Resumen de Desarrollo - 13.04.2025

Introducción

En esta sesión se trabajó en la integración de una funcionalidad para mostrar un historial de estados de ánimo guardados por el usuario, utilizando JavaScript (AJAX), PHP y MySQL. El enfoque fue respetar el modelo MVC y seguir buenas prácticas como la separación de archivos por funcionalidad.

1. Envío de formulario con Fetch

Se utilizó JavaScript para enviar los datos del formulario (emociones y texto del diario) usando la función fetch() en lugar del envío tradicional. Esto permite una experiencia más dinámica y sin recargar la página.

Archivo: emociones.js

- Captura los valores del formulario.
- Usa fetch() para enviar los datos en formato JSON.
- Muestra mensaje de éxito y limpia el formulario tras unos segundos.

2. Guardado de datos en PHP

Archivo: moodController.php

- Lee los datos enviados con php://input y json_decode.
- Valida que el usuario esté autenticado.
- Llama a la función guardar() del modelo estadoAnimo.
- Devuelve un mensaje de confirmación en JSON.

3. Estructura del modelo PHP

Archivo: estadoAnimo.php

- Método guardar(): inserta en la tabla estados_animo usando PDO y prepared statements.
- Método historial($usuario_id): recupera los registros ordenados por fecha descendente.
- Método historialPorEstado($usuario_id, $estado): permite filtrar por estado emocional.

4. Implementación del botón 'Ver historial'

Archivo: dashboard.php

Resumen de Desarrollo - 13.04.2025

- Se añadió un botón con id='btn-ver-historial'.

Archivo: historial.js

- Captura el evento click sobre el botón.

- Envía una petición fetch al mismo controlador PHP, con un campo 'accion' => 'ver_historial'.

5. Respuesta del backend al botón de historial

Archivo: moodController.php

- Detecta si la acción es 'ver_historial'.

- Obtiene el usuario desde la sesión.

- Llama a estadoAnimo::historial($usuario_id).

- Devuelve un array JSON con los resultados.

6. Mostrar resultados en pantalla

Archivo: historial.js

- Cuando se recibe el array desde PHP, se genera dinámicamente una lista HTML (<ul> con <li>).

- Se insertan en un contenedor con id='historial-container'.

- Se limpia el contenido anterior para evitar duplicados.

7. Buenas prácticas seguidas

- Separación de lógica en archivos independientes (emociones.js y historial.js).

- Verificación de existencia de elementos en el DOM antes de manipularlos.

- Validaciones en PHP para evitar errores de ejecución.

- Uso de prepared statements para seguridad en las consultas SQL.

- Respuestas del servidor en JSON, claras y estructuradas.

8. Flujo general entre archivos

1. Usuario pulsa botón y JS (historial.js) envía petición.

2. moodController.php recibe y procesa según el valor de 'accion'.

3. Si es 'ver_historial', se obtiene el historial con el modelo estadoAnimo.

4. El resultado JSON vuelve al JS.

Resumen de Desarrollo - 13.04.2025

5. JS crea elementos HTML y los muestra en el dashboard.

6. Todo esto ocurre sin recargar la página, gracias a fetch().