Integração One Beat

- Introdução
- O que este sistema faz?
- Instalação e preparação
- Arquitetura do sistema
- Fluxo de execução principal
- Como os dados são processados
- Estrutura da tabela de destino
- Regras de negócio e situações
- Glossário e referência rápida

Introdução

Este material explica o funcionamento do módulo **onebeat_pedidos_faturados**, que faz parte do sistema ISNAPP. O texto foi escrito em duas camadas: uma linguagem simples para quem usa o dia a dia da operação, e detalhes técnicos para quem gerencia a área de Tecnologia da Informação.

Em poucas palavras: este programa pega as informações dos pedidos feitos pelas lojas, verifica o que já foi recebido e o que ainda falta chegar, e envia tudo para uma tabela especial que o sistema **OneBeat** usa para tomar decisões de estoque.
Resumo técnico: biblioteca PHP (CoreVersion 3) que sincroniza movimentações de pedidos do ERP para a tabela `franquia_1beat` no banco `franquia_osklen_relatorio`, com controle incremental por ponteiro de execução, lock de concorrência em arquivo e campos de status (`INATIVO`, `CONFIRMADO`) e quantidade cortada (`QTDE_CORTADA`).

Para quem é esta documentação?

Perfil	O que você vai encontrar
Usuário da operação / loja	Explicações sem termos difíceis sobre o que o sistema faz com os pedidos
Gestor ou diretor de T.I.	Arquitetura, bancos de dados, fluxos, instalação e regras de negócio
Equipe de suporte	Fluxos de execução, mensagens de erro e glossário

Arquivos do módulo

Arquivo	Função
`controle.php`	Lógica principal: coleta, transforma e grava os pedidos
`instalador.php`	Cria o banco e a tabela de destino na primeira instalação
`visual.html`	Tela simples de confirmação de carregamento do módulo

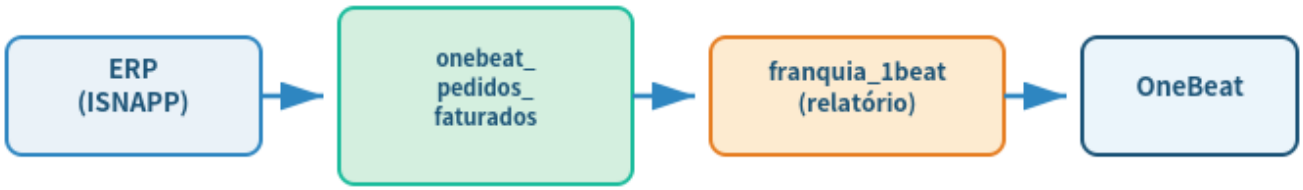


Figura 1 — O módulo fica no meio: lê os pedidos do ERP e prepara os dados para o OneBeat.

O que este sistema faz?

Imagine que uma loja da franquia faz um pedido de produtos para a matriz. O pedido é registrado no sistema. Depois, conforme os produtos chegam na loja, o sistema registra o que foi recebido.

Este módulo acompanha essa história e responde quatro perguntas importantes:

- **1. Quanto foi pedido?** — A quantidade total de cada produto no pedido.
- **2. Quanto já chegou?** — A quantidade que a loja já recebeu.
- **3. Quanto ainda falta?** — A diferença entre o pedido e o recebido que ainda pode chegar.
- **4. Quanto foi cortado?** — A parte do pedido que a matriz decidiu não enviar mais (quando o item tem corte).



Figura 2 — Dois exemplos com o mesmo pedido (10) e recebimento (7): sem corte ficam 3 pendentes; com corte ficam 3 cortadas e 0 pendentes.

Quando isso acontece?

O processo roda **automaticamente**, sem que ninguém precise clicar em botões. Ele verifica se há pedidos novos ou alterados desde a última vez que rodou e atualiza as informações.

Analogia: é como um carteiro que passa na loja todo dia, olha a lista de pedidos, anota o que mudou e leva a informação atualizada para o OneBeat.

O que acontece com pedidos cancelados ou abertos?

Se um pedido for **cancelado**, o sistema marca o registro como *inativo* (`INATIVO = 1`) e informa ao OneBeat que ele **não está confirmado** (`CONFIRMADO = 0`). O registro permanece na tabela, mas com a data de exclusão preenchida.

Se o pedido ainda estiver **aberto** (não finalizado), ele também fica com `CONFIRMADO = 0`, indicando que ainda não está apto para consumo pelo OneBeat. Diferente do cancelado, o registro **não** é marcado como inativo.

