Actualización del Enunciado: Desarrollo de un CRUD con PHP, Docker, Composer y DotEnv

Título: Construcción de un Sistema CRUD con Registro y Login usando PHP, MySQL y Docker

---

Objetivo General:

Implementar un sistema CRUD (Crear, Leer, Actualizar, Borrar) para gestionar los datos de una tienda. Este sistema incluirá funcionalidad de **registro y login**, garantizando que en todo momento un usuario debe estar registrado para interactuar con la aplicación. El sistema deberá informar mediante mensajes las acciones realizadas y el usuario conectado deberá mostrarse en todo momento.

Detalles del Proyecto:

Funcionalidades Principales:

1. **Registro y Login en `index.php`**:
    - **Registro**:
        - Permite crear un nuevo usuario con un nombre único y contraseña.
        - La contraseña debe almacenarse de forma segura utilizando `password_hash()`.
    - **Login**:
        - Valida las credenciales del usuario contra los datos almacenados en la base de datos.
        - Si el usuario es autenticado correctamente, se almacena su información en la sesión.
    - En ambos casos, muestra mensajes claros al usuario sobre el éxito o fallo de la acción realizada.
2. **Persistencia del Usuario**:
    - En todo momento, si un usuario está registrado, debe mostrarse su nombre en la aplicación.
    - Permitir al usuario desconectarse con un botón `Logout`.
3. **Restricción de Acceso**:
    - Las páginas internas (`sitio.php`, `listado.php`, etc.) deben estar protegidas.
    - Si no hay un usuario en sesión, redirigir automáticamente a `index.php`.
4. **CRUD**:
    - Crear, Leer, y Borrar registros en las tablas de la base de datos. Opcionalmente se podrá modificar datos, pero no se pide dicha funcionalidad
        - `productos`
        - `tiendas`
        - `usuarios`
        - `stock`

- Muestra un mensaje cada vez que se realiza una acción (p. ej., "Producto añadido correctamente", "Error al eliminar producto, puede ser que haya integridad referencial").

Requisitos Técnicos:

1. **Base de Datos**:
   - Usa la base de datos `tienda` configurada en el contenedor MySQL.
   - El archivo `datos.sql` inicializa las tablas con datos de ejemplo.
2. **Docker**:
   - Usa el archivo `docker-compose.yaml` proporcionado para levantar los contenedores:
     - `web` (PHP + Apache).
     - `mysql` (base de datos).
     - `phpmyadmin` (gestión visual de la base de datos).
3. **Composer y DotEnv**:
   - Usa Composer para instalar dependencias.
   - Usa la librería `vlucas/phpdotenv` para manejar los parámetros de conexión a la base de datos.
   - Los parámetros como `DB_HOST`, `DB_USER`, `DB_PASS`, y `DB_NAME` deben configurarse en un archivo `.env`.
4. **Sesiones**:
   - Utiliza sesiones PHP para manejar la autenticación del usuario y su estado en la aplicación.
5. **Mensajes Informativos**:
   - Implementa un sistema para mostrar mensajes en cada acción del usuario.
   - Ejemplos:
     - "Usuario registrado con éxito."
     - "Credenciales incorrectas."
     - "Producto eliminado correctamente."
6. **Uso de PDO o MySQLi**:
   - Se recomienda **PDO** por su flexibilidad, soporte para múltiples bases de datos y su manejo integrado de excepciones.
   - Implementa consultas con sentencias preparadas para evitar inyecciones SQL.

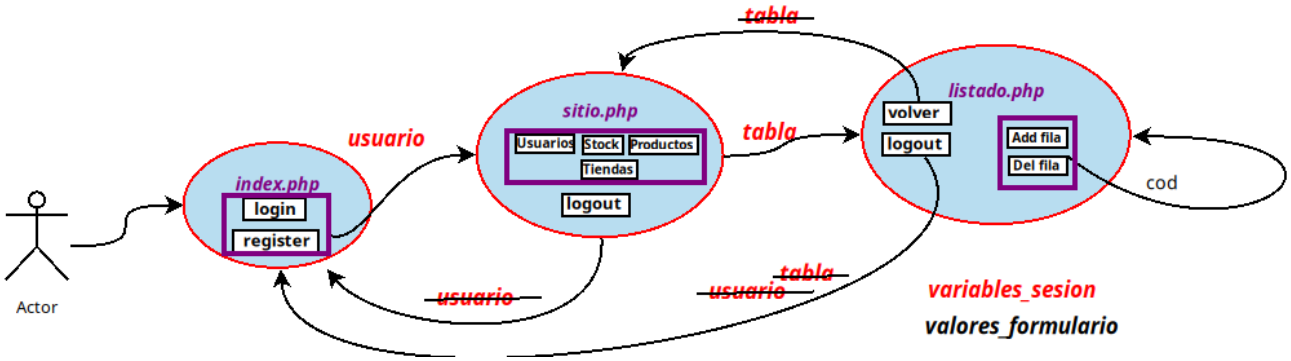
Estructura del Proyecto Actualizada:

```
proyecto/
├── app/
│   ├── Crud/
│   │   ├── DB.php        (Gestión de conexión y operaciones con la base de datos)
│   │   ├── Plantilla.php (Generación de tablas HTML dinámicas)
│   │   ├── Auth.php      (Gestión de login, registro y autenticación)
├── public/
│   ├── index.php         (Registro y login de usuarios)
│   ├── sitio.php         (Panel principal de gestión con opciones del CRUD)
│   ├── listado.php       (Vista de registros con opciones de edición y eliminación)
│   ├── add.php           (Formulario para añadir registros)
│   └── css/
├── docker-compose.yaml    (Configuración del entorno Docker)
├── mysql/
│   ├── datos.sql          (Inicialización de la base de datos)
├── vendor/                (Dependencias de Composer)
├── composer.json          (Archivo de configuración de Composer)
├── .env                   (Configuración de la base de datos)
├── README.md              (Instrucciones de instalación y uso)
```

Rúbrica de Corrección Actualizada

Criterio	Descripción
Configuración Docker	El entorno Docker está configurado y funcional.
Uso de Composer y DotEnv	Se usa Composer para instalar dependencias y DotEnv para manejar la configuración.
Autenticación con Sesiones	Login, registro, logout y persistencia del usuario funcionan correctamente.
Mensajes Informativos	Se muestran mensajes claros al usuario en todas las acciones realizadas.
Funcionalidad CRUD	Todas las operaciones CRUD funcionan correctamente en todas las tablas.
Uso de PDO	Se utiliza PDO para todas las operaciones con la base de datos con sentencias preparadas.
Estructura del Proyecto	El código está organizado en clases y carpetas, siguiendo la estructura propuesta.
Uso de clases correctas	
Implementación de los métodos	
Claridad y legibilidad	

Navegación de la aplicación



Los valores rojos representan posibles variables de sesión

Los valores negros representan valores de formularios