Cambios a Implementar en Archivos PHP para Usar Funciones SQL Refactorizadas

Este documento detalla los cambios necesarios en los archivos PHP para integrar las funciones SQL refactorizadas. El objetivo es eliminar la lógica duplicada y utilizar los nuevos procedimientos almacenados de manera eficiente.

1. Archivos SQL Refactorizados Clave (Recordatorio)

Las principales refactorizaciones se centraron en:

- **`eliminar_logicamente(p_tabla, p_id, p_id_usuario)`** : Nueva función unificada para eliminación lógica de registros en `archivos` , `carpetas` y `solicitudes_compartidos` .
- **`cambiar_estado_archivo_compartido(p_estado, p_id_detalle)`** : Función consolidada para cambiar el estado de un archivo compartido.
- **`obtener_estadisticas_compartidos_globales()`** y **`obtener_estadisticas_compartidos_usuario(p_correo_usuario)`** : Funciones consolidadas para estadísticas de compartidos.
- **`registrar_usuario(...)`** y **`modificar_usuario(...)`** : Procedimientos mejorados para la gestión de usuarios, incluyendo validaciones de unicidad.
- **`cambiar_estado_usuario(p_id_usuario)`** : Función para activar/desactivar usuarios.
- **`cambiar_clave_usuario(p_id_usuario, p_nueva_clave)`** : Función para cambiar la contraseña de un usuario.
- **`actualizar_avatar_usuario(p_id_usuario, p_avatar_url)`** : Función para actualizar el avatar.

2. Cambios por Archivo PHP

2.1. `Admin.php`

No se requieren cambios directos en `Admin.php` ya que las llamadas a las funciones del modelo (`AdminModel`) se mantienen, y los cambios se realizarán en el modelo.

2.2. `AdminModel.php`

Este archivo requiere ajustes para usar la nueva función `eliminar_logicamente` y para asegurar que las llamadas a `getVerificar` sean consistentes con la nueva lógica.

- **Función `delete($id)` (para carpetas):**
  - **Cambio:** Reemplazar la llamada `UPDATE carpetas SET estado = ?, elimina = ? WHERE id = ?` por la nueva función `eliminar_logicamente` .
  - **Código Anterior:**

- **Código Nuevo:**
- **Nota:** El controlador `Admin.php` en su función `eliminarCarpeta` ya llama a `$this->model->eliminarCarpeta($id);`. Deberá modificarse para pasar el `$this->id_usuario` si no lo hace ya. La función `delete` en `AdminModel` debe ser renombrada a `eliminarCarpeta` para mayor claridad y consistencia con la nueva función SQL.

- **Función `getEstadisticasGlobales()`:**
  - **Cambio:** Consolidar las consultas de estadísticas en una sola llamada al nuevo procedimiento almacenado `obtener_estadisticas_globales()`.
  - **Código Anterior (fragmento):**
  - **Código Nuevo:**
- **Función `getTiposArchivos()`:**
  - **Cambio:** Utilizar la nueva función `obtener_total_archivos_por_tipo()`.
  - **Código Anterior:**
  - **Código Nuevo:**
- **Función `getUsuariosActivos()`:**
  - **Cambio:** Utilizar la nueva función `obtener_usuarios_mas_activos()`.
  - **Código Anterior:**
  - **Código Nuevo:**
- **Función `getActividadReciente()`:**
  - **Cambio:** Utilizar la nueva función `obtener_actividad_reciente()`.
  - **Código Anterior:**
  - **Código Nuevo:**
- **Función `getArchivosCompartidosRecientes()`:**
  - **Cambio:** Utilizar la nueva función `obtener_archivos_compartidos_recientes()`.
  - **Código Anterior:**
  - **Código Nuevo:**

2.3. `Archivos.php`

- **Función `eliminar($id)`:**
  - **Cambio:** Llamar a la función `eliminar_logicamente` en el modelo.
  - **Código Anterior:**
  - **Código Nuevo:**

- **Función `eliminarCompartido($id)`:**
  - **Cambio:** Llamar a la función `eliminar_logicamente` en el modelo para `solicitudes_compartidos`.
  - **Código Anterior:**
  - **Código Nuevo:**
- **Función `eliminarCarpeta($id)`:**
  - **Cambio:** Llamar a la función `eliminar_logicamente` en el modelo.
  - **Código Anterior:**
  - **Código Nuevo:**

2.4. `ArchivosModel.php`

Este archivo es crucial para la integración de la función `eliminar_logicamente`.

