Fluxo de execução principal

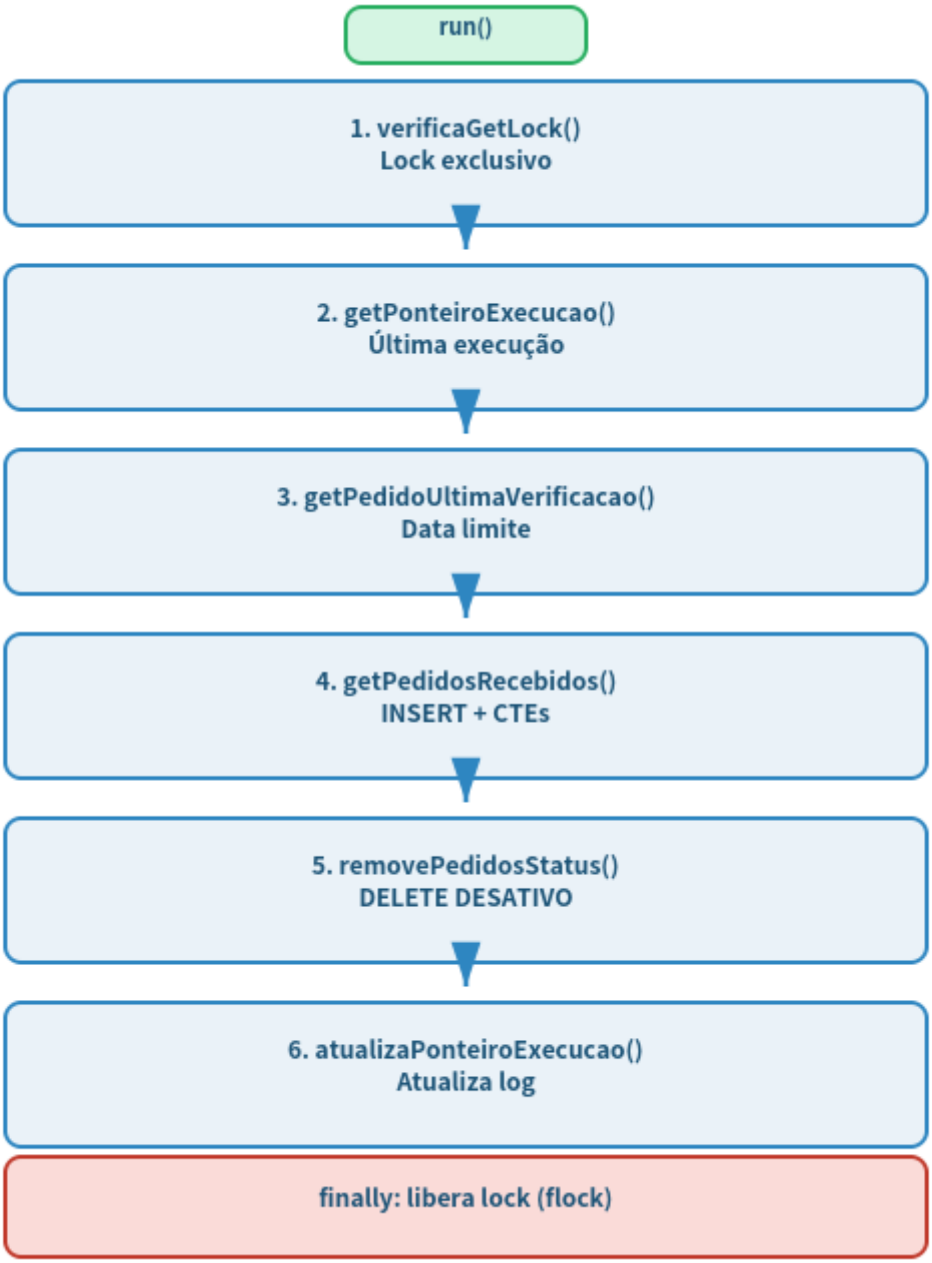

Figura 7 — Fluxograma completo do método run(), com 6 etapas e liberação do lock no finally.

Detalhamento de cada etapa

#	Método	O que faz (simples)	O que faz (técnico)
1	`verificaGetLock()`	Garante que só uma cópia rode por vez	`flock` exclusivo em arquivo temporário
2	`getPonteiroExecucao()`	Pergunta: "desde quando devo buscar?"	`MAX(execucao)` em `log` onde `procedure='franquia_1beat'`
3	`getPedidoUltimaVerificacao()`	Acha o pedido mais novo no período	`MAX(data)` em movimentação PEDIDO > ponteiro
4	`getPedidosRecebidos()`	Copia e calcula quantidades	INSERT com CTEs + ON DUPLICATE KEY UPDATE
5	`removePedidosStatus()`	Apaga pedidos cancelados/abertos	`DELETE WHERE SITUACAO = 'DESATIVO'`
6	`atualizaPonteiroExecucao()`	Marca que terminou até tal data	INSERT/UPDATE em `log`

Processamento incremental

O módulo **não reprocessa tudo** a cada execução. Ele usa um "marcador de página" (ponteiro) para buscar apenas pedidos alterados entre a última execução e a data do pedido mais recente.

Figura 8 — Apenas pedidos entre o ponteiro e a data mais recente são processados em cada execução.

Mensagens de erro comuns

- **"Ja tem outro processo em execucao, aguarde o termino"** — outra instância do `run()` está ativa.
- **"Não foram encontrados pedidos para atualizar"** — não há pedidos novos desde o ponteiro.
- **"Nao foi possivel criar o arquivo de lock"** — problema de permissão no diretório temporário.

Tratamento de erros: exceções no `run()` são capturadas e enviadas a `error_byexception()`. Os métodos intermediários (`getPedidosRecebidos`, `removePedidosStatus`, `atualizaPonteiroExecucao`) também possuem try/catch interno que retorna `null` em falha sem interromper o fluxo principal.

Revision #4
Created 2026-06-24 19:52:21 UTC by Gustavo Filgueiras
Updated 2026-06-25 12:34:27 UTC by Gustavo Filgueiras