❌ 18ª VALIDAÇÃO - SPRINT 65 - CAUSA RAIZ IDENTIFICADA

Data: 20 de novembro de 2025

Horário: 11:45 (GMT-3)

Validador: Sistema de Testes Automatizado

Sprint: 65 - Fix React Error #310 (Hoisting Components)

📋 SUMÁRIO EXECUTIVO

Status: ❌ FALHA - BUG #3 PERSISTE

A Sprint 65 implementou correção parcial movendo componentes `BarChart`, `MetricCard` e `DonutChart` para fora do render (hoisting), mas o **React Error #310 (infinite re-render loop) ainda persiste**.

✅ Progresso Positivo:

- Novo build gerado com sucesso (`Analytics-Bsx6e2-N.js`)
- PM2 reiniciado corretamente
- Componentes hoisted conforme planejado
- Build sendo servido corretamente

❌ Problema Crítico:

- **CAUSA RAIZ IDENTIFICADA:** Funções `calculateStats()` e `calculateSystemHealth()` são chamadas **diretamente no corpo do componente** (linha 531-532), causando recálculo a cada render e loop infinito

---

🔍 ANÁLISE DETALHADA

1. Validação do Build da Sprint 65

✅ Build Gerado Corretamente

```bash
# Arquivo no servidor
/home/flavio/webapp/dist/client/assets/Analytics-Bsx6e2-N.js
```

```
Tamanho: 31KB
Data: 20 Nov 13:11
```

✅ PM2 Restart Executado

Plain Text

```
PID: 767545
Uptime: 14 minutos
Status: online
Restarts: 28
```

✅ Novo Build Carregado no Browser

JavaScript

```
// Console do browser confirma:
at o (http://localhost:3001/assets/Analytics-Bsx6e2-N.js:1:9264 )
// Antes era: Analytics-CwqmYoum.js (Sprint 64)
```

2. Erro Persistente

❌ React Error #310 Ainda Ocorre

Plain Text

```
Error: Minified React error #310
at o (http://localhost:3001/assets/Analytics-Bsx6e2-N.js:1:9264 )
```

Significado do Erro #310:

> "Too many re-renders. React limits the number of renders to prevent an infinite loop."

3. Causa Raiz Identificada

🔴 Problema no Código (AnalyticsDashboard.tsx)

Linhas 531-532:

TypeScript

```
const stats = calculateStats();
```

```
const health = calculateSystemHealth();
```

Por que isso causa loop infinito:

1. **Render inicial** → `calculateStats()` e `calculateSystemHealth()` são executadas
2. **tRPC queries atualizam** (refetchInterval: 10s) → trigger re-render
3. **Re-render** → `calculateStats()` e `calculateSystemHealth()` executam NOVAMENTE
4. **Funções criam novos objetos** → React detecta mudança
5. **Trigger outro re-render** → LOOP INFINITO! 🔄

✅ Componentes Já Hoisted (Sprint 65)

TypeScript

```
// Linha 19 - FORA do AnalyticsDashboard
const BarChart: React.FC<{ data: ChartData; colors: string[] }> = ...

// Linha 46 - FORA do AnalyticsDashboard
const MetricCard: React.FC<{...}> = ...

// Linha 72 - FORA do AnalyticsDashboard
const DonutChart: React.FC<{...}> = ...

// Linha 109 - Componente principal
export const AnalyticsDashboard: React.FC = () => {
```

Análise: Componentes estão corretos, mas o problema está nas **funções de cálculo**.

🐛 BUG #3 - ANÁLISE COMPLETA

Histórico de Tentativas de Correção

Sprint	Tentativa	Resultado
60	Aumentar timeout de métricas para 60s	❌ Falhou
61	Remover useEffect redundante de auto-refresh	❌ Falhou
62	Corrigir cache e conexão MySQL	✅ Sucesso (outros bugs)

63	Adicionar arquivo .env com credenciais	✅ Sucesso (outros bugs)
64	Remover setRenderError do catch block	❌ Falhou
65	**Hoist components fora do render**	❌ **Falhou**

Código Problemático Atual

TypeScript

```
// ❌ PROBLEMA: Executado a cada render
export const AnalyticsDashboard: React.FC = () => {
  // ... queries tRPC ...

  const calculateStats = () => {
    // Cria novo objeto a cada chamada
    return {
      totalTasks: tasks.length,
      totalProjects: projects.length,
      // ... mais propriedades ...
    };
  };

  const calculateSystemHealth = () => {
    // Cria novo objeto a cada chamada
    return {
      status: 'healthy',
      color: 'text-green-500',
      // ...
    };
  };

  // ❌ CAUSA DO LOOP: Chamadas diretas no corpo do componente
  const stats = calculateStats();        // ← Linha 531
  const health = calculateSystemHealth(); // ← Linha 532

  // ... resto do render ...
}
```

