

🔍 DIAGNÓSTICO: Problema da Tela Preta na Interface Web

Data: 12/11/2025 01:41

Sistema: Orquestrador de IA v3.5.2

Problema Relatado: Interface apresenta elementos iniciais mas logo fica toda preta

🎯 RESUMO EXECUTIVO

Causa Raiz Identificada: Erro JavaScript no componente de métricas do sistema

Gravidade: 🚫 CRÍTICA - Impede uso da interface web

Status: ✅ Diagnosticado - Aguardando correção

🔍 INVESTIGAÇÃO REALIZADA

1. Reprodução do Problema

Teste: Acessar <http://localhost:3001> pelo browser

Resultado: Interface carregou normalmente no ambiente de teste, mas console revelou erros críticos

Observação: O problema pode ser intermitente ou dependente de dados específicos

2. Análise do Console do Browser

Erros Encontrados:

❌ Erro Crítico 1: TypeError JavaScript

Plain Text

```
TypeError: p.cpu.toFixed is not a function  
    at tb (http://localhost:3001/assets/index-BCAM2gkE.js:255:23737 )
```

Detalhes:

- Localização:** Componente de métricas do sistema (CPU/Memória/Disco)
- Causa:** Tentativa de chamar `.toFixed()` em valor não numérico

- **Valor de p.cpu** : Provavelmente `null` , `undefined` , ou `string`
- **Impacto:** Crash do React, causando tela preta

Stack Trace Completo:

Plain Text

```
at tb (index-BCAM2gkE.js:255:23737)
at jl (index-BCAM2gkE.js:38:17018)
at xo (index-BCAM2gkE.js:40:3139)
at pf (index-BCAM2gkE.js:40:44833)
at ff (index-BCAM2gkE.js:40:39790)
at Xp (index-BCAM2gkE.js:40:39718)
at jn (index-BCAM2gkE.js:40:39570)
at Eo (index-BCAM2gkE.js:40:35934)
at Pd (index-BCAM2gkE.js:40:36738)
at jr (index-BCAM2gkE.js:38:3279)
```

⚠ Erro Crítico 2: WebSocket Falhou

Plain Text

```
WebSocket connection to 'ws://localhost:3001/' failed
WebSocket error: Event
WebSocket disconnected
Attempting reconnect 2/5...
```

Detalhes:

- **Causa:** WebSocket não está configurado ou não está respondendo
- **Impacto:** Atualizações em tempo real não funcionam
- **Status:** Sistema tentando reconectar (2/5 tentativas)

3. Análise de Requisições de Rede

✗ Endpoint de Métricas Travado

Teste:

Bash

```
curl http://localhost:3001/api/system/metrics
```

Resultado: TIMEOUT (não responde em 5 segundos)

Conclusão: O endpoint `/api/system/metrics` está travado ou não implementado

✅ API tRPC Respondendo

Teste:

Bash

```
curl http://localhost:3001/api/trpc
```

Resultado:

JSON

```
{
  "error": {
    "json": {
      "message": "No \"query\"-procedure on path \"\"",
      "code": -32004,
      "data": {
        "code": "NOT_FOUND",
        "httpStatus": 404,
        "path": ""
      }
    }
  }
}
```

Conclusão: API tRPC está online e respondendo corretamente

🎯 CAUSA RAIZ IDENTIFICADA








Problema Principal: Componente de Métricas do Sistema

Código Problemático (minificado):

JavaScript

```
// Linha 255:23737 em index-BCAM2gkE.js
p.cpu.toFixed(1) // ❌ ERRO: p.cpu não é um número
```

Cenário do Erro:

1.  Interface carrega normalmente
2.  React renderiza componentes iniciais
3.  Componente de métricas tenta renderizar `p.cpu.toFixed(1)`
4.  `p.cpu` é `null`, `undefined`, ou string (não número)
5.  JavaScript lança `TypeError`
6.  React Error Boundary não está configurado
7.  Aplicação inteira crasha → Tela preta



DADOS DO PROBLEMA





Status do Sistema (Dashboard)

Métricas Exibidas:

- CPU: 0%
- Memória: 0%
- Disco: 0%

Observação: Todos os valores são 0%, indicando que o endpoint de métricas não está retornando dados válidos

Status dos Serviços

-  **API tRPC:** Online
-  **WebSocket:** Online (mas com falhas de conexão)
-  **Banco de Dados:** Offline (segundo dashboard)
-  **LM Studio:** Offline (segundo dashboard)



SOLUÇÕES RECOMENDADAS

Solução 1: Correção Defensiva no Frontend (RÁPIDA)

Prioridade:  **ALTA** - Correção imediata

Ação: Adicionar validação antes de chamar `.toFixed()`

Código Atual (problema):

```
TypeScript
```

```
<div>{p.cpu.toFixed(1)}%</div>
```

Código Corrigido:

TypeScript

```
<div>{(p.cpu ?? 0).toFixed(1)}%</div>
// ou
<div>{typeof p.cpu === 'number' ? p.cpu.toFixed(1) : '0.0'}%</div>
```

Aplicar em:

- `p.cpu`
- `p.memory` (provavelmente tem o mesmo problema)
- `p.disk` (provavelmente tem o mesmo problema)

