Proyecto_fin_curso_2

Portada

Nombre del Proyecto: Proyecto_fin_curso_2
Autor: JuanGR
Fecha: Febrero 2025

Índice

Descripción por encima de las Carpetas y archivos

- **assets/**: Contiene los archivos estáticos como CSS, JavaScript e imágenes.
- **autoload.php**: Archivo de autoload para cargar automáticamente las clases del proyecto.
- **config.php**: Archivo de configuración principal del proyecto.
- **controllers/**: Contiene los controladores de la aplicación.
- **database/**: Contiene los archivos relacionados con la base de datos.
- **helpers/**: Contiene funciones auxiliares que se utilizan en el proyecto.
- **index.php**: Archivo principal de entrada de la aplicación.
- **lib/**: Contiene librerías y clases auxiliares.
- **models/**: Contiene los modelos de datos de la aplicación.
- **views/**: Contiene las vistas de la aplicación.

Instalación

1. Clona el repositorio en tu máquina local.
2. Ejecuta `composer install` para instalar las dependencias.
3. Configura el archivo `.env` con tus credenciales de base de datos.
4. Ejecuta las migraciones de la base de datos.

Uso

1. Inicia el servidor web.
2. Accede a la aplicación a través de tu navegador web en la URL configurada.

Funcionalidades

- Gestión de usuarios.
- Gestión de productos.
- Gestión de categorias

- Carrito de la compra.
- Gestión de pedidos

Breve explicación de lo que hace cada archivo

1. assets

   1.1. css

   - Es el archivo el cual contiene todo el css que se utiliza en el resto de la página.

   1.2. img

   - Es una carpeta en la cual se contendrán las imagenes y la carpeta upload
   - En la carpeta upload se guardarán las imagenes que se guarden con el metodo save() de ProductoController, es decir cada vez que se cree o se modifique un producto

2. controllers

   2.1. CarritoController.php

   - Este controlador gestiona el carrito de compras mediante sesiones (`$_SESSION`).
   - Este archivo permite agregar, visualizar y eliminar productos del carrito de compras.
   - Funciones principales
   - **`index()`**: Muestra los productos en el carrito cargando `views/carrito/index.php`.
   - **`add()`**:
   - Obtiene el `id` del producto de `$_GET['id']`.
   - Si el producto ya está en el carrito, incrementa su cantidad.
   - Si no, lo obtiene de la base de datos y lo agrega.
   - Redirige a la vista del carrito.
   - **`remove()`**: Debería eliminar un producto específico.
   - **`delete_toditos()`**: Vacía el carrito eliminando `$_SESSION['carrito']` y redirige.

   2.2. CategoriaController.php

   Este controlador gestiona las categorías de productos y sus vistas.

   - Funciones principales:
   - **`index()`**: Muestra la lista de categorías.
   - **`crear()`**: Carga el formulario para crear una nueva categoría.
   - **`eliminar()`**: Carga la vista para eliminar una categoría.

- `save()`: Guarda una nueva categoría en la base de datos.
- `delete()`: Elimina una categoría existente.
- `ver()`: Muestra los detalles de una categoría y sus productos asociados.

Todas las funciones verifican si el usuario es administrador mediante `Utils::isAdmin()`.

2.3. ErrorController.php

Este controlador maneja los errores de la aplicación.

- Función principal:
- `index()`: Muestra un mensaje indicando que la página solicitada no existe.

Se utiliza cuando un usuario intenta acceder a una ruta no válida.

2.4. PedidoController.php

Maneja las operaciones relacionadas con la creación, visualización y gestión de pedidos. El controlador interactúa con el modelo Pedido para realizar las operaciones de base de datos y con el modelo Producto para obtener los productos en un pedido. Además, valida la entrada de datos y redirige al usuario según el estado de sus pedidos.