Solução Proposta - Sprint 66

TypeScript

```typescript
import React, { useState, useEffect, useMemo } from 'react'; // ← Adicionar useMemo

export const AnalyticsDashboard: React.FC = () => {
  // ... queries tRPC ...

  // ✅ SOLUÇÃO: Usar useMemo para memoizar resultados
  const stats = useMemo(() => {
    try {
      console.log('[SPRINT 66] Calculating stats...');
      return {
        totalTasks: tasks.length,
        totalProjects: projects.length,
        completedTasks: tasks.filter(t => t.status === 'completed').length,
        activeTasks: tasks.filter(t => t.status === 'in_progress').length,
        activeProjects: projects.filter(p => p.status === 'active').length,
        totalWorkflows: workflows.length,
        activeWorkflows: workflows.filter(w => w.status === 'active').length,
        avgWorkflowSteps: workflows.length > 0
          ? workflows.reduce((sum, w) => sum + (w.steps?.length || 0), 0) /
workflows.length
          : 0,
        totalTemplates: templates.length,
        publicTemplates: templates.filter(t => t.isPublic).length,
        totalTemplateUsage: templates.reduce((sum, t) => sum + (t.usageCount
|| 0), 0),
        totalTeams: teams.length,
        totalMembers: teams.reduce((sum, t) => sum + (t.memberCount || 0),
0),
        avgTasksPerProject: projects.length > 0 ? tasks.length /
projects.length : 0,
        avgPromptsPerProject: projects.length > 0 ? prompts.length /
projects.length : 0,
        systemHealth: calculateSystemHealth(),
      };
    } catch (error) {
      console.error('[SPRINT 66] Error calculating stats:', error);
      return {
        totalTasks: 0,
        totalProjects: 0,
        completedTasks: 0,
        activeTasks: 0,
        activeProjects: 0,
        totalWorkflows: 0,
        activeWorkflows: 0,
        avgWorkflowSteps: 0,
        totalTemplates: 0,
```

```
        publicTemplates: 0,
        totalTemplateUsage: 0,
        totalTeams: 0,
        totalMembers: 0,
        avgTasksPerProject: 0,
        avgPromptsPerProject: 0,
        systemHealth: { status: 'unknown', color: 'text-gray-500', label:
'Erro', icon: '⚠️' },
      };
    }
  }, [tasks, projects, workflows, templates, prompts, teams]); // ←
Dependências

  const health = useMemo(() => {
    try {
      console.log('[SPRINT 66] Calculating system health...');

      if (!metrics?.metrics) {
        return { status: 'unknown', color: 'text-gray-500', label:
'Desconhecido', icon: '?' };
      }

      const cpu = metrics.metrics.cpu || 0;
      const memory = metrics.metrics.memory || 0;
      const disk = metrics.metrics.disk || 0;

      const cpuHealth = cpu < 80;
      const memoryHealth = memory < 85;
      const diskHealth = disk < 90;

      if (cpuHealth && memoryHealth && diskHealth) {
        return { status: 'healthy', color: 'text-green-500', label:
'Saudável', icon: '✓' };
      } else if (cpuHealth && memoryHealth) {
        return { status: 'warning', color: 'text-yellow-500', label:
'Atenção', icon: '⚠' };
      } else {
        return { status: 'critical', color: 'text-red-500', label:
'Crítico', icon: '✗' };
      }
    } catch (error) {
      console.error('[SPRINT 66] Error calculating health:', error);
      return { status: 'error', color: 'text-red-500', label: 'Erro', icon:
'✗' };
    }
  }, [metrics]); // ← Dependência
```

```
  // ... resto do render ...
}
```

Mudanças Necessárias:

1. ✅ Importar `useMemo` do React
2. ✅ Remover funções `calculateStats()` e `calculateSystemHealth()`
3. ✅ Substituir por `useMemo` com dependências corretas
4. ✅ Mover lógica de `calculateSystemHealth()` para dentro do `useMemo` de `stats` ou criar `useMemo` separado

---

🧪 TESTES REALIZADOS

1. Verificação do Build

Bash

```
✅ Arquivo existe no servidor
✅ Timestamp correto (13:11)
✅ Tamanho: 31KB
```

2. Verificação do PM2

Bash

```
✅ Status: online
✅ PID: 767545
✅ Uptime: 14 minutos
✅ MySQL conectado
```

