Fluxo Funcional

Passo a passo de uma requisição, validações e tratamento de erros.

Diagrama de sequência

FullStore isnapp run() + validação MySQL OSK SQL Server Linx/BI POST getVendas Normaliza período verifica limite 200 dias PDV / Trocas Omni + filiais consultas MySQL E-commerce W_BI_FAT_VENDEDOR... placeholders por filial mapeia vendas_ecommerce Monta JSON vendas / trocas / omni Figura 3 - Sequencia completa de uma chamada a acao getVendas.

Diagrama de fluxo (decisões)

POST getVendas ação válida? success:false Acao nao encontrada parâmetros preenchidos? success:false O parametro {campo} é obrigatorio dentro de 200 dias? success:false fallback interno não exposto consulta quatro canais se algum canal tem dados: success:true não sim não sim não sim Figura 4 - Fluxo de decisão completo do endpoint.

Passo a passo detalhado

1. **Recebimento da requisição**
   Recebemos o payload via `run(array $params)`. O parâmetro `acao` deve ser `'getVendas'`; qualquer outro valor resulta em erro imediato.
2. **Validação de parâmetros**
   Validamos via `validarParametrosObrigatorios()`: `dataInicio`, `dataFim` e `cnpjLoja` — presença e valor não vazio. Se algum faltar, lançamos uma exception com o nome do campo faltante.
3. **Normalização do período**
   Via `normalizarPeriodo()`, adicionamos horários às datas: início fica `dataInicio 00:00:00` e fim `dataFim 23:59:59`, garantindo cobertura integral dos dias extremos.
4. **Verificação do limite**
   Via `periodoExcedeLimite()`, checamos se alguma das datas está além de 200 dias no passado. Se sim, `getVendasConsolidadas()` monta arrays vazios e um objeto `fallback`, mas

`run()` ainda trata o resultado como nenhum registro encontrado e retorna erro ao consumidor.

5. **Vendas PDV**
   Executamos query com 2 CTEs nas tabelas `movimentacao + movimentacao_detalhe + movimentacao_nfe`. Retornamos uma linha por item e depois agrupamos por `identificador` via `agruparItensDasVendas()`, gerando o sub-array `itens`.
6. **Trocas**
   Buscamos movimentações de entrada com `tipo_estoque = 'TROCA'`, vinculando à venda original via `movimentacao_movimentacao`. Consolidamos múltiplas trocas do mesmo documento somando valores em `agruparTrocasPorDocumento()`.
7. **Omni**
   Buscamos pedidos omni criados pelo vendedor na loja (orçamentos com `tipo_estoque = 'ORCAMENTO'` e situação Confirmado/Finalizado), vinculados à venda real via `movimentacao_movimentacao`.
8. **E-commerce**
   Em dois sub-passos: (1) resolvemos os códigos de filial ativas via MySQL/CNPJ; (2) consultamos a view SQL Server com placeholders paramétricos dinâmicos para cada filial.
9. **Resposta**
   Montamos o objeto `data` com os quatro arrays e retornamos via `setSubmit(true, 'Vendas retornadas com sucesso!', $data)`.

Tratamento de erros

Situação	Comportamento	success
Parâmetro obrigatório ausente	Mensagem com nome do campo faltante	false
Ação inválida (≠ getVendas)	Mensagem de ação não encontrada	false
Período além de 200 dias	Fallback interno é montado, mas a resposta final atual é erro de nenhum registro	false
CNPJ sem filiais ativas no MySQL	`vendas_ecommerce` vazio; se os demais canais também estiverem vazios, retorna erro de nenhum registro	Depende
Falha no SQL Server	Exception capturada → retorno `false`	false
Parâmetro `falha` enviado	Força erro (uso em testes)	false

Logs automáticos de requisição Toda chamada — mesmo as que resultam em erro — é salva automaticamente nas tabelas `wosk_webservice*` do banco integrador. Em caso de disputa ou diagnóstico, basta consultar `wosk_webservice_retorno` pelo timestamp.

Revision #1
Created 2026-06-09 18:53:37 UTC by Mauricio Francelino

Updated 2026-06-09 18:54:44 UTC by Mauricio Francelino