- Funciones principales
- `hacer()`: Muestra la vista para realizar un pedido (views/pedido/hacer.php).
- `add()`: Recibe la información del formulario de un nuevo pedido, valida los datos de entrada y guarda el pedido en la base de datos. También maneja la creación de las líneas de pedido (productos) y redirige al usuario a una página de confirmación.
- `confirmado()`: Muestra una vista de confirmación del pedido (views/pedido/confirmado.php) con los detalles del pedido y los productos asociados. -`mis_pedidos()` : Muestra todos los pedidos de un usuario autenticado, obteniendo los datos de la base de datos.
- `detalle()`: Muestra los detalles de un pedido específico, incluyendo los productos asociados, si se proporciona un ID de pedido.
- `gestion()`: Función administrativa que permite gestionar todos los pedidos, solo accesible para administradores.
- `estado_pedidos()`: Actualiza el estado de un pedido específico, solo accesible para administradores.

2.5. ProductoController.php

Este controlador gestiona las acciones relacionadas con los productos.

- Funciones principales:
- `index()`: Muestra una lista de productos destacados.
- `ver()`: Muestra los detalles de un producto específico.
- `gestion()`: Permite la gestión de productos (solo administradores).
- `crear()`: Muestra el formulario para añadir un nuevo producto.

- `save()`: Guarda o actualiza un producto con validaciones de datos e imagen.
- `editar()`: Muestra el formulario de edición de un producto existente.
- `eliminar()`: Elimina un producto de la base de datos.

Este controlador se encarga de la administración y visualización de productos en la tienda.

2.6. UsuarioController.php

Este controlador gestiona las acciones relacionadas con los usuarios.

Funciones principales:

- `index()`: Muestra un mensaje de prueba.
- `registro()`: Carga la vista de registro de usuarios.
- `sesion()`: Carga la vista de inicio de sesión.
- `listado()`: Muestra una lista de usuarios registrados.
- `save()`: Registra un nuevo usuario con validaciones estrictas.
- `login()`: Autentica a los usuarios y maneja la sesión.
- `logout()`: Cierra la sesión del usuario y elimina cookies.
- `editar()`: Permite modificar los datos de un usuario.
- `update()`: Actualiza los datos de un usuario en la base de datos.

Este controlador maneja la autenticación, gestión de usuarios y edición de perfiles con roles diferenciados (usuario/admin).

3. database

3.1. database.sql

- Muestra las consultas SQL necesarias para crear la base de datos de la pagina web

4. helpers

4.1. utils

Este archivo define la clase `Utils`, que proporciona métodos auxiliares para la gestión de sesiones, administración y visualización de categorías y carrito de compras.

- Funciones principales:
- `cerrarSesion($nombre)`: Elimina la variable de sesión especificada.
- `isAdmin()`: Verifica si el usuario es administrador y redirige si no lo es.
- `mostrar_categorias()`: Recupera y devuelve la lista de categorías desde el modelo `Categoria`.
- `Carrito_mostrar()`: Calcula y devuelve la cantidad de productos y el total del carrito.
- `Sesion_iniciada()`: Verifica si hay un usuario autenticado, de lo contrario, redirige a la página de inicio de sesión.

Esta clase facilita el manejo de sesiones, permisos y datos esenciales dentro de la aplicación.

5. lib

5.1. BaseDatos.php

Este archivo define la clase `BaseDatos`, que gestiona la conexión a la base de datos utilizando `PDO`.

- Funciones principales:
- `__construct()`: Inicializa los parámetros de conexión (`host`, `dbname`, `user`, `pass`) utilizando constantes definidas en `config.php`.
- `conectar_datos()`: Establece la conexión con la base de datos MySQL y configura el modo de error en `PDO::ERRMODE_EXCEPTION` para manejar errores de manera adecuada.
- `getConnection()`: Devuelve la conexión establecida con la base de datos para ser utilizada en otras clases.

Esta clase proporciona una forma centralizada y reutilizable de manejar la conexión a la base de datos dentro de la aplicación, además se utiliza en los archivos de la carpeta model para modificar la base de datos.