3. Verificação do Browser

Bash

```
✅ Novo build carregado (Analytics-Bsx6e2-N.js)
✅ Cache limpo com sucesso
✅ Hard refresh executado
❌ Erro React #310 persiste
```

4. Análise do Console

JavaScript

```
// Logs mostram queries funcionando:
[SPRINT 55]: # "Analytics queries starting..."
[SPRINT 55]: # "Calling tasks.getStats..."
[SPRINT 55]: # "tasks.getStats result: {data: Object, error: null}"
[SPRINT 55]: # "Extracted data counts: {tasks: 9, projects: 30, workflows:
7}"

// Mas então:
Error: Minified React error #310
at o (http://localhost:3001/assets/Analytics-Bsx6e2-N.js:1:9264 )
```

5. Análise do Código Fonte

Bash

```
✅ Componentes hoisted (linhas 19, 46, 72)
✅ useEffect correto com [] dependencies
❌ calculateStats() chamado diretamente (linha 531)
❌ calculateSystemHealth() chamado diretamente (linha 532)
```

🔧 INCIDENTES DURANTE VALIDAÇÃO

1. Túnel SSH Caiu Durante Testes

Problema:

Plain Text

```
ERR_CONNECTION_REFUSED
localhost refused to connect
```

Causa: Túnel SSH foi encerrado após PM2 restart

Solução Aplicada:

Bash

```
# Recriado túnel com keepalive
nohup sshpass -p "sshflavioia" ssh -N -L 3001:localhost:3001 \
  -o StrictHostKeyChecking=no \
```

```
  -o ServerAliveInterval=60 \
  -p 2224 flavio@31.97.64.43 > /tmp/ssh_tunnel.log 2>&1 &
```

Status: ✅ Resolvido

2. Cache Agressivo do Browser

Problema: Browser continuava carregando build antigo após PM2 restart

Solução Aplicada:

JavaScript

```
// Limpeza programática de cache
await caches.keys().then(names =>
  Promise.all(names.map(name => caches.delete(name)))
);
localStorage.clear();
sessionStorage.clear();
```

Status: ✅ Resolvido

📊 STATUS DOS 3 BUGS PRINCIPAIS

Bug	Descrição	Status Sprint 65	Tentativas
#1	Chat - Send button não funciona	✅ **RESOLVIDO** (Sprint 54)	4
#2	Follow-up - Chat não envia mensagens	✅ **RESOLVIDO** (Sprint 54)	4
#3	Analytics - React Error #310	❌ **PERSISTE**	**6**

🎯 RECOMENDAÇÕES PARA SPRINT 66

Prioridade CRÍTICA

1. Implementar useMemo para Cálculos

Arquivo: `/home/flavio/webapp/client/src/components/AnalyticsDashboard.tsx`

Mudanças:

TypeScript

```typescript
// Linha 6 - Adicionar useMemo
import React, { useState, useEffect, useMemo } from 'react';

// Linhas 370-493 - REMOVER funções calculateStats e calculateSystemHealth

// Linha 531 - SUBSTITUIR por:
const stats = useMemo(() => {
  // Mover toda lógica de calculateStats aqui
  return {
    totalTasks: tasks.length,
    totalProjects: projects.length,
    // ... todas as propriedades ...
    systemHealth: health, // Usar health memoizado
  };
}, [tasks, projects, workflows, templates, prompts, teams, health]);

// Linha 532 - SUBSTITUIR por:
const health = useMemo(() => {
  // Mover toda lógica de calculateSystemHealth aqui
  if (!metrics?.metrics) {
    return { status: 'unknown', color: 'text-gray-500', label:
'Desconhecido', icon: '?' };
  }
  // ... resto da lógica ...
}, [metrics]);
```

Importante: `health` deve ser calculado ANTES de `stats` pois `stats` depende de `health`.

2. Build e Deploy

Bash

```bash
# 1. Fazer mudanças no código
cd /home/flavio/webapp/client/src/components
# Editar AnalyticsDashboard.tsx

# 2. Build
cd /home/flavio/webapp
npm run build

# 3. Restart PM2
pm2 restart orquestrador-v3
```

```
# 4. Verificar logs
pm2 logs orquestrador-v3 --lines 50
```

3. Validação

1. ✅ Verificar novo build gerado em `dist/client/assets/Analytics-*.js`
2. ✅ Confirmar PM2 restart com novo PID
3. ✅ Hard refresh no browser (Ctrl+Shift+R)
4. ✅ Verificar console para logs `[SPRINT 66]`
5. ✅ Confirmar que página Analytics carrega com 10 cards de dados
6. ✅ Verificar que não há React Error #310