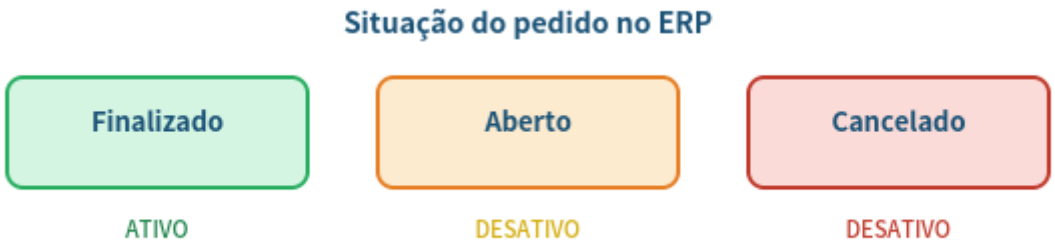


Figura 3 — Pedidos cancelados ficam inativos; pedidos abertos ficam não confirmados para o OneBeat.

E quando parte do pedido é cortada?

Corte de item: quando a matriz decide não enviar mais uma parte do pedido (por exemplo, faltou estoque), o sistema registra essa quantidade em `QTDE_CORTADA` em vez de deixá-la como pendente. Assim o OneBeat sabe que aquela diferença não vai mais chegar.

O que o usuário vê na tela?

A tela do módulo (`visual.html`) mostra apenas uma mensagem de confirmação: **"Carregado onebeat_pedidos_faturados com sucesso!"**. Não há formulários nem botões para o usuário comum.

Para o diretor de T.I.: a interface é mínima por design. O valor do módulo está na rotina `executar()`, executada via agendador (cron) ou chamada direta à URL `/bibliotecas/6b5b2af8-ab25-4135-9551-b3cb3e055f1a/onebeat_pedidos_faturados/executar`.

Instalação e preparação

O arquivo `instalador.php` cuida dessa preparação em **quatro etapas**, chamadas `v1`, `v2`, `v3` e `v4`. Cada etapa deve ser executada na ordem.

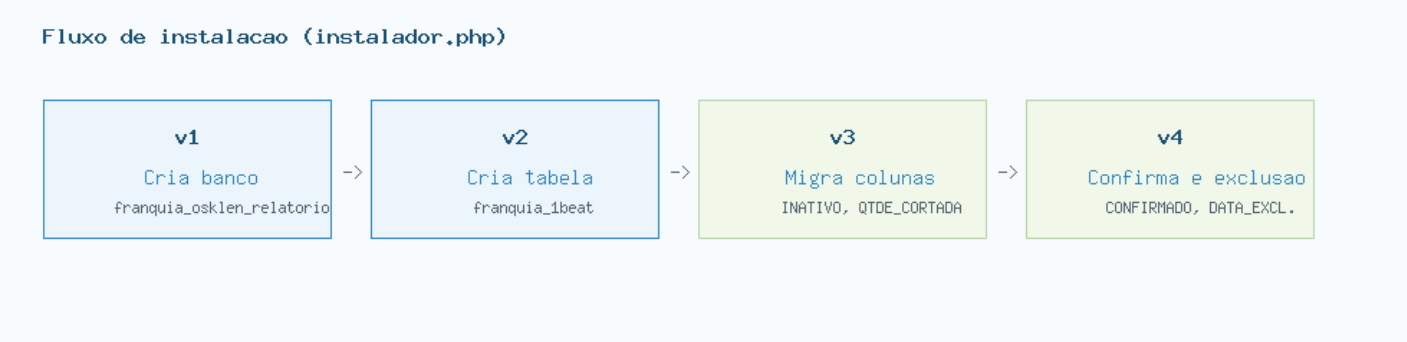


Figura 4 — Fluxo de instalação: v1 cria o banco, v2 cria a tabela, v3 e v4 aplicam migrações.

Etapa 1 — Criar o banco de dados (v1)

Item	Detalhe
Método	`instalador::v1()`
Ação	`CREATE DATABASE IF NOT EXISTS franquia_osklen_relatorio`
Retorno de sucesso	`setSubmit(true, "Feito")`
Em caso de erro	Exceção registrada via `error_byexception()`

URL de instalação:
`/bibliotecas/6b5b2af8-ab25-4135-9551-b3cb3e055f1a/onebeat_pedidos_faturados/install`
O instalador herda de `biblioteca` e usa a classe `util` para executar SQL com permissão elevada (`query($sql, true)`).

