Documentación Técnica : Gestor de Tareas
Fecha : 22 de mayo del 2025
Autor: Jorge Gael Rodriguez Ibarra

1. Introducción y Objetivos del Proyecto

El sistema "Gestor de Tareas" es una aplicación web diseñada para permitir a los usuarios registrarse, iniciar sesión y administrar sus tareas personales de manera eficiente. El objetivo principal es proporcionar una interfaz intuitiva para crear, visualizar, editar, marcar como completadas y archivar tareas, ayudando a los usuarios a mantenerse organizados.

Tecnologías Utilizadas:

- **Backend:** PHP
- **Base de Datos:** MySQL (interactuando mediante la extensión MySQLi de PHP)
- **Frontend:** HTML, CSS, JavaScript
- **Fuentes y Librerías Externas:**
    - Google Fonts (Poppins)
    - Font Awesome (para iconos

2. Arquitectura del Sistema y Estructura de Carpetas

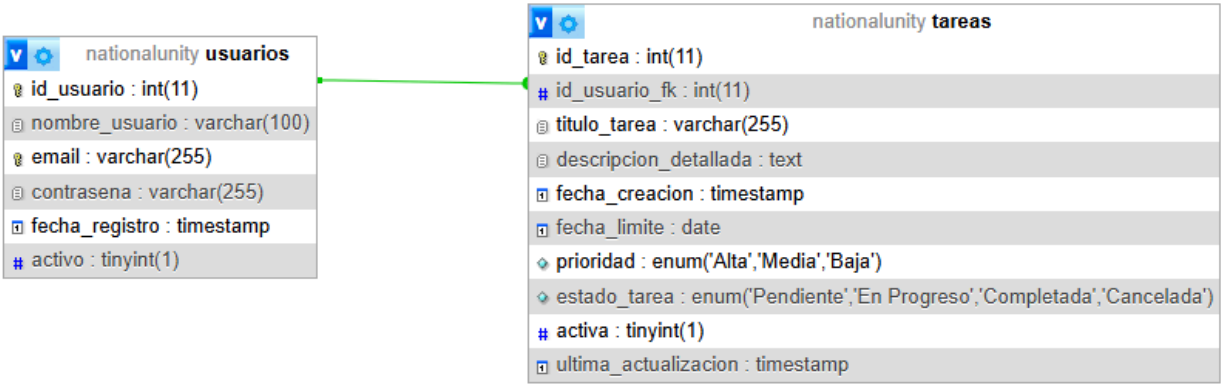
El proyecto sigue una estructura modular para separar las preocupaciones y facilitar el mantenimiento. La organización principal de carpetas es la siguiente:

- **`ACCIONES/`**: Contiene los scripts PHP que manejan la lógica de procesamiento de formularios y las interacciones con la base de datos que no son de simple visualización.
    - `Auth/`: Scripts relacionados con la autenticación de usuarios.
        - `procesar_login.php`: Valida credenciales y gestiona el inicio de sesión.
        - `procesar_registro.php`: Valida datos y registra nuevos usuarios.
    - `Tareas/`: Scripts para la lógica CRUD de las tareas.
        - `registrar_nueva_tarea.php`: Procesa la creación de nuevas tareas.
        - `actualizar_datos_tarea.php`: Procesa la edición de tareas existentes.
        - `marcar_completada.php`: Cambia el estado de una tarea a "Completada".
        - `archivar_tarea.php`: Realiza la eliminación lógica de una tarea (la marca como inactiva).
        - `obtener_tareas.php`: Script incluido para cargar las tareas del usuario (usado en `index.php`).

