Preguntes de reflexió

Rutas y controladores: ventajas de centralizar la validación y la gestión de sesión en el controlador

Ventajas:

- **Organización y mantenibilidad:** Toda la lógica de validación y almacenamiento de datos está concentrada en un único lugar (el controlador), lo que hace el código más limpio y fácil de mantener.
- **Reutilización:** Puedes reutilizar la misma validación o reglas en distintas vistas o formularios.
- **Seguridad:** Garantiza que solo los datos validados se guarden en la sesión, evitando errores o valores manipulados desde el cliente.
- **Separación clara de responsabilidades:** Las vistas Blade solo muestran información, mientras que el controlador se encarga de la lógica. Esto sigue el patrón **MVC (Modelo-Vista-Controlador)** correctamente.

Blade: cómo evitar errores cuando una clave puede no existir en un array

Si una clave de un array podría no existir, puedes evitar errores como *"Undefined array key"* usando diferentes métodos:

- **Operador null-coalescente (`??`):**

  ```
  {{ $equipo['nombre'] ?? 'Nombre desconocido' }}
  ```

  → Muestra "Nombre desconocido" si la clave `nombre` no existe.

- **Función `optional()` (para objetos):**

  ```
  {{ optional($jugadora)['posicion'] ?? 'Sin posición' }}
  ```

- **Directiva `@isset`:**

  ```
  @isset($equipo['estadio'])
      <p>Estadio: {{ $equipo['estadio'] }}</p>
  @endisset
  ```

Así evitas errores al acceder a índices o propiedades que no existen.

Vite: por qué es más robusto tener un solo `@vite` en el layout

- **Eficiencia y coherencia:** Laravel Vite genera un archivo `manifest.json` que describe los recursos compilados. Si colocas un solo `@vite` en el layout principal, Laravel sabe exactamente qué CSS y JS cargar sin duplicar.
- **Evita conflictos y recargas innecesarias:** Si hubiera varios `@vite` repartidos en distintas vistas, podrían producirse errores de compilación o carga duplicada de estilos/scripts.
- **Modularidad:** Es mejor importar los estilos adicionales dentro del archivo principal de CSS usando `@import`, manteniendo un único punto de entrada:

  ```
  @import './equips.css';
  @import './estadis.css';
  ```

  Esto simplifica el mantenimiento y aprovecha la caché del navegador.

Sesión vs Base de Datos

Limitaciones de la sesión:

- Los datos **se pierden al cerrar el navegador o expirar la sesión**.
- **Cada usuario tiene su propia sesión**, no se comparten datos entre ellos.
- No es adecuada para **grandes volúmenes de información**.
- No permite **consultas complejas** (como ordenar, filtrar o buscar fácilmente).

Cuándo migrar a modelos y migraciones (base de datos):

- Cuando necesitas **persistencia real** (que los datos no se borren al reiniciar el servidor).
- Si varios usuarios deben **compartir o modificar** la misma información.
- Cuando el proyecto crece y requiere **consultas avanzadas** o relaciones entre tablas (por ejemplo, un equipo con muchas jugadoras).
- Si buscas **escalabilidad y control de integridad de datos**.