strlen junto con trim para asegurarse de que no haya espacios en blanco.
- **Validación de Estructura de Email:** Mediante filter_var con FILTER_VALIDATE_EMAIL, se verifica que el email tenga una estructura válida.
- **Validación de Coincidencia de Contraseñas:** Se implementa una función validaPassword que utiliza strcmp para comparar las contraseñas. Si son diferentes, se retorna un error.
- **Validación de Duplicados en la Base de Datos:**
    - **Usuario Existente:** Se crea la función usuarioExiste que consulta la base de datos para verificar si el nombre de usuario ya está registrado.
    - **Email Existente:** Similar a la función anterior, emailExiste verifica si el correo electrónico ya está registrado en la base de datos.

3. Implementación en el Registro

- Se incluyen todas las validaciones en el proceso de registro antes de insertar los datos en la base de datos. Si alguna validación falla, se agregan los errores a un arreglo errors.
- **Mostrar Errores:** Se implementa la función mostrarMensajes, que recorre el arreglo errors y despliega los mensajes correspondientes utilizando una alerta de Bootstrap.

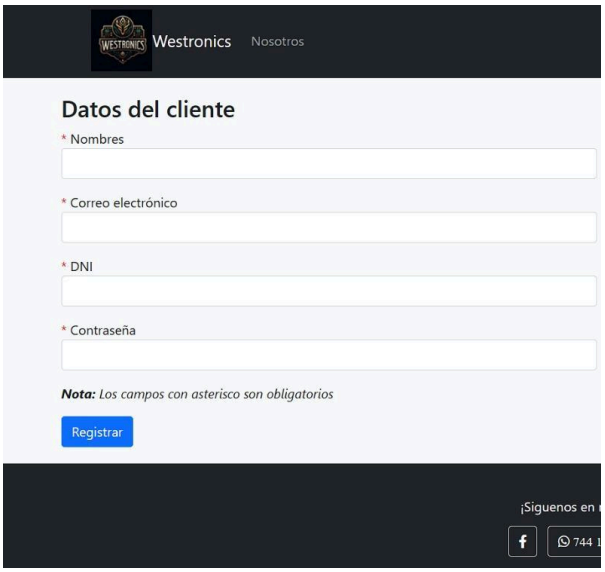
4. Prueba y Manejo de Errores

- Se realizan pruebas para verificar que los mensajes de error aparezcan correctamente en caso de:
    - Campos vacíos.
    - Email con estructura inválida.
    - Contraseñas que no coinciden.
    - Usuario o email duplicados.
- Se explica cómo mantener los datos ingresados en el formulario en caso de error para que el usuario no tenga que reingresarlos.

5. Mejoras Futuras

- **Validaciones en Tiempo Real:** Se planea validar el usuario en tiempo real mientras se está escribiendo, mostrando alertas si ya existe.
- **Confirmación por Email:** Una vez registrado, el usuario deberá

confirmar su correo electrónico para activar su cuenta.



9. Validación a Tiempo Real

Implementar validación en tiempo real en un formulario de registro para una tienda en línea. Se busca mejorar la experiencia del usuario al validar datos como el correo electrónico, nombre de usuario y DNI mientras se introducen, sin necesidad de enviar el formulario completo.

1. Validación en Tiempo Real

- El sistema valida en tiempo real si los datos introducidos en ciertos campos (como correo y usuario) ya están registrados.

- Esto evita que el usuario envíe el formulario solo para recibir errores, permitiendo una corrección instantánea.

2. Uso de AJAX para Validación

- Se crea un archivo `clienteAjax.php` que gestiona las solicitudes AJAX.

- Mediante JavaScript y AJAX, se envían los datos introducidos al backend (PHP) para su validación.

- La respuesta del servidor se procesa y se muestra al usuario en tiempo real.

3. Implementación en Visual Studio Code

- Se añade un evento `blur` en el campo de usuario, que dispara la función de validación cuando el usuario cambia de campo.

- La función `existeUsuario` envía los datos a `clienteAjax.php` y recibe la respuesta del servidor.

4. Respuesta y Procesamiento

- Si el usuario o correo ya existe, se limpia el campo y se muestra un mensaje de error.