- **`CONFIG/`**: Archivos de configuración centralizados.
    - `db_connection.php`: Establece y configura la conexión a la base de datos MySQL utilizando MySQLi.
- **`CSS/`**: Hojas de estilo para la presentación visual.
    - `estilos_auth.css`: Estilos para las páginas de login y registro.
    - `estilos_main.css`: Estilos principales para el panel de tareas y otras páginas internas.
    - `estilos_forms_tareas.css`: Estilos específicos para los formularios de creación y edición de tareas.
- **`IMG/`**: Contiene imágenes estáticas utilizadas en la interfaz.
- **`INCLUDES/`**: Fragmentos de código PHP/HTML reutilizables.
    - `header.php`: Contiene la sección `<head>` del HTML, el encabezado de la página (logo, bienvenida, botón de logout) y el botón de modo oscuro.
    - `footer.php`: Contiene el cierre de etiquetas HTML y la inclusión del script JavaScript principal.
- **`JS/`**: Archivos JavaScript para la interactividad del frontend.
    - `Modo_oscuro.js`: Maneja la funcionalidad de cambio de tema (modo oscuro/claro).
    - Registro.js: Hace las validaciones correspondientes antes de enviar el formulario
- **Archivos PHP en la Raíz:** Páginas principales visibles por el usuario.
    - `index.php`: Panel principal donde se listan las tareas del usuario logueado.
    - `login.php`: Formulario de inicio de sesión.
    - `registro.php`: Formulario de registro de nuevos usuarios.
    - `logout.php`: Script para cerrar la sesión del usuario.
    - `crear_tarea.php`: Formulario para añadir nuevas tareas.
    - `editar_tarea.php`: Formulario para modificar tareas existentes.

3. Diseño de la Base de Datos

La aplicación utiliza una base de datos MySQL para almacenar la información de los usuarios y sus tareas.



- **Tabla `Usuarios`**
    - `id_usuario` (INT, PK, AI): Identificador único autoincremental del usuario.
    - `nombre_usuario` (VARCHAR(100), NOT NULL): Nombre completo o de visualización del usuario.
    - `email` (VARCHAR(255), NOT NULL, UNIQUE): Correo electrónico del usuario, utilizado para el inicio de sesión y debe ser único.
    - `contrasena` (VARCHAR(255), NOT NULL): Almacena la contraseña del usuario después de ser procesada por `password_hash()`.
    - `fecha_registro` (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP): Fecha y hora en que el usuario se registró.
    - `activo` (BOOLEAN/TINYINT(1), NOT NULL, DEFAULT 1): Indica si la cuenta del usuario está activa (1) o desactivada/eliminada lógicamente (0).
- **Tabla `Tareas`**
    - `id_tarea` (INT, PK, AI): Identificador único autoincremental de la tarea.
    - `id_usuario_fk` (INT, NOT NULL, FK): Clave foránea que referencia a `Usuarios(id_usuario)`. Indica a qué usuario pertenece la tarea.
    - `titulo_tarea` (VARCHAR(255), NOT NULL): Título o descripción corta de la tarea.
    - `descripcion_detallada` (TEXT, NULL): Descripción más extensa de la tarea (opcional).
    - `fecha_creacion` (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP): Fecha y hora en que se creó la tarea.

- `fecha_limite` (DATE, NULL): Fecha límite opcional para completar la tarea.
- `prioridad` (ENUM('Alta', 'Media', 'Baja'), NOT NULL, DEFAULT 'Media'): Nivel de prioridad de la tarea.
- `estado_tarea` (ENUM('Pendiente', 'En Progreso', 'Completada', 'Cancelada'), NOT NULL, DEFAULT 'Pendiente'): Estado actual de la tarea.
- `activa` (BOOLEAN/TINYINT(1), NOT NULL, DEFAULT 1): Indica si la tarea está activa (1) o archivada/eliminada lógicamente (0).
- `ultima_actualizacion` (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP): Se actualiza automáticamente cada vez que se modifica la fila.

Relación y Clave Foránea:

- Se define una restricción de clave foránea en `Tareas.id_usuario_fk` que referencia a `Usuarios.id_usuario`.
- `CONSTRAINT fk_tareas_usuarios FOREIGN KEY (id_usuario_fk) REFERENCES Usuarios(id_usuario) ON DELETE RESTRICT ON UPDATE CASCADE`
  - `ON DELETE RESTRICT`: Impide eliminar un usuario si tiene tareas asociadas.
  - `ON UPDATE CASCADE`: Si el `id_usuario` cambiara (improbable), se actualizaría en las tareas.