6. models

6.1. categoria.php

Este archivo define la clase `Categoria`, que representa las categorías en la base de datos.

- Propiedades:
- `$id`: Identificador único de la categoría.
- `$nombre`: Nombre de la categoría.
- `$db`: Instancia de la clase `BaseDatos` para gestionar la conexión.
- Métodos principales:
- `__construct()`: Inicializa la conexión a la base de datos mediante `BaseDatos`.
- `getId() / setId($id)`: Métodos para obtener y establecer el ID de la categoría.
- `getNombre() / setNombre($nombre)`: Métodos para obtener y establecer el nombre de la categoría.
- `getCategorias()`: Obtiene todas las categorías de la base de datos y las devuelve en orden descendente.
- `get_id_categorias()`: Obtiene una categoría específica según su ID y la devuelve como un objeto.
- `save()`: Inserta una nueva categoría en la base de datos y almacena su ID generado automáticamente.
- `delete()`: Elimina una categoría de la base de datos según su nombre.

Esta clase encapsula todas las operaciones necesarias para trabajar con categorías dentro de la aplicación.

6.2. producto.php

Este archivo define la clase Producto, que representa los productos en la base de datos. Permite realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre los productos y manejar las relaciones con las categorías.

- Propiedades:
- `$id`: Identificador único del producto.
- `$categoria_id`: Identificador de la categoría a la que pertenece el producto.
- `$nombre`: Nombre del producto.
- `$descripcion`: Descripción detallada del producto.
- `$precio`: Precio del producto.
- `$stock`: Cantidad disponible del producto.
- `$oferta`: Descuento u oferta aplicada al producto.
- `$fecha`: Fecha de creación del producto.
- `$imagen`: Nombre de la imagen asociada al producto (si existe).
- `$db`: Instancia de la clase BaseDatos, encargada de gestionar la conexión a la base de datos.
- Métodos principales:
- `__construct()`: Inicializa la conexión a la base de datos mediante la clase BaseDatos y establece la conexión.
- `getId() / setId($id)`: Métodos para obtener y establecer el ID del producto.
- `getCategoria_id() / setCategoria_id($categoria_id)`: Métodos para obtener y establecer el ID de la categoría del producto.
- `getNombre() / setNombre($nombre)`: Métodos para obtener y establecer el nombre del producto.
- `getDescripcion() / setDescripcion($descripcion)`: Métodos para obtener y establecer la descripción del producto.
- `getPrecio() / setPrecio($precio)`: Métodos para obtener y establecer el precio del producto.
- `getStock() / setStock($stock)`: Métodos para obtener y establecer la cantidad de stock del producto.

- `getOferta() / setOferta($oferta)`: Métodos para obtener y establecer la oferta del producto (si aplica).
- `getFecha() / setFecha($fecha)`: Métodos para obtener y establecer la fecha de creación del producto.
- `getImagen() / setImagen($imagen)`: Métodos para obtener y establecer la imagen del producto.
- `getProductos()`: Obtiene todos los productos de la base de datos, ordenados por ID en orden descendente.
- `getProductos_categoria()`: Obtiene los productos de una categoría específica, uniéndolos con la tabla categorias para obtener el nombre de la categoría. Los productos se ordenan por ID en orden descendente.
- `get_id_productos()`: Obtiene un producto específico por su ID y lo devuelve como un objeto.
- `getRandom($limit)`: Devuelve una cantidad limitada de productos aleatorios desde la base de datos.
- `save()`: Inserta un nuevo producto en la base de datos con todos sus atributos (categoría, nombre, descripción, precio, stock, oferta, imagen) y almacena su ID generado automáticamente.
- `edit()`: Actualiza los detalles de un producto existente en la base de datos. Si el producto tiene una imagen nueva, la actualiza también.
- `delete()`: Elimina un producto de la base de datos según su ID.

6.3. usuario.php

Este archivo define la clase Usuario, que gestiona las operaciones relacionadas con los usuarios en la base de datos, como el registro, inicio de sesión, actualización de datos y obtención de información sobre los usuarios.