- Se utiliza un `span` con Bootstrap para mostrar mensajes de error como "Usuario no disponible".

5. Manejo de Errores

- Se explica cómo usar la consola del navegador para depurar errores en las solicitudes AJAX.

- Se ajustan detalles en el código para asegurar que las solicitudes AJAX se procesen correctamente.

6. Validación Extendida

- La validación se extiende al campo de correo electrónico de manera similar, utilizando la misma lógica.

- Se menciona que este enfoque puede aplicarse a otros campos del formulario.

7. Beneficios

- Mejora la usabilidad y la experiencia del usuario al permitir correcciones inmediatas.

- Facilita la validación de formularios complejos de manera más eficiente y amigable

10. Activa Cuenta via Correo Electronico

Explica cómo implementar la funcionalidad para activar una cuenta de

usuario mediante un enlace enviado por correo electrónico.

Pasos principales:

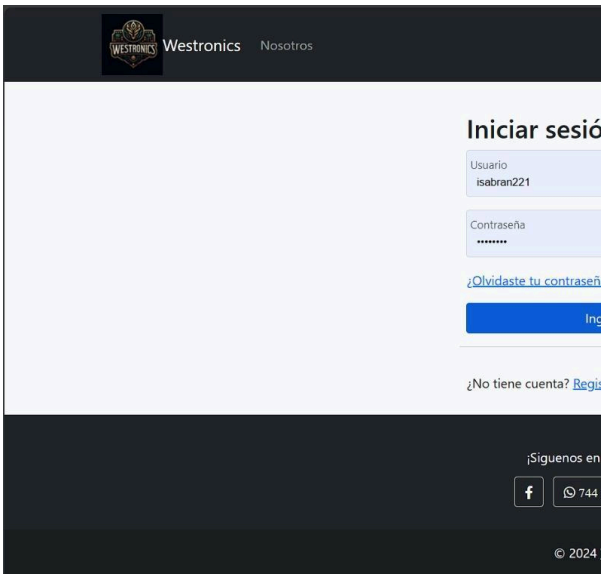
1. **Preparación inicial:**
   - Se trabaja en un formulario que ya fue creado y validado
   - Ahora, se agrega la funcionalidad para enviar un correo electrónico que permitirá al usuario activar su cuenta.
2. **Configuración del correo electrónico:**
   - Se crea una nueva clase `Mailer` en PHP para gestionar el envío de correos.
   - Esta clase reutiliza configuraciones anteriores almacenadas en un archivo de configuración (`config.php`).
   - La clase `Mailer` incluye métodos para recibir el correo del destinatario, asunto y cuerpo del mensaje, y manejar el envío de manera genérica.
3. **Implementación del envío de correo:**
   - Se configura el correo utilizando la clase `Mailer` y se asegura que la URL de activación esté bien formada.
   - Se incorpora un token de seguridad en la URL para garantizar que solo el destinatario pueda activar la cuenta.
   - Se realiza una validación para confirmar que el correo electrónico se envía correctamente.
4. **Generación de la URL de activación:**
   - La URL incluye un token único y el ID del usuario, que son esenciales para la activación segura.
   - Se enfatiza la importancia de construir correctamente la URL para evitar problemas al momento de la activación.
5. **Activación de la cuenta:**
   - Se crea un script llamado `activar_cliente.php` que procesa la activación de la cuenta.
   - El script valida el token y el ID del usuario, y si son correctos, activa la cuenta en la base de datos.
   - También se muestra cómo gestionar posibles errores, como la falta de parámetros en la URL.
6. **Pruebas y validaciones:**
   - Se realizan pruebas para verificar el correcto funcionamiento del proceso de activación.
   - Se asegura que el sistema no permita la activación de cuentas con datos manipulados o incorrectos.
   - Se recomienda cifrar el ID del usuario para añadir una capa adicional de seguridad.
7. **Mejoras adicionales:**
   - Se sugiere limpiar el token después de la activación para evitar posibles usos indebidos.
   - También se muestra cómo redireccionar al usuario a la página principal en caso de errores en la activación.