📈 PROGRESSO GERAL DO PROJETO

Bugs Resolvidos (2/3)

1. ✅ **Bug #1 - Chat Send** (Sprint 54)
2. ✅ **Bug #2 - Follow-up Chat** (Sprint 54)

Bugs Pendentes (1/3)

1. ❌ **Bug #3 - Analytics React Error #310** (Sprint 65 falhou)

Taxa de Sucesso

- **Bugs Resolvidos:** 66.67% (2/3)
- **Sprints Executadas:** 65
- **Validações Realizadas:** 18
- **Tempo Decorrido:** ~3 semanas

🔄 PRÓXIMOS PASSOS

Sprint 66 - Implementar useMemo (CRÍTICO)

Objetivo: Resolver React Error #310 definitivamente

Ações:

1. Adicionar `useMemo` ao import do React

2. Substituir `calculateStats()` por `useMemo` com dependências
3. Substituir `calculateSystemHealth()` por `useMemo` com dependências
4. Build e deploy
5. Validação completa

Estimativa: 15 minutos

Prioridade: 🔴 CRÍTICA

Após Sprint 66

Se Bug #3 for resolvido:

1. ✅ Validar os 3 bugs principais novamente
2. ✅ Continuar teste completo das 23 páginas do sistema
3. ✅ Validar CRUD operations em todas as páginas
4. ✅ Testar cenários de erro e edge cases
5. ✅ Gerar relatório final de 100% de cobertura

📝 NOTAS TÉCNICAS

Diferença entre Sprint 64 e 65

Sprint 64:

- Build: `Analytics-CwqmYoum.js`
- Componentes definidos DENTRO do render
- Erro na linha 7000

Sprint 65:

- Build: `Analytics-Bsx6e2-N.js`
- Componentes hoisted FORA do render ✅
- Erro na linha 9264 (diferente!)

Conclusão: Hoisting ajudou parcialmente, mas não resolveu completamente.

Por que useMemo é Necessário

```
TypeScript
```

```
// ❌ SEM useMemo - Cria novo objeto a cada render
const stats = calculateStats(); // Novo objeto toda vez

// ✅ COM useMemo - Reutiliza objeto se dependências não mudaram
const stats = useMemo(() => calculateStats(), [deps]); // Mesmo objeto
```

React compara objetos por referência:

- Sem useMemo: `stats !== stats` (novo objeto) → trigger re-render
- Com useMemo: `stats === stats` (mesmo objeto) → sem re-render

🎓 LIÇÕES APRENDIDAS

1. Hoisting Não é Suficiente

Mover componentes para fora do render ajuda, mas **cálculos pesados também precisam ser memoizados**.

2. React Error #310 Tem Múltiplas Causas

- ❌ Componentes definidos dentro do render
- ❌ Funções de cálculo chamadas diretamente
- ❌ setState em catch blocks
- ❌ useEffect com dependências que mudam sempre

3. Debugging de Build Minificado é Difícil

Linha 9264 não corresponde a linha no código fonte. Análise do código fonte TypeScript é essencial.

4. Cache do Browser é Agressivo

Hard refresh nem sempre funciona. Limpeza programática via JavaScript é mais confiável.

✅ CHECKLIST PARA SPRINT 66

- [ ] Adicionar `useMemo` ao import
- [ ] Criar `useMemo` para `health` com dependência `[metrics]`

- [ ] Criar `useMemo` para `stats` com dependências `[tasks, projects, workflows, templates, prompts, teams, health]`
- [ ] Remover funções `calculateStats()` e `calculateSystemHealth()`
- [ ] Mover lógica para dentro dos `useMemo`
- [ ] Adicionar logs `[SPRINT 66]` para debugging
- [ ] Build: `npm run build`
- [ ] Deploy: `pm2 restart orquestrador-v3`
- [ ] Validar: Hard refresh e verificar console
- [ ] Confirmar: Página carrega sem erro
- [ ] Testar: 10 cards de dados aparecem
- [ ] Validar: Auto-refresh funciona sem loop

📞 CONTATO

Para dúvidas sobre este relatório:

- **Sistema:** Orquestrador de IAs V3.5.1
- **Ambiente:** Produção (192.168.1.247:3001)
- **Acesso:** Via SSH tunnel (31.97.64.43:2224)

Relatório gerado automaticamente em: 20/11/2025 11:45 GMT-3
Próxima validação: Após Sprint 66
Status: ⏳ Aguardando correção do desenvolvedor