Estrutura de Componentes (Livewire + Blade)

1. Remover componentes gigantes → quebrar em subcomponentes menores

Componentes Livewire muito grandes geram:

- snapshots enormes
- múltiplos listeners
- muitos dados sincronizados
- renderizações lentas

➡️ **Solução: dividir em componentes menores.**

Exemplo:

- Header
- Notifications
- UserMenu
- FavoritesMenu
- TaskTimer

Assim cada parte controla apenas seu próprio estado — diminuindo drasticamente o snapshot.

✅ 2. Remover propriedades públicas pesadas

Propriedades públicas fazem parte do snapshot do Livewire.
Logo, tudo o que você coloca nelas é transmitido para o navegador e devolvido ao backend.

✔️ **Evite guardar:**

- coleções grandes (Task::all())
- listas de objetos Eloquent
- arrays com múltiplos níveis
- blobs de base64
- JSONs enormes

➡️ **Solução:** guardar apenas IDs, flags, ou dados mínimos realmente necessários.

---

3. Substituir propriedades públicas por computed properties

Computed properties são métodos que calculam algo quando necessário, sem armazenar no snapshot.

Exemplo ruim:

```
public $tasks;
$this->tasks = Task::all();
```

Exemplo bom:

```
public function getTasksProperty()
{
    return Task::all();
}
```

Vantagens:

- Não vai para o snapshot.
- Atualiza somente quando acessado no Blade.
- Carrega dados mais rápido e com menos memória.

---

4. Conferir snapshots grandes

Snapshots Livewire são o "estado serializado" do componente enviado ao navegador.

Se o tamanho estiver muito grande (axios request de 200kb+):

- a UX vai ficar lenta
- interações travam
- o servidor envia e recebe dados demais

✔️ **Verifique no DevTools → aba Network → requisições livewire/message.**

➡️ **Se estiver muito grande, revise:**

- propriedades públicas
- arrays carregados no render()
- relacionamentos inteiros sendo enviados
- dados inúteis sendo expostos

---

5. Usar lazy-loading

Lazy loading evita fazer consultas e cálculos antes da hora.

Exemplos:

- wire:model.lazy: só sincroniza quando o input perde o foco
- wire:init="loadData": só carrega dados pesados depois que o componente aparece
- computed properties (não carregam antes da hora)

➡️ **Reduz queries e snapshot, aumenta performance.**

---

6. Usar wire:key corretamente

wire:key ajuda o Livewire a identificar cada elemento de uma lista.

Sem wire:key, Livewire pode:

- perder referências
- duplicar elementos
- renderizar blocos inteiros quando só 1 item mudou

Exemplo certo:

```
@foreach ($tasks as $task)
    <div wire:key="task-{{ $task->id }}">
        {{ $task->title }}
    </div>
@endforeach
```

✅ 7. Remover models completos das propriedades

Propriedades Livewire são serializadas no snapshot.
Modelos Eloquent:

❌ trazem atributos + relacionamentos + casts + metadata
❌ deixam o snapshot enorme
❌ podem trazer relacionamentos não carregados
❌ tornam o componente lento e pesado

❌ **Errado:**

```
public Task $task;
```

✅ **Certo:**

```
public $taskId;

// E usar o model quando precisar:
public function getTaskProperty()
{
    return Task::find($this->taskId);
}
```

✅ 8. Dividir responsabilidades

Um componente Livewire não deve:

- emitir eventos para 10 lugares
- controlar notificações, favoritos, menu, tasks, status, uploads, etc.
- chamar 5 queries no render()
- ter 20 listeners diferentes

Isso causa:

- confusão
- lentidão
- snapshots gigantes
- bugs difíceis de rastrear

❌ **Errado:**

Um componente gigante como Header controlando:

- Timer
- Notificações
- Favoritos
- Menu
- Upload de imagem
- Status Do Not Disturb
- Acesso ao SAR e SAT
- Tasks

✅ **Certo:**

Separar em pequenos componentes Livewire:

- header-user-menu
- header-notifications
- header-tasks
- header-favorites
- header-status-indicator

Cada componente:

- tem o próprio snapshot
- atualiza somente seu estado
- fica mais fácil manter

---

9. Usar select() explícito

Por padrão, o Eloquent faz:

```
SELECT * FROM tasks
```

Isso é ruim porque:

- traz colunas desnecessárias
- aumenta tráfego
- aumenta memória
- deixa snapshots Livewire maiores quando você envia os dados ao front

❌ **Errado:**

```
$task = Task::find($id); // carrega tudo
```

✅ **Certo:**

```
$task = Task::select('id', 'title', 'id_creator')->find($id);
```

Livewire snapshots

- Não colocar arrays grandes em propriedades públicas.
- Não armazenar modelos Eloquent completos.
- Guardar apenas IDs.
- Remover dados estáticos do fluxo Livewire.
- Usar Alpine para estados voláteis.
- Evitar get()->toArray() no render().

---

10. Resolver N+1

O N+1 acontece quando você carrega uma lista e depois acessa relacionamentos dentro dela.

Como resolver:

✔️ **Use with()**

```
$comments = Post::with('user')->get();
```

✔️ **Use load() para carregar depois**

```
$posts = Post::all();
$posts->load('user', 'category');
```

✔️ **Use eager loading condicional**

```
$tasks = Task::with(['user' => fn($q) => $q->select('id','name')])->get();
```

O Telescope alerta sobre N+1