11. Creación de un Formulario de Inicio de Sesion

Crear un formulario de inicio de sesión para una tienda en línea, permitiendo a los usuarios iniciar sesión, asociar sus compras y acceder a su historial.

Pasos Clave:

1. **Preparación del Entorno:**
    - Se abre el proyecto en Visual Studio Code.
    - Se duplica el archivo registro.php, renombrándolo como login.php.
    - Se depura la vista eliminando partes innecesarias y dejando solo la estructura básica del encabezado.
2. **Creación del Formulario de Inicio de Sesión:**
    - Se agrega un formulario sencillo con campos para el nombre de usuario y la contraseña.
    - Se utiliza la clase form-floating para que los input tengan un estilo flotante.
    - Se ajusta la hoja de estilos para centrar y limitar el ancho del formulario a 350 píxeles.
    - Se añade una opción para recuperar la contraseña en caso de olvido, aunque se implementará completamente en un video posterior.
    - Se incluye un botón de "Ingresar" y un enlace para registrarse si no se tiene una cuenta.
3. **Validaciones en el Backend:**
    - Se configura la validación de los campos en el servidor.
    - Se crea una función login para verificar la existencia del usuario y comparar la contraseña ingresada con la almacenada en la base de datos, utilizando la función password_verify de PHP.
    - Se añade una función es_activo para verificar si la cuenta del usuario está activada antes de permitir el inicio de sesión.
    - Se manejan errores de forma genérica para evitar que los usuarios malintencionados obtengan información sensible.
4. **Implementación de la Sesión:**
    - Si las credenciales son correctas, se inicia la sesión y se almacenan los datos del usuario en variables de sesión (user_id, user_name).
    - Se redirecciona al usuario a la página de inicio y se termina la ejecución del script con exit.
5. **Pruebas y Validaciones Finales:**
    - Se realizan pruebas para verificar el correcto funcionamiento de las validaciones y el inicio de sesión.
    - Se imprime la variable de sesión en la vista para confirmar que se ha iniciado sesión.
    - Se añade un icono junto al nombre de usuario y se valida la existencia de la sesión para mostrar u ocultar enlaces de "Ingresar" o el nombre de usuario.
    - Finalmente, se realiza una prueba completa del flujo de inicio de sesión, asegurando que los productos en el carrito se asocian correctamente al usuario.



12. Recuperar Contraseña

Creación de una función para recuperar la contraseña en un proyecto de tienda en línea. El proceso incluye la solicitud de restablecimiento de contraseña de manera segura, utilizando un token y validaciones.

Estructura del Proceso

1. **Creación del Formulario de Recuperación**:
   - Se crea un formulario sencillo con Bootstrap en un archivo recupera.php.
   - El formulario solicita al usuario ingresar su correo electrónico asociado a la cuenta.
2. **Validación del Correo Electrónico**:
   - Se verifica que el correo ingresado existe en la base de datos.
   - Se genera un token para el restablecimiento de la contraseña.
   - Se envía un correo electrónico al usuario con un enlace que contiene el token.
3. **Desarrollo del Backend**:
   - Conexión a la base de datos para verificar y extraer la información del usuario.
   - Preparación de una consulta SQL para validar la existencia del correo electrónico.
   - Generación y almacenamiento del token de recuperación en la base de datos.
4. **Envío del Correo de Recuperación**:
   - Utilización de una clase mailer para enviar el correo.
   - El correo contiene un enlace con el token, que permite al usuario acceder a la página de restablecimiento de contraseña.
5. **Validación y Restablecimiento de Contraseña**:
   - Se crea una página reset_password.php que recibe y valida el token.
   - El usuario ingresa su nueva contraseña y se realiza la actualización en la base de datos.

- Se verifica que el token sea válido y que no haya sido alterado.

6. **Actualización de la Contraseña**:
   - Si la validación es correcta, la nueva contraseña es guardada en la base de datos.
   - El token es invalidado después de su uso, asegurando que no pueda ser reutilizado.

13. Asignar Compras a Cliente

- **Objetivo:** Asignar compras a clientes en una tienda online, permitiendo un histórico de compras por cliente.
- **Contexto:** Ya se tiene implementado el registro e inicio de sesión para los clientes.