Etapa 2 — Criar a tabela (v2)

A tabela `franquia_1beat` é criada com todos os campos necessários para armazenar pedidos, recebimentos e informações da filial. Detalhes completos no capítulo 7.

Característica	Valor
Motor	InnoDB
Charset	utf8mb4
Chave primária	ID_PRODUTO_GRADE + COD_FILIAL + ID_FILIAL + ID_MOVIMENTACAO_PAI + CNPJ_FILIAL
Índice adicional	IDX_1BEAT_SKU (campo SKU)

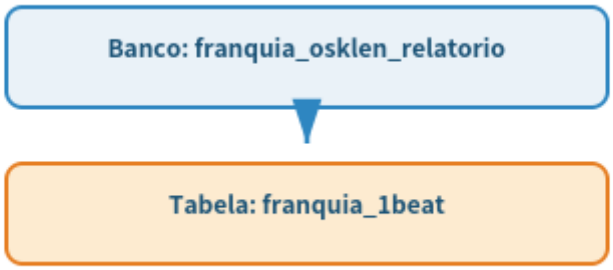


Figura 5 — Estrutura criada pelo instalador: um banco com uma tabela principal.

Etapa 3 — Migração de colunas (v3)

A etapa v3 evolui a tabela para o formato atual de controle de status e quantidades cortadas. Executa em transação e é idempotente (verifica colunas antes de alterar).

Ação	Detalhe
Adiciona `INATIVO`	Coluna int (0/1) que substitui o antigo enum `SITUACAO`
Migra dados	Registros com `SITUACAO = 'DESATIVO'` recebem `INATIVO = 1`
Remove `SITUACAO`	Enum obsoleto removido após migração
Remove `DATA_ATUALIZACAO`	Atualizações passam a ser rastreadas por `DATA_TRANSFERENCIA`
Adiciona `QTDE_CORTADA`	Quantidade cortada por item (decimal 10,2)

Etapa 4 — Confirmação e exclusão (v4)

A etapa v4 adiciona campos de controle para o consumo pelo OneBeat. Também executa em transação e é idempotente.

Ação	Detalhe
Adiciona `CONFIRMADO`	Flag int (0/1) — indica se o pedido está apto para o OneBeat
Adiciona `DATA_EXCLUSAO_MOVIMENTACAO`	Timestamp com a data de exclusão/cancelamento do pedido

Ambientes já instalados: se a instalação parou na v2, execute v3 e v4 para que a tabela tenha todas as colunas usadas pelo módulo atual.

Dependências

- Classe base `biblioteca` (framework ISNAPP)
- Classe `util` do token `12262a22-30ab-11e9-bb4c-127101af6b0d`
- Array `$requires` está vazio — sem dependências de outras bibliotecas versionadas
- Bancos do tenant já devem existir: `{banco}`, `{banco}_relatorio`, `{banco}_integrador`

Atenção: a instalação só precisa ser feita *uma vez* por ambiente. Executar novamente é seguro graças ao `IF NOT EXISTS`, mas não é necessário em rotina.

Arquitetura do sistema

O módulo é uma **biblioteca ISNAPP** (padrão de extensão do ERP) composta por três arquivos e integrada a múltiplos bancos MySQL.