4. Lógica Principal de la Aplicación

- **Autenticación de Usuarios:**
  - **Registro (`registro.php` y `ACCIONES/Auth/procesar_registro.php`):**
    1. El usuario llena el formulario en `registro.php`.
    2. Los datos se envían a `procesar_registro.php`.
    3. Se realizan validaciones (campos vacíos, formato de email, longitud de contraseña, coincidencia de contraseñas).
    4. Se verifica si el email ya existe en la tabla `Usuarios` mediante una consulta preparada.
    5. Si el email no existe y las validaciones pasan, la contraseña se hashea usando `password_hash()`.
    6. Se inserta el nuevo usuario en la tabla `Usuarios` con una sentencia preparada. `activo` se establece a `TRUE`.
    7. Se manejan mensajes de éxito o error y se redirige al usuario. (Actualmente, el registro exitoso inicia sesión automáticamente y redirige a `index.php`).
  - **Inicio de Sesión (`login.php` y `ACCIONES/Auth/procesar_login.php`):**

1. El usuario llena el formulario en `login.php`.
2. Los datos se envían a `procesar_login.php`.
3. Se valida que los campos no estén vacíos y que el email tenga un formato válido.
4. Se consulta la tabla `Usuarios` por el email proporcionado usando una sentencia preparada.
5. Si se encuentra el usuario y la cuenta está `activo = TRUE`:
    - Se verifica la contraseña ingresada contra el `contrasena` almacenado usando `password_verify()`.
    - Si la contraseña es correcta, se establecen las variables de sesión `$_SESSION['id_usuario']` y `$_SESSION['nombre_usuario']`.
    - Se regenera el ID de sesión con `session_regenerate_id(true)`.
    - Se redirige a `index.php`.
6. Si las credenciales son incorrectas o el usuario no existe/está inactivo, se muestra un mensaje de error en `login.php`.

- **Manejo de Sesiones:**
    1. `session_start()` se llama al inicio de cada script que necesita acceso a variables de sesión.
    2. La variable `$_SESSION['id_usuario']` se usa para identificar al usuario logueado y proteger el acceso a las páginas.
- **Cerrar Sesión (`logout.php`):**
    1. Se inicia la sesión.
    2. Se vacía el array `$_SESSION`.
    3. Se destruyen las cookies de sesión (si se usan).
    4. Se destruye la sesión con `session_destroy()`.
    5. Se redirige a `login.php`.

- **Gestión de Tareas (CRUD):**

    - **Crear Tarea (`crear_tarea.php` y `ACCIONES/Tareas/registrar_nueva_tarea.php`):**
        1. El usuario accede a `crear_tarea.php` (requiere estar logueado).
        2. Llena el formulario (título, descripción opcional, fecha límite opcional, prioridad).
        3. Al enviar, los datos van a `registrar_nueva_tarea.php`.
        4. Se valida que el título no esté vacío y que la prioridad sea válida. Se valida formato de fecha.
        5. Se inserta la nueva tarea en la tabla `Tareas` asociada al `id_usuario_fk` del usuario en sesión, con `estado_tarea = 'Pendiente'` y `activa = TRUE`, usando sentencias preparadas.
        6. Redirección a `index.php` con mensaje de éxito o de vuelta a `crear_tarea.php` con error.