Implementación del Menú en la Tienda

1. **Creación del Script menu.php:**
   - Se crea un script para el encabezado de la tienda, reutilizando el código del archivo index.php.
   - Este script se incluye en otros archivos (como detalles.php) para unificar la barra de navegación.
2. **Botón de Inicio de Sesión:**
   - Se modifica el botón de inicio de sesión para que sea un dropdown.
   - Dependiendo del estado de la sesión, muestra opciones como cerrar sesión o redirigir a la página de inicio de sesión (login.php).

Validación del Estado de la Sesión en el Proceso de Compra

1. **Modificación en el checkout:**
   - Se verifica si el cliente ha iniciado sesión al intentar realizar un pago. Si no lo ha hecho, se redirige a la página de inicio de sesión.
   - Se envía un parámetro pago para gestionar la redirección correcta después de iniciar sesión.
2. **Modificaciones en login.php:**
   - Se añade la funcionalidad para manejar el parámetro pago enviado en el proceso de checkout, redirigiendo al cliente de vuelta al proceso de pago tras iniciar sesión.

Finalización del Pago y Registro del Cliente

1. **Captura de Pago con PayPal:**
   - Se asigna el ID del cliente a la compra y se registra su correo electrónico en la base de datos.
   - Se implementa el envío de un correo de confirmación usando un script optimizado (mailer.php).
2. **Captura de Pago con Mercado Pago:**
   - Se realizan ajustes similares a los del proceso de PayPal, asegurando que el cliente esté vinculado correctamente a la compra.

Ajustes Finales y Consideraciones

- **Manejo del Carrito de Compras:** Se ajusta el manejo de la sesión para que, al cerrar sesión, el carrito no se borre, solo las variables de usuario.
- **Histórico de Compras:** Con estas implementaciones, cada cliente puede revisar su histórico de compras, asegurando un control más preciso en la tienda online.

14. Historial de Compra

explica cómo implementar una función en una tienda en línea para mostrar un historial de compras realizado por el cliente. El proceso incluye la creación de un nuevo script en PHP que recupera y presenta la información de las compras desde la base de datos.

Pasos Principales

1. **Creación de la opción "Mis Compras":**
   - Se agrega una nueva opción en el menú de navegación bajo el nombre "Mis Compras".
   - El script asociado se llamará compras.php, que se creará en la raíz del proyecto.
2. **Creación del Script compras.php:**
   - Se utiliza una plantilla existente (plantilla.php) para la estructura básica del HTML.
   - Se eliminan validaciones innecesarias y se adapta el código para que funcione correctamente con el sistema.
3. **Consulta a la Base de Datos:**
   - Se realiza una consulta SQL para recuperar las compras del cliente que haya iniciado sesión, seleccionando campos como ID de transacción, fecha, estado, total y medio de pago.
   - Se ordenan las compras de forma descendente por la fecha.
4. **Visualización de las Compras:**
   - Utilizando Bootstrap, se crea una estructura de "cards" (tarjetas) para mostrar cada compra de manera visualmente atractiva.
   - Se implementa un bucle while para iterar sobre cada compra y generar una tarjeta para cada una, mostrando detalles como la fecha, ID de transacción y total.
5. **Personalización del Diseño:**
   - Se agregan clases CSS para mejorar la separación entre las tarjetas.
   - Se exploran opciones para personalizar aún más el diseño, como cambiar el color de las tarjetas o sus bordes.
6. **Preparación para Detalles de la Compra:**
   - Se prepara un botón que llevará a un script futuro (compra_detalle.php) que mostrará más información detallada sobre cada compra, como los artículos comprados y el método de pago utilizado.

15. Detalles de la Compra

Se añade una sección para ver los detalles de cada compra, incluyendo los productos comprados y los totales.

Implementación

1. **Creación de la Plantilla PHP**
   - Se crea un archivo compra_detalle.php para mostrar los detalles de una compra específica.
   - Se pasa el ID de la transacción a la plantilla a través de la URL.
2. **Seguridad**
   - Se genera un token para evitar accesos no autorizados.
   - Se valida que el token en la sesión coincida con el token enviado en la URL.