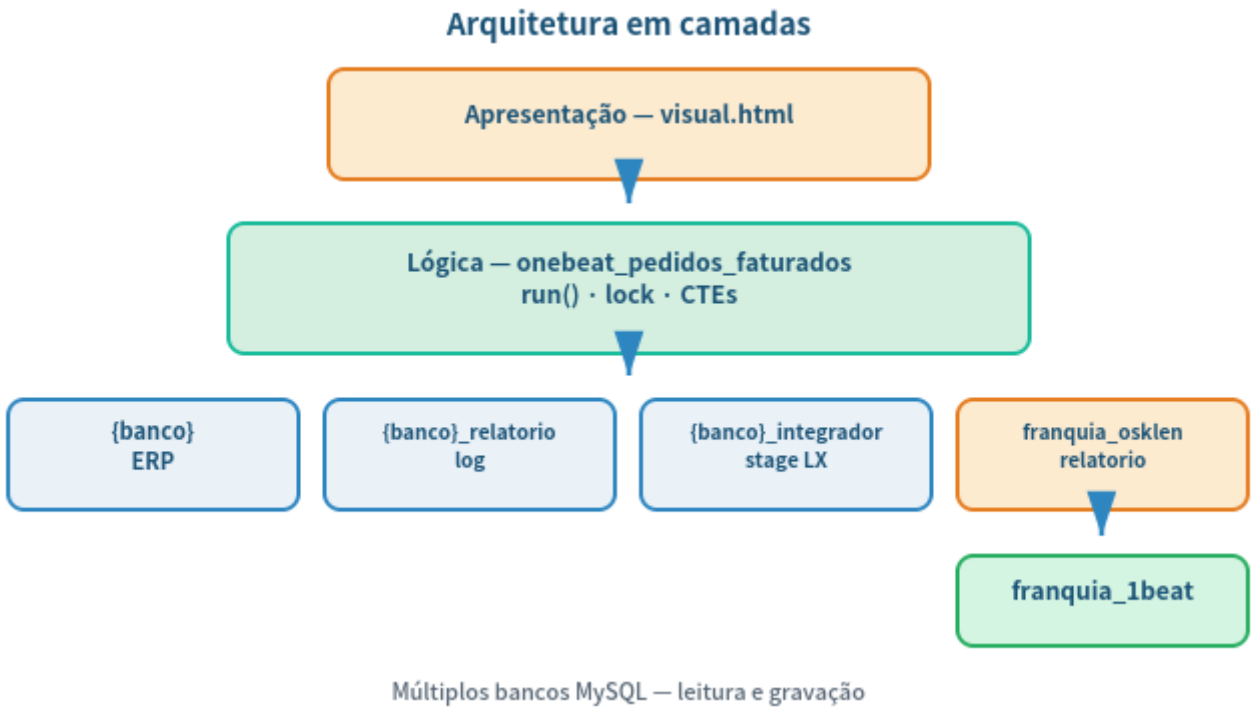


Figura 6 — Arquitetura em camadas: apresentação, lógica PHP e múltiplos bancos MySQL.

Bancos de dados envolvidos

Variável PHP	Nome do banco	Uso
`$banco`	Dinâmico (tenant)	Pedidos, itens, entidades, recebimentos
`$bancoRelatorio`	`{banco}_relatorio`	Tabela `log` com ponteiro de última execução
`$bancoIntegrador`	`{banco}_integrador`	Stage de integração LX (`stage_pedido_lx`)
`$bancoMasterRelatorio`	`franquia_osklen_relatorio`	Tabela de destino `franquia_1beat` (compartilhada)

Para entender sem jargão: o sistema lê informações de três "pastas" diferentes no banco de dados e grava o resultado organizado em uma "pasta central" que o OneBeat consulta.

Tabelas de origem (leitura)

Tabela	Banco	Papel
`movimentacao`	{banco}	Cabeçalho dos pedidos e recebimentos
`movimentacao_detalhe`	{banco}	Itens (produtos) de cada movimentação
`movimentacao_movimentacao`	{banco}	Vínculo pai-filho (pedido → recebimento)
`entidade`	{banco}	Dados da filial (loja)
`juridica`	{banco}	CNPJ da filial
`stage_pedido_lx`	{banco}_integrador	Integração com sistema LX (LEFT JOIN)
`log`	{banco}_relatorio	Controle de ponteiro (`procedure = 'franquia_1beat'`)

Classe principal

Classe: `onebeat_pedidos_faturados` extends `util`

Token: `6b5b2af8-ab25-4135-9551-b3cb3e055f1a`

CoreVersion: 3

Métodos públicos principais:

- `index()` — exibe visual.html
- `executar()` — execução principal (sincronização)
- `verificaLock()` — controle de concorrência
- `getPonteiroExecucao()` — última data processada
- `getPedidoUltimaVerificacao()` — data do pedido mais recente
- `getPedidosRecebidos()` — INSERT/UPDATE na tabela destino
- `atualizaPonteiroExecucao()` — grava novo ponteiro

Nota: o método `removePedidosStatus()` existe no código mas está *comentado*. A lógica de status (INATIVO, CONFIRMADO) é tratada diretamente no INSERT.

Mecanismo de lock (concorrência)

Para evitar que duas execuções rodem ao mesmo tempo, o módulo usa um **arquivo de lock** no diretório temporário do sistema operacional:

`/tmp/{banco}.onebeat_pedidos_faturados_executar.lock`

Usa `flock(LOCK_EX | LOCK_NB)` — lock exclusivo não-bloqueante. Se outro processo já estiver rodando, lança exceção: "*Ja tem outro processo em execucao, aguarde o termino*". O lock é liberado no bloco `finally` via `__liberaLock()`.

Fluxo de execução principal