- **Eliminar funciones `eliminar($fecha, $id)` y `eliminarCompartido($fecha, $id)`:**
  - Estas funciones se vuelven redundantes y deben ser eliminadas.
- **Nueva función `eliminarLogicamente($tabla, $id, $id_usuario)`:**
  - **Cambio:** Crear una nueva función en `ArchivosModel` que encapsule la llamada al procedimiento `eliminar_logicamente`.
  - **Código Nuevo:**
- **Función `eliminarCarpeta($id, $id_usuario)`:**
  - **Cambio:** Reemplazar la llamada al procedimiento `eliminar_carpeta` por la nueva función `eliminar_logicamente`.
  - **Código Anterior:**
  - **Código Nuevo:**
- **Función `eliminarCarpetaPermanente($id, $id_usuario = null)`:**
  - **Cambio:** Asegurarse de que el procedimiento `eliminar_carpeta_permanente` se use correctamente.
  - **Nota:** Esta función ya usa un SP, pero es importante verificar que el SP `eliminar_carpeta_permanente` ahora maneje la eliminación recursiva de archivos y subcarpetas, y la eliminación física de los archivos asociados.
- **Función `eliminarArchivoPermanente($id, $id_usuario = null)`:**
  - **Cambio:** Asegurarse de que el procedimiento `eliminar_archivo_permanente` se use correctamente y que la eliminación física se realice solo si el SP indica éxito.
  - **Código Anterior (fragmento):**

- **Código Nuevo (ajuste):**
- **Nota:** Si el SP `eliminar_archivo_permanente` no devuelve la ruta física, la lógica de obtener la ruta antes de llamar al SP debe mantenerse.

2.5. `Compartidos.php`

- **Función `eliminar($id)`:**
  - **Cambio:** La lógica de eliminación de archivos compartidos ahora se maneja por `eliminar_logicamente` en el modelo.
  - **Código Anterior:**
  - **Código Nuevo:**

2.6. `CompartidosModel.php`

- **Función `cambiarEstado($estado, $id_detalle)`:**
  - **Cambio:** Esta función debe llamar al nuevo procedimiento `cambiar_estado_archivo_compartido`.
  - **Código Anterior:**
  - **Nota:** La función `cambiar_estado_archivo_compartido` ya fue refactorizada en SQL, por lo que el PHP ya debería estar llamándola correctamente. Solo verificar que el SP `cambiar_estado_archivo_compartido` ahora también maneje la eliminación lógica (estado = 0) si es el caso.
- **Función `marcarComoVisto($id_detalle)`:**
  - **Cambio:** Esta función debe llamar al nuevo procedimiento `marcar_archivo_como_visto`.
  - **Código Anterior:**
  - **Nota:** Similar a `cambiarEstado`, esta función ya debería estar llamando al SP correcto.
- **Función `getEstadisticasCompartidos($correo_usuario = null)`:**
  - **Cambio:** Consolidar las llamadas a los nuevos procedimientos `obtener_estadisticas_compartidos_usuario` y `obtener_estadisticas_compartidos_globales`.
  - **Código Anterior:**
  - **Nota:** La implementación actual ya es correcta si los SPs `obtener_estadisticas_compartidos_usuario` y `obtener_estadisticas_compartidos_globales` fueron creados como se indicó en la refactorización SQL.
- **Nueva función `eliminarLogicamente($tabla, $id, $id_usuario)`:**

- **Cambio:** Añadir esta función para que el controlador `Compartidos.php` pueda usarla.
- **Código Nuevo:**

