

# Fix: Correção de Renderização de Produtos no Sistema V2



## Problema Identificado

Os produtos estavam sendo recebidos e processados corretamente pelo backend (logs confirmavam "Produtos divididos - TopBox: 3 Feed: 7"), mas **NÃO apareciam na interface visual** do sistema V2.



## Análise da Causa Raiz

### 1. Problema no Parsing de SSE (VendorAPI.ts)

- O backend envia produtos usando o marcador `__PRODUCTS__` no stream SSE
- O código frontend não estava detectando e parseando este marcador corretamente
- Formato do backend: `data: \n\n__PRODUCTS__{"products": [...]}`

### 2. Problema no Mapeamento de Produtos (VendorCore.tsx e useIntelligentVendor.ts)

- O backend envia produtos com campos: `id`, `name`, `price`, `imageUrl`, `storeName`, etc.
- O frontend estava tentando mapear incorretamente os campos
- Faltava tratamento robusto para diferentes formatos de preço (string vs number)

### 3. Problema na Chamada da API (VendorCore.tsx)

- O `VendorCore` estava usando `searchProducts()` (método antigo)
- Deveria usar `sendMessageToV2()` que processa SSE corretamente



## Correções Implementadas

### 1. VendorAPI.ts - Correção do Parser SSE

```
// ANTES: Não detectava o marcador __PRODUCTS__
if (data.products && Array.isArray(data.products)) {
  products = data.products;
}

// DEPOIS: Detecta e parseia o marcador __PRODUCTS__
if (dataContent.includes('__PRODUCTS__')) {
  const productsMatch = dataContent.match(/__PRODUCTS__(.+)/);
  if (productsMatch && productsMatch[1]) {
    const productsData = JSON.parse(productsMatch[1]);
    if (productsData.products && Array.isArray(productsData.products)) {
      products = productsData.products;
    }
  }
}
```

#### Melhorias:

- Detecta o marcador `__PRODUCTS__` no stream
- Extrai e parseia o JSON corretamente

- ☒ Mantém compatibilidade com formato JSON direto
- ☒ Adiciona buffer para lidar com chunks parciais

## 2. VendorCore.tsx - Uso Correto da API V2

```
// ANTES: Usava método antigo
products = await vendorAPI.current.searchProducts(analysis.searchQuery, {...});

// DEPOIS: Usa API V2 com SSE
v2Response = await vendorAPI.current.sendMessageToV2(
  session.userId || 'anonymous',
  userMessage,
  session.context.storeId
);
```

### Melhorias:

- ☒ Chama a API V2 que processa SSE corretamente
- ☒ Extraí produtos da resposta V2
- ☒ Mapeia campos corretamente do backend para frontend
- ☒ Fallback para método antigo em caso de erro
- ☒ Usa resposta de texto do backend quando disponível

## 3. Mapeamento Robusto de Produtos

```
products = v2Products.map((p: any) => {
  const priceValue = typeof p.price === 'string' ? parseFloat(p.price) : (p.price || 0);

  return {
    id: p.id || `product_${Date.now()}_${Math.random()}`,
    title: p.name || p.title || 'Produto sem título',
    name: p.name || p.title || 'Produto sem título',
    price: {
      USD: priceValue,
      BRL: priceValue * 5.5
    },
    imageUrl: p.imageUrl || p.image || '/placeholder-product.jpg',
    storeName: p.storeName || p.store || 'Loja não informada',
    category: p.category || 'geral',
    brand: p.brand || 'Marca não informada',
    // ... outros campos com fallbacks
  };
});
```

### Melhorias:

- ☒ Trata preço como string ou número
- ☒ Fallbacks para todos os campos obrigatórios
- ☒ Mapeia corretamente `name` → `title` e `name`
- ☒ Gera IDs únicos se não fornecidos
- ☒ Valores padrão para rating, reviews, availability

## 4. useIntelligentVendor.ts - Mesmas Correções

Aplicadas as mesmas correções de mapeamento no hook para consistência.



## Logs de Debug Adicionados

```
console.log(`📦 [V2] ✅ ${products.length} produtos recebidos via __PRODUCTS__ marker`);
sole.log(`📦 [V2] Produtos:`, products.map(p => p.name || p.title));
console.log(`📦 [VendorCore] Primeiro produto (raw):`, v2Products[0]);
console.log(`📦 [VendorCore] Produtos mapeados (primeiro):`, products[0]);
console.log(`✅ [VendorCore] Mensagem do assistente adicionada com ${products.length} produtos`);
```



## Como Testar

### 1. Ativar modo V2:

- Clicar no toggle "V2" na barra de busca
- Ou incluir "v2" na busca

### 2. Fazer uma busca:

"iPhone mais barato"

"Drone com câmera"

"Perfume masculino"

### 3. Verificar logs no console:

- 🔍 [V2] Searching products for: "..."
- 📦 [V2] Found X products
- 📦 [V2] ✅ X produtos recebidos via \_\_PRODUCTS\_\_ marker
- ✅ [VendorCore] Mensagem do assistente adicionada com X produtos

### 4. Verificar UI:

- Produtos devem aparecer no grid à direita
- Cada produto deve ter: imagem, título, preço, loja
- Contador de produtos no header deve mostrar número correto



## Arquivos Modificados

### 1. client/src/components/intelligent-vendor-v2/services/VendorAPI.ts

- Correção do parser SSE para detectar \_\_PRODUCTS\_\_
- Adição de buffer para chunks parciais
- Logs detalhados de debug








### 2. client/src/components/intelligent-vendor-v2/VendorCore.tsx

- Mudança de searchProducts() para sendMessageToV2()
- Mapeamento robusto de produtos
- Uso de resposta do backend V2
- Fallback para método antigo



### 3. client/src/components/intelligent-vendor-v2/hooks/useIntelligentVendor.ts

- Mapeamento robusto de produtos
- Logs detalhados de debug
- Tratamento de preço como string/number

## Resultado Esperado

-  Produtos aparecem na interface visual
-  TopBox mostra até 3 produtos em destaque
-  Feed mostra produtos restantes em grid
-  Todos os campos são exibidos corretamente
-  Imagens carregam corretamente
-  Preços formatados em USD/BRL
-  Logs confirmam fluxo completo

## Fluxo Completo Corrigido

1. Usuário digita busca  "iPhone"
2. VendorCore.processUserMessage() chamado
3. VendorAPI.sendMessageToV2() enviado ao backend
4. Backend processa e retorna SSE com \_\_PRODUCTS\_\_
5. VendorAPI.handleSSEResponse() detecta \_\_PRODUCTS\_\_
6. Produtos extraídos e parseados
7. Produtos mapeados para **formato** frontend
8. Produtos adicionados à mensagem do assistente
9. VendorInterface.**get**DisplayedProducts() obtém produtos
10. ProductGrid renderiza produtos na UI 

## Próximos Passos

- [ ] Testar com diferentes tipos de produtos
- [ ] Verificar performance com muitos produtos
- [ ] Adicionar testes unitários para parser SSE
- [ ] Melhorar tratamento de erros
- [ ] Adicionar loading states mais granulares