- **Leer Tareas (`index.php` con `ACCIONES/Tareas/obtener_tareas.php`):**
    1. `index.php` (requiere estar logueado) incluye `obtener_tareas.php`.
    2. `obtener_tareas.php` consulta la tabla `Tareas` para obtener todas las tareas donde `id_usuario_fk` coincida con el usuario en sesión y `activa = TRUE`.
    3. Las tareas se ordenan por prioridad, fecha límite y fecha de creación.
    4. Los resultados se muestran en una tabla en `index.php`.
- **Actualizar Tarea (`editar_tarea.php` y `ACCIONES/Tareas/actualizar_datos_tarea.php`):**
    1. Desde `index.php`, el usuario hace clic en editar una tarea, pasando el `id_tarea` vía GET a `editar_tarea.php`.
    2. `editar_tarea.php` carga los datos de la tarea específica (verificando que pertenezca al usuario logueado) y los muestra en un formulario.
    3. El usuario modifica los datos y envía el formulario a `actualizar_datos_tarea.php`.
    4. Se validan los datos.
    5. Se actualiza la tarea en la tabla `Tareas` usando una sentencia preparada, asegurándose de que la actualización solo ocurra si `id_tarea` e `id_usuario_fk` coinciden.
    6. Redirección a `index.php` con mensaje de éxito o de vuelta a `editar_tarea.php` con error.
- **Marcar como Completada (`ACCIONES/Tareas/marcar_completada.php`):**
    1. Se accede vía GET con el `id_tarea` desde `index.php`.
    2. El script actualiza el `estado_tarea` a 'Completada' para la tarea especificada, solo si pertenece al usuario logueado y está activa.
    3. Redirección a `index.php` con mensaje.
- **Archivar Tarea ("Eliminación" Lógica) (`ACCIONES/Tareas/archivar_tarea.php`):**
    1. Se accede vía GET con el `id_tarea` desde `index.php`.
    2. El script actualiza `activa` a `FALSE` y `estado_tarea` a 'Cancelada' para la tarea especificada, solo si pertenece al usuario logueado.
    3. Redirección a `index.php` con mensaje.

- **Funcionalidades Adicionales:**
    - **Modo Oscuro/Claro:** Implementado con JavaScript (`JS/Auth.js`) que añade/quita una clase `dark-mode` al `<body>`. Las preferencias se guardan en `localStorage`. El CSS (`estilos_main.css` y `estilos_auth.css`) contiene variables de color para ambos modos.

Validación de Datos:

- **Frontend (JavaScript en `JS/Registro.js`):** Se implementa validación básica para los formularios de registro(campos vacíos, formato de email, longitud de contraseña, coincidencia de contraseñas) para mejorar la UX. Previene el envío del formulario si hay errores.
- **Backend (PHP):** Todos los scripts en `ACCIONES/` realizan validaciones exhaustivas de los datos recibidos antes de interactuar con la base de datos. Esta es la validación principal y de seguridad.

5. Seguridad Implementada

- **Prevención de Inyección SQL:** Todas las interacciones con la base de datos MySQL se realizan utilizando **sentencias preparadas** a través de la extensión MySQLi. Esto asegura que las entradas del usuario sean tratadas como datos y no como código SQL ejecutable.
- **Prevención de Cross-Site Scripting (XSS):** Se utiliza la función `htmlspecialchars()` de PHP antes de mostrar cualquier dato ingresado por el usuario en el HTML. Esto convierte caracteres especiales en entidades HTML, evitando que se interpreten como código ejecutable por el navegador.
- **Hashing de Contraseñas:** Las contraseñas de los usuarios se hashean utilizando `password_hash()` (con el algoritmo `PASSWORD_DEFAULT`) antes de almacenarse en la base de datos. Para la verificación durante el inicio de sesión, se utiliza `password_verify()`.
- **Protección de Sesiones:**
    - `session_start()` se usa para iniciar y manejar sesiones.
    - Se verifica `isset($_SESSION['id_usuario'])` en las páginas protegidas para asegurar que solo usuarios autenticados puedan acceder.
    - `session_regenerate_id(true)` se llama después de un inicio de sesión exitoso para prevenir ataques de fijación de sesión.
    - El script `logout.php` destruye la sesión y borra las cookies de sesión.
- **Validación de Datos del Lado del Servidor:** Toda la entrada del usuario que llega a los scripts de acción PHP es validada (tipos de datos, obligatoriedad, formatos específicos) antes de ser procesada o almacenada.
- **Control de Acceso a Datos:** Las consultas para editar, completar o archivar tareas incluyen una cláusula `WHERE id_usuario_fk = ?` para asegurar que un usuario solo pueda modificar sus propias tareas.