3. **Validaciones**
    - Se comprueba que el ID de la transacción y el token no sean nulos y que coincidan con los valores esperados.
    - Se redirige a la página de compras si alguna validación falla.
4. **Consultas a la Base de Datos**
    - Se realiza una consulta para obtener los detalles de la compra basados en el ID de la transacción.
    - Se consulta la tabla detalle_compra para obtener información sobre los productos comprados.
5. **Visualización**
    - Se muestra la información de la compra, incluyendo fecha, ID de transacción y total pagado.
    - Se presenta una tabla con los detalles de los productos: nombre, precio, cantidad y subtotal.
    - Se utiliza formato de moneda para los valores numéricos.
6. **Formato de Fecha**
    - Se utiliza la clase DateTime para dar formato a la fecha de la compra según el estilo deseado.

16. Panel de Administración

Objetiv: Implementar un panel de administración para organizar el proyecto y facilitar la gestión.

Estructura del Proyecto

- **Problema:** La raíz del proyecto puede volverse desorganizada con la adición de más archivos.
- **Solución Propuesta:** Crear una carpeta admin para mantener separados los archivos de administración del resto del proyecto.

Creación de la Carpeta de Administración

- **Paso 1:** Crear una carpeta llamada admin en la raíz del proyecto.
- **Paso 2:** Dentro de admin, crear archivos como inicio.php para el panel de administración y index.php para el login.

Diseño del Panel de Administración

- **Elección de Plantilla:** Utilizar la plantilla SB Admin, disponible en dos versiones (Bootstrap 4 y Bootstrap 5).
- **Descarga y Configuración:**
    - Descargar y extraer la plantilla.
    - Copiar el contenido necesario al archivo inicio.php.
    - Eliminar datos y gráficos no necesarios.

Personalización

- **Modificar Plantilla:** Limpiar y ajustar el diseño del dashboard, menú y otras secciones.
- **Reutilización de Recursos:** Utilizar los estilos y scripts necesarios desde el archivo descargado.

Estructuración del Código

- **Separación de Elementos Comunes:** Crear archivos header.php y footer.php para incluir en todas las vistas con include.
- **Configuración de la Base de Datos:** Crear una carpeta config para manejar la conexión a la base de datos y otras constantes.

17. Login de Administración

La creación del inicio de sesión para los administradores.

1. Preparación del Entorno

- **Base de Datos:** Se crea una tabla admin con 9 columnas:
  - ID (clave primaria)
  - usuario (varchar 30)
  - password (varchar 120)
  - nombre (varchar 100)
  - email (varchar 50)
  - token_password (varchar 40, opcional)
  - password_request (int, predefinido 0)
  - activo (tinyint)
  - fecha_alta (datetime)
- **Comparación con la Tabla de Usuarios:** La estructura es similar, pero se añaden columnas específicas para administración.

2. Diseño del Formulario de Inicio de Sesión

- **Archivos:** Se reutiliza el diseño del login de clientes y se adapta al nuevo archivo index.php dentro de la carpeta admin.
- **Formulario:**
  - El formulario se ajusta para enviar datos al mismo archivo index.php usando el método POST.
  - Se modifican campos para usuario y password, eliminando la opción de registrar.
  - Se ajusta el botón de submit y se aseguran las validaciones básicas en el frontend.

3. Implementación de la Lógica de Inicio de Sesión

- **Configuración de Archivos PHP:**
  - Se crea admin_funciones.php para manejar la lógica de inicio de sesión, similar a las funciones del cliente.
  - Se configura la conexión a la base de datos y se realiza una inserción inicial de un usuario administrador.
- **Validaciones y Autenticación:**
  - Se valida que los campos usuario y password no estén vacíos.
  - Se verifica la existencia del usuario y la validez del password usando password_hash.
  - Se crean variables de sesión al iniciar sesión exitosamente y se redirige al panel de administración.

4. Errores y Mensajes

- **Manejo de Errores:** Se implementa la función mostrar_mensajes para mostrar errores en el formulario, como campos vacíos o credenciales incorrectas.
- **Pruebas:** Se realizan pruebas para asegurar que el formulario muestre mensajes de error apropiados y redirija correctamente al dashboard tras un inicio de sesión exitoso.