2.7. `Principal.php`

- **Función `validar()`:**
  - **Cambio:** No se requieren cambios directos en esta función, ya que `PrincipalModel->getUsuario($correo)` ya debería devolver la `fecha_ultimo_cambio_clave` correctamente.
- **Función `cambiarClaveInicial()`:**
  - **Cambio:** No se requieren cambios directos en esta función, ya que `PrincipalModel->cambiarClaveInicial()` ya debería estar llamando al SP `cambiar_clave_inicial_usuario`.
- **Función `registrar()`:**
  - **Cambio:** La llamada a `registrarSolicitud` en el modelo ya no necesita manejar la generación de contraseña temporal en PHP, ya que el SP `registrar_solicitud_usuario` la maneja internamente.
  - **Código Anterior (fragmento):**
  - **Nota:** El código ya está adaptado para la nueva lógica del SP, solo se debe asegurar que el SP `registrar_solicitud_usuario` no devuelva la contraseña temporal al PHP.

2.8. `PrincipalModel.php`

- **Función `registrarSolicitud($nombre, $apellido, $correo, $telefono, $direccion)`:**
  - **Cambio:** Asegurarse de que esta función llame al SP `registrar_solicitud_usuario` y maneje su respuesta correctamente.
  - **Código Actual (ya refactorizado):**
  - **Nota:** Este código ya está correcto si el SP `registrar_solicitud_usuario` devuelve un array con `success`, `mensaje` e `id_solicitud`.
- **Función `cambiarClaveInicial($id_usuario, $nueva_clave)`:**
  - **Cambio:** Asegurarse de que esta función llame al SP `cambiar_clave_inicial_usuario` y maneje su respuesta correctamente.
  - **Código Actual (ya refactorizado):**
  - **Nota:** Este código ya está correcto.

2.9. `Recuperar.php`

- **Función `solicitar()`:**

- **Cambio:** No se requieren cambios directos en esta función, ya que `RecuperarModel->invalidarTokensUsuario()` y `RecuperarModel->guardarToken()` ya deberían estar llamando a los SPs correctos.

- **Función `procesar()`:**
  - **Cambio:** No se requieren cambios directos en esta función, ya que `RecuperarModel->validarToken()`, `RecuperarModel->actualizarContrasena()` y `RecuperarModel->marcarTokenUsado()` ya deberían estar llamando a los SPs correctos.

2.10. `RecuperarModel.php`

- **Función `getUsuario($correo)` y `getUsuarioPorId($id)`:**
  - **Cambio:** Asegurarse de que estas funciones llamen a los SPs `obtener_usuario_por_correo` y `obtener_usuario_por_id` respectivamente.
  - **Código Anterior (fragmento `getUsuario`):**
  - **Código Nuevo (para `getUsuario`):**
  - **Código Nuevo (para `getUsuarioPorId`):**
- **Función `actualizarContrasena($idUsuario, $hashClave)`:**
  - **Cambio:** Asegurarse de que esta función llame al SP `cambiar_clave_usuario`.
  - **Código Anterior:**
  - **Código Nuevo:**

2.11. `Usuarios.php`

- **Función `guardar()`:**
  - **Cambio:** La lógica de validación de correo y teléfono debe basarse en la respuesta del SP `verificar_campo_usuario`.
  - **Código Anterior (fragmento):**
  - **Código Nuevo (fragmento para `guardar` - Nuevo Usuario):**
- **Función `delete($id)`:**
  - **Cambio:** Llamar a la función `cambiar_estado_usuario` en el modelo.
  - **Código Anterior:**
  - **Código Nuevo:**
- **Función `actualizarPerfil()`:**
  - **Cambio:** La validación de correo y teléfono debe basarse en la respuesta del SP `modificar_usuario` (o `actualizar_perfil_usuario` si se mantiene separado).

    - **Código Anterior (fragmento):**
    - **Código Nuevo (fragmento):**
- **Función `cambiarClave()`:**
    - **Cambio:** Llamar a la función `cambiar_clave_usuario` en el modelo.
    - **Código Anterior:**
    - **Código Nuevo:**
- **Función `cambiarAvatar()`:**
    - **Cambio:** Llamar a la función `actualizar_avatar_usuario` en el modelo.
    - **Código Anterior (fragmento):**
    - **Código Nuevo (fragmento):**

2.12. `UsuariosModel.php`

Este archivo es el que más cambios requiere para integrar los nuevos procedimientos almacenados de gestión de usuarios.