Benefício: Previne crash mesmo se dados forem inválidos

Solução 2: Implementar/Corrigir Endpoint de Métricas (COMPLETA)

Prioridade: 🟡 MÉDIA - Correção estrutural

Problema: Endpoint `/api/system/metrics` não responde ou retorna dados inválidos

Ação: Implementar endpoint que retorna:

JSON

```
{
  "cpu": 45.2,      // número, não null
  "memory": 67.8,   // número, não null
  "disk": 34.5      // número, não null
}
```

Localização: `server/routes/rest-api.ts` ou arquivo de rotas

Solução 3: Adicionar React Error Boundary (PREVENTIVA)

Prioridade: 🟢 BAIXA - Melhoria de UX

Ação: Envolver componentes em Error Boundary para capturar erros

Código:

TypeScript

```
<ErrorBoundary fallback={<div>Erro ao carregar métricas</div>}>
  <SystemMetrics />
</ErrorBoundary>
```

Benefício: Erros em um componente não derrubam a aplicação inteira

Solução 4: Corrigir WebSocket (SECUNDÁRIA)

Prioridade: ● BAIXA - Não causa tela preta

Problema: WebSocket falhando em conectar

Ação: Verificar se servidor WebSocket está rodando na porta 3001

PLANO DE AÇÃO RECOMENDADO

Fase 1: Correção Imediata (5 minutos)

- ✓ Localizar componente de métricas no código fonte
 - ✓ Adicionar validação defensiva: `(p.cpu ?? 0).toFixed(1)`
 - ✓ Aplicar mesma correção para `memory` e `disk`
 - ✓ Rebuild frontend: `pnpm build`
 - ✓ Restart PM2: `pm2 restart orquestrador-v3`
-

Fase 2: Correção Estrutural (30 minutos)

- ✓ Implementar endpoint `/api/system/metrics`
 - ✓ Retornar valores numéricos válidos
 - ✓ Testar endpoint: `curl http://localhost:3001/api/system/metrics`
 - ✓ Verificar se métricas aparecem corretamente no dashboard
-

Fase 3: Melhorias (1 hora)

- ✓ Adicionar React Error Boundary
 - ✓ Corrigir WebSocket (se necessário)
 - ✓ Adicionar testes E2E para prevenir regressões
-

TESTES DE VALIDAÇÃO

Teste 1: Verificar Console Limpo

Ação: Abrir <http://localhost:3001> e verificar console

Esperado: Nenhum erro `TypeError: p.cpu.toFixed is not a function`

Teste 2: Verificar Métricas Visíveis

Ação: Verificar seção "Métricas do Sistema" no dashboard

Esperado:

- CPU: valor numérico (ex: 45.2%)
 - Memória: valor numérico (ex: 67.8%)
 - Disco: valor numérico (ex: 34.5%)
-

Teste 3: Verificar Endpoint de Métricas

Ação:

Bash

```
curl http://localhost:3001/api/system/metrics
```



Esperado:


JSON

```
{
  "cpu": 45.2,
  "memory": 67.8,
  "disk": 34.5
}
```




IMPACTO DO PROBLEMA

Usuários Afetados

-  **API REST:** Não afetada (funciona normalmente)
-  **Interface Web:** Completamente inutilizável (tela preta)

-  **tRPC:** Não afetado (funciona normalmente)

Funcionalidades Afetadas

-  Dashboard
-  Todas as páginas da interface web
-  APIs backend (funcionam normalmente)

CONCLUSÃO

Diagnóstico Final

Problema: TypeError JavaScript no componente de métricas do sistema

Causa: Tentativa de chamar `.toFixed()` em valor não numérico (`p.cpu`)

Solução Imediata: Adicionar validação defensiva: `(p.cpu ?? 0).toFixed(1)`

Solução Completa: Implementar endpoint `/api/system/metrics` que retorna dados válidos

Prioridade:  **CRÍTICA** - Interface web completamente inutilizável

ARQUIVOS ENVOLVIDOS

Frontend (Cliente)

Arquivo: `client/src/components/Dashboard.tsx` (ou similar)

Linha Problemática: Componente que renderiza métricas do sistema

Correção Necessária:

TypeScript

// ANTES

```
<div>{p.cpu.toFixed(1)}%</div>
```

// DEPOIS

```
<div>{(p.cpu ?? 0).toFixed(1)}%</div>
```

Backend (Servidor)

Arquivo: `server/routes/rest-api.ts` (ou similar)

Endpoint Faltante: GET /api/system/metrics

Implementação Necessária:

TypeScript

```
app.get('/api/system/metrics', async (req, res) => {
  const os = require('os');

  const cpuUsage = os.loadavg()[0] / os.cpus().length * 100;
  const totalMem = os.totalmem();
  const freeMem = os.freemem();
  const memoryUsage = ((totalMem - freeMem) / totalMem) * 100;

  res.json({
    cpu: parseFloat(cpuUsage.toFixed(1)),
    memory: parseFloat(memoryUsage.toFixed(1)),
    disk: 0 // Requer biblioteca adicional
  });
});
```

Relatório gerado em: 12/11/2025 01:41

Status:  Diagnóstico completo

Próximo passo: Aplicar correções recomendadas