Canal E-commerce

Consulta ao SQL Server (Linx) e mapeamento de colunas para o padrão FullStore.

Coletamos pedidos do e-commerce processados pelo sistema **Linx**. Esses dados ficam em um SQL Server separado (`LX_ZERO_300`), em uma view que já consolida faturamento por vendedor. Nosso processo ocorre em **dois passos**.

Métodos: `buscarDadosEcommercePorLoja()` → `getVendasEcommerce()`

Diagrama de sequência — dois passos

isnapp buscarDadosEcommerce por Loja() Passo 1 - MySQL juridica + entidade j.cnpj = :cnpj e.situacao = ATIVO retorna codigos filial Placeholders :filial0, :filial1... bindValue() Passo 2 - SQL Server W_BI_FAT_VENDEDOR... JOIN FILIAIS TIPO_FILIAL != FRANQUIA vendas_ecommerce[] resolve CNPJ monta IN consulta BI Figura 8 - Sequencia dos dois passos de consulta do canal e-commerce.

Mapeamento de colunas SQL Server → JSON

Coluna SQL Server	Campo no JSON	Descrição
`CGC_CPF`	`documento_cliente`	CPF/CNPJ do cliente
`VENDEDOR`	`cod_vendedor`	Código do vendedor no Linx
`CUPOM_VENDEDOR`	`cupom_vendedor`	Cupom aplicado pelo vendedor
`CODIGO_FILIAL`	`codigo_filial`	Código da filial (padded com zeros)
`EMISSAO`	`data_documento`	Data de emissão no formato retornado pelo SQL Server; o método atual não converte para ISO 8601

Coluna SQL Server	Campo no JSON	Descrição
`PEDIDO_SITE`	`pedido_site`	Número do pedido no e-commerce
`PEDIDO_WMS`	`pedido_wms`	Número do pedido no WMS
`TICKET`	`ticket`	Ticket de atendimento
`NF`	`documento`	Número da NF
`SERIE`	`serie`	Série da NF
`VALOR_LIQUIDO`	`valor`	Valor líquido do pedido
`QTDE_LIQUIDA`	`qtde`	Quantidade líquida de itens

Tratamento de espaços Aplicamos `rtrim()` e `ltrim()` em todos os campos mapeados — o SQL Server da Linx pode devolver strings com espaços à esquerda ou direita.

Retorno da API

```
// Um elemento do array "vendas_ecommerce"
{
  "documento_cliente": "123.456.789-00",
  "cod_vendedor":      "EC001",
  "cupom_vendedor":    "VEND10OFF",         // Pode ser string vazia
  "codigo_filial":     "000001",
  "data_documento":    "2026-05-22 09:15:00",
  "pedido_site":       "PS-98765",
  "pedido_wms":        "WMS-54321",
  "ticket":            "TKT-11111",
  "documento":         "000005678",
  "serie":             "001",
  "valor":             "259.90",
  "qtde":              "2"
}
```

Segurança — placeholders dinâmicos

O número de filiais varia por CNPJ. Em vez de concatenar valores na query, geramos placeholders e usamos `bindValue()` para cada um:

```
// Geração dos placeholders (PHP 7.3)
$placeholders = [];
foreach ($filiais as $i => $codigo) {
    $key = ':filial' . $i;
    $placeholders[] = $key;
    $stmt->bindValue($key, $codigo);
}
// Resultado na query: IN (:filial0, :filial1, :filial2)
// Nenhum valor externo é interpolado na string SQL.
```

Proteção contra SQL injection Todos os parâmetros são vinculados via prepared statements — datas com `bindParam()` e códigos de filial com `bindValue()`. Nenhuma concatenação de input externo ocorre.
CNPJ sem filial ativa `buscarDadosEcommercePorLoja()` retorna um envelope de erro quando não encontra filial ativa no MySQL, mas `getVendasConsolidadas()` consome apenas a chave `data`. Na resposta final, isso aparece como `vendas_ecommerce: []`; se os demais canais também estiverem vazios, o endpoint retorna erro de nenhum registro.

Revision #1
Created 2026-06-09 19:00:50 UTC by Mauricio Francelino
Updated 2026-06-09 19:02:02 UTC by Mauricio Francelino