- **Función `getVerificar($item, $nombre, $id)`:**
    - **Cambio:** Llamar al SP `verificar_campo_usuario` y devolver su resultado completo.
    - **Código Anterior:**
    - **Código Nuevo:**
- **Función `registrar($nombre, $apellido, $correo, $telefono, $direccion, $clave, $rol)`:**
    - **Cambio:** Llamar al SP `registrar_usuario` y devolver su resultado completo (success, mensaje, id_usuario_creado).
    - **Código Anterior (fragmento):**
    - **Código Nuevo:**
- **Función `modificar($nombre, $apellido, $correo, $telefono, $direccion, $rol, $id)`:**
    - **Cambio:** Llamar al SP `modificar_usuario` y devolver su resultado completo.
    - **Código Anterior (fragmento):**
    - **Código Nuevo:**
- **Función `delete($id)`:**
    - **Cambio:** Llamar al SP `cambiar_estado_usuario` y devolver su resultado completo.
    - **Código Anterior (fragmento):**
    - **Código Nuevo (renombrar a `cambiarEstadoUsuario` para claridad):**
- **Función `actualizarPerfil($nombre, $apellido, $correo, $telefono, $direccion, $id)`:**

    - **Cambio:** Llamar al SP `modificar_usuario` (o `actualizar_perfil_usuario` si se mantiene separado) y devolver su resultado completo.
    - **Código Anterior (fragmento):**
    - **Código Nuevo (asumiendo que `modificar_usuario` se usa para esto):**
    - **Nota:** Si se mantiene `actualizar_perfil_usuario` como un SP separado, su llamada en el modelo debe ser similar a `registrar` y `modificar` para manejar el `success` y `mensaje`.
- **Función `cambiarClave($clave, $id)`:**
    - **Cambio:** Llamar al SP `cambiar_clave_usuario` y devolver su resultado completo.
    - **Código Anterior (fragmento):**
    - **Código Nuevo (renombrar a `cambiarClaveUsuario` para claridad):**
- **Función `actualizarAvatar($id, $ruta_avatar)`:**
    - **Cambio:** Llamar al SP `actualizar_avatar_usuario` y devolver su resultado completo.
    - **Código Anterior (fragmento):**
    - **Código Nuevo (renombrar a `actualizarAvatarUsuario` para claridad):**

3. Consideraciones Adicionales

- **Manejo de Errores:** Asegúrate de que todas las llamadas a `$this->model->` manejen las excepciones o los valores de retorno de `success` y `mensaje` de los nuevos procedimientos almacenados. Los ejemplos anteriores ya incorporan esto.
- **`Query.php`:** La clase `Query.php` (asumiendo que es la clase base para los modelos) debe tener un método `select` y `selectAll` que puedan manejar los resultados de los procedimientos almacenados, incluyendo los campos `success` y `mensaje`.
- **Nombres de Funciones:** Se recomienda renombrar las funciones en los modelos PHP para que coincidan con los nombres de los procedimientos almacenados SQL cuando sea posible, para mantener la coherencia.
- **Validaciones en PHP:** Algunas validaciones de entrada (ej. `empty`, `is_numeric`, `filter_var`) pueden mantenerse en PHP antes de llamar a los procedimientos almacenados para reducir la carga en la base de datos y proporcionar retroalimentación más rápida al usuario. Sin embargo, las validaciones de lógica de negocio (ej. unicidad de correo/teléfono) deben delegarse a los procedimientos almacenados para asegurar la atomicidad y consistencia.
- **`AuthManager.php`:** Si `AuthManager` utiliza funciones SQL para obtener datos de usuario (como `getUsuario`), asegúrate de que también se actualice para usar los nuevos SPs si es necesario.

Al implementar estos cambios, se logrará una mayor modularidad, se reducirá la duplicación de código y se aprovechará la lógica de negocio centralizada en la base de datos.