- Propiedades:
- `$id`: Identificador único del usuario.
- `$nombre`: Nombre del usuario.
- `$apellidos`: Apellidos del usuario.
- `$email`: Correo electrónico del usuario, utilizado para la autenticación.
- `$password`: Contraseña del usuario, almacenada de manera segura.
- `$rol`: Rol del usuario (por ejemplo, "admin", "user").
- `$imagen`: Imagen asociada al usuario (opcional).
- `$db`: Instancia de la clase BaseDatos, responsable de gestionar la conexión a la base de datos.
- Métodos principales:

- `__construct()`: Inicializa la conexión a la base de datos utilizando la clase BaseDatos.
- `getId() / setId($id)`: Métodos para obtener y establecer el ID del usuario.
- `getNombre() / setNombre($nombre)`: Métodos para obtener y establecer el nombre del usuario.
- `getApellidos() / setApellidos($apellidos)`: Métodos para obtener y establecer los apellidos del usuario.
- `getEmail() / setEmail($email)`: Métodos para obtener y establecer el correo electrónico del usuario.
- `getPassword() / setPassword($password)`: Métodos para obtener y establecer la contraseña del usuario.
- `getRol() / setRol($rol)`: Métodos para obtener y establecer el rol del usuario.
- `getImagen() / setImagen($imagen)`: Métodos para obtener y establecer la imagen del usuario.
- `save()`: Inserta un nuevo usuario en la base de datos utilizando los datos del objeto actual (nombre, apellidos, email, password, rol) y almacena el ID generado automáticamente. La contraseña es almacenada de forma segura utilizando password_hash().
- `login()`: Verifica si un usuario existe con el correo electrónico proporcionado, compara la contraseña ingresada con la almacenada en la base de datos mediante password_verify(), y devuelve el objeto del usuario si la autenticación es correcta. Si la autenticación falla, devuelve false.
- `getUsuarios()`: Obtiene todos los usuarios de la base de datos, ordenados por ID de forma ascendente, y los devuelve como un array de objetos.
- `get_id_editar()`: Recupera los detalles de un usuario específico según su ID, utilizado generalmente para la edición de un usuario.
- `update()`: Actualiza los datos de un usuario específico en la base de datos (nombre, apellidos, email, password, rol) mediante su ID. Este método es útil para modificar la información de un usuario existente.

6.4. pedidos.php

Este archivo define la clase Pedido, que gestiona las operaciones relacionadas con los pedidos de los usuarios, como la creación de pedidos, la obtención de pedidos, la actualización de su estado y la recuperación de productos asociados a un pedido. La clase también maneja la inserción de productos en las líneas de pedidos.

- Propiedades:
- `$id`: Identificador único del pedido.
- `$usuario_id`: Identificador del usuario que realiza el pedido.

- `$provincia`: Provincia de envío del pedido.
- `$localidad`: Localidad de envío del pedido.
- `$direccion`: Dirección de envío del pedido.
- `$coste`: Coste total del pedido.
- `$estado`: Estado actual del pedido (por ejemplo, 'confirmado').
- `$fecha`: Fecha en la que se realiza el pedido.
- `$hora`: Hora en la que se realiza el pedido.
- `$db`: Instancia de la clase BaseDatos, responsable de gestionar la conexión a la base de datos.

Métodos principales:

- `__construct()`: Inicializa la conexión a la base de datos utilizando la clase BaseDatos. En este caso, el constructor no realiza nada dentro de la clase (aunque la conexión es manejada en los métodos individuales).
- `getId() / setId($id)`: Métodos para obtener y establecer el ID del pedido.
- `getUsuario_id() / setUsuario_id($usuario_id)`: Métodos para obtener y establecer el ID del usuario que realiza el pedido.
- `getProvincia() / setProvincia($provincia)`: Métodos para obtener y establecer la provincia de envío.
- `getLocalidad() / setLocalidad($localidad)`: Métodos para obtener y establecer la localidad de envío.
- `getDireccion() / setDireccion($direccion)`: Métodos para obtener y establecer la dirección de envío.
- `getCoste() / setCoste($coste)`: Métodos para obtener y establecer el coste total del pedido.
- `getEstado() / setEstado($estado)`: Métodos para obtener y establecer el estado del pedido.
- `getFecha() / setFecha($fecha)`: Métodos para obtener y establecer la fecha del pedido.
- `getHora() / setHora($hora)`: Métodos para obtener y establecer la hora del pedido.
- `getPedidos()`: Recupera todos los pedidos de la base de datos, ordenados por ID de forma descendente.
- `get_id_pedidos()`: Recupera un pedido específico mediante su ID.
- `getPedidosByUser()`: Recupera el último pedido realizado por un usuario específico, ordenado por ID de forma descendente.

- `get_todos_pedidos()`: Recupera todos los pedidos de un usuario específico, ordenados por ID de forma descendente.
- `get_Productos_Pedido($id)`: Recupera todos los productos asociados a un pedido específico. Utiliza una consulta JOIN para obtener los productos del pedido en función de su relación con la tabla lineas_pedidos.
- `save()`: Guarda un nuevo pedido en la base de datos utilizando los datos del objeto actual (usuario_id, provincia, localidad, dirección, coste, estado, fecha, hora). Al insertar el pedido, se genera automáticamente el ID del nuevo pedido y se devuelve.
- `save_linea($pedido_id)`: Guarda las líneas del pedido en la tabla lineas_pedidos asociada al pedido. Recibe el ID del pedido como argumento y, por cada producto en el carrito del usuario, inserta una línea de pedido correspondiente.
- `edit()`: Actualiza el estado de un pedido en la base de datos. Este método es útil para cambiar el estado de un pedido (por ejemplo, de 'confirmado' a 'enviado').

7. views

7.1. carrito

- Contiene un index en el que se listan todos los productos que se han añadido al carrito

7.2. categoria

- Contiene un index el cual contiene dos botones, uno para crear una categoria y otro para borrarla además en el index se listaran todas las categorias existentes en ese momento
- Los botones te llevarán a distintas páginas las cuales pediran el nombre de la categoria ya sea para crearla o eliminarla
- Además en header.php se mostrarán las categorias en el menú gracias a la ayuda de un metodo del archivo utils.
- Cuando se pinche en alguna de las categorias se las llevará al archivo ver.php el cual dependiendo del id que se haya puesto en la url mostrará o una imagen por defecto o la imagen asiganda al producto

7.3. layout

- Contiene los archivos php que representarán las vistas del header, footer y la barra lateral para que puedan ser llamados en el index posteriormente

7.4. producto

- El sidebar contendrá un enlace que llevará a gestion.php el cual es un archivo en el que se listaran los productos que existen actualmente
- En cada producto te darán la opcion de eliminarlos o editarlos
- También te darán la opcion de crear un nuevo producto por medio de un boton que te llevará al archivo crear.php en el cual dependiendo de si el producto ya existe o no se utilizara como un formulario para editar o crear un nuevo producto
- Por último, estara el archivo destacados en donde se mostrarán todos los productos existentes mediante un bucle y este archivo sera el archivo por defecto que se muestre en el index

7.5. usuario

- Contiene las vistas relacionadas con la gestión de usuarios.

7.5.1. sesion.php

- Muestra el formulario de inicio de sesión para los usuarios.

7.5.2. registro.php

- Muestra el formulario de registro para nuevos usuarios.

7.5.3. listado.php

- Muestra una lista de todos los usuarios registrados en el sistema (solo accesible para administradores).
- En el sidebar se da un enlace que llevara a este archivo y en cada usuario si eres admin se dara acceso para editar cada usuario sino eres admin solo podrás modificarte a ti mismo

7.5.4. modificar.php

- Se dara un formulario con los datos actuales del usuario y tu solo tendrás que cambiarlos para que se guarden

7.6. Pedido

- confirmado.php:
  - Muestra un mensaje de confirmación de pedido y detalles sobre el pedido (número, coste, productos).
  - Si el pedido no se completó, muestra un mensaje de error.
- detalle.php:
  - Muestra los detalles de un pedido específico, incluyendo la dirección de envío, estado del pedido y lista de productos.
  - Si el usuario es administrador, permite cambiar el estado del pedido.
- hacer.php:
  - Permite al usuario realizar un nuevo pedido, solicitando información como la provincia, localidad y dirección de envío.
  - Muestra un mensaje de éxito o error según el estado del pedido en la sesión.
  - Requiere que el usuario esté logueado para hacer un pedido.
- pedidos.php:
  - Muestra una lista de pedidos, ya sea para que el usuario vea los suyos o para que un administrador los gestione.
  - Muestra el número de pedido, coste, fecha y estado de cada uno, con enlaces a los detalles del pedido.

8. Archivos en la raiz del proyecto

8.1. autoload.php

Este archivo define una función para cargar automáticamente las clases cuando se necesitan.

- Función principal:
  - controllers_autoload($clase): Convierte el nombre de la clase en una ruta de archivo. Si el archivo existe, lo incluye; si no, genera un error de autoload.
- Registro de autoload:
  - spl_autoload_register('controllers_autoload'): Registra la función controllers_autoload para que PHP cargue las clases automáticamente cuando se requieran.

8.2. config.php

- En config.php se guardarán las constantes necesarias para conectar con nuestra base de datos.
- Constantes definidas:
  - DB_HOST: Dirección del host de la base de datos (localhost).
  - DB_NAME: Nombre de la base de datos (tienda).
  - DB_USER: Usuario para acceder a la base de datos (root).
  - DB_PASSWORD: Contraseña para el acceso a la base de datos (vacío en este caso).
  - base_url: URL base de la aplicación, utilizada para generar enlaces y redirecciones.
  - controller_default: Controlador predeterminado de la aplicación (productoController).
  - action_default: Acción predeterminada que se ejecuta (index).
- Resumen:
  - Este archivo contiene las configuraciones clave para la conexión a la base de datos y las rutas predeterminadas dentro de la aplicación.

8.3. index.php

Este archivo es el punto de entrada principal de la aplicación, donde se gestionan las sesiones, las solicitudes de controladores y acciones, y la visualización de las vistas.

- Flujo de Ejecución:
  - Inicio de sesión y restauración de sesión:
    - Inicia una sesión con session_start().
    - Si no existe una sesión activa, pero sí una cookie user_login, se intenta restaurar la sesión mediante el ID almacenado en la cookie. Si el usuario existe en la base de datos, se guarda en $_SESSION['identidad'] y, si es un administrador, se guarda $_SESSION['admin'] = true.

- Inclusión de Archivos:
    - Se incluyen los archivos necesarios: autoload.php (para autoload de clases), config.php (configuraciones generales), y helpers/utils.php (funciones auxiliares). Se incluyen también las vistas principales del diseño: header.php y sidebar.php. Función de manejo de errores:
    - Si no se encuentra el controlador o la acción especificada, se llama al método show_error() que instancia el errorController y muestra la página de error.
- Controlador y Acción:
    - Verifica si se ha especificado un controlador y acción en la URL (a través de $_GET['controller'] y $_GET['action']).
    - Si no se especifica controlador ni acción, se carga el controlador y la acción predeterminados definidos en config.php.
    - Verifica si la clase del controlador solicitado existe. Si es así, se instancia y ejecuta la acción correspondiente.
    - Si la clase o acción no existe, se muestra la página de error.
- Carga de pie de página:
    - Al final de la ejecución, se incluye footer.php, completando el ciclo de carga de la vista.

8.4. .htaccess

- En este archivo se incluye la configuración necesaria para poder utilizar rutas amigables

Tecnologías Utilizadas

- PHP
- MySQL
- Composer
- HTML
- CSS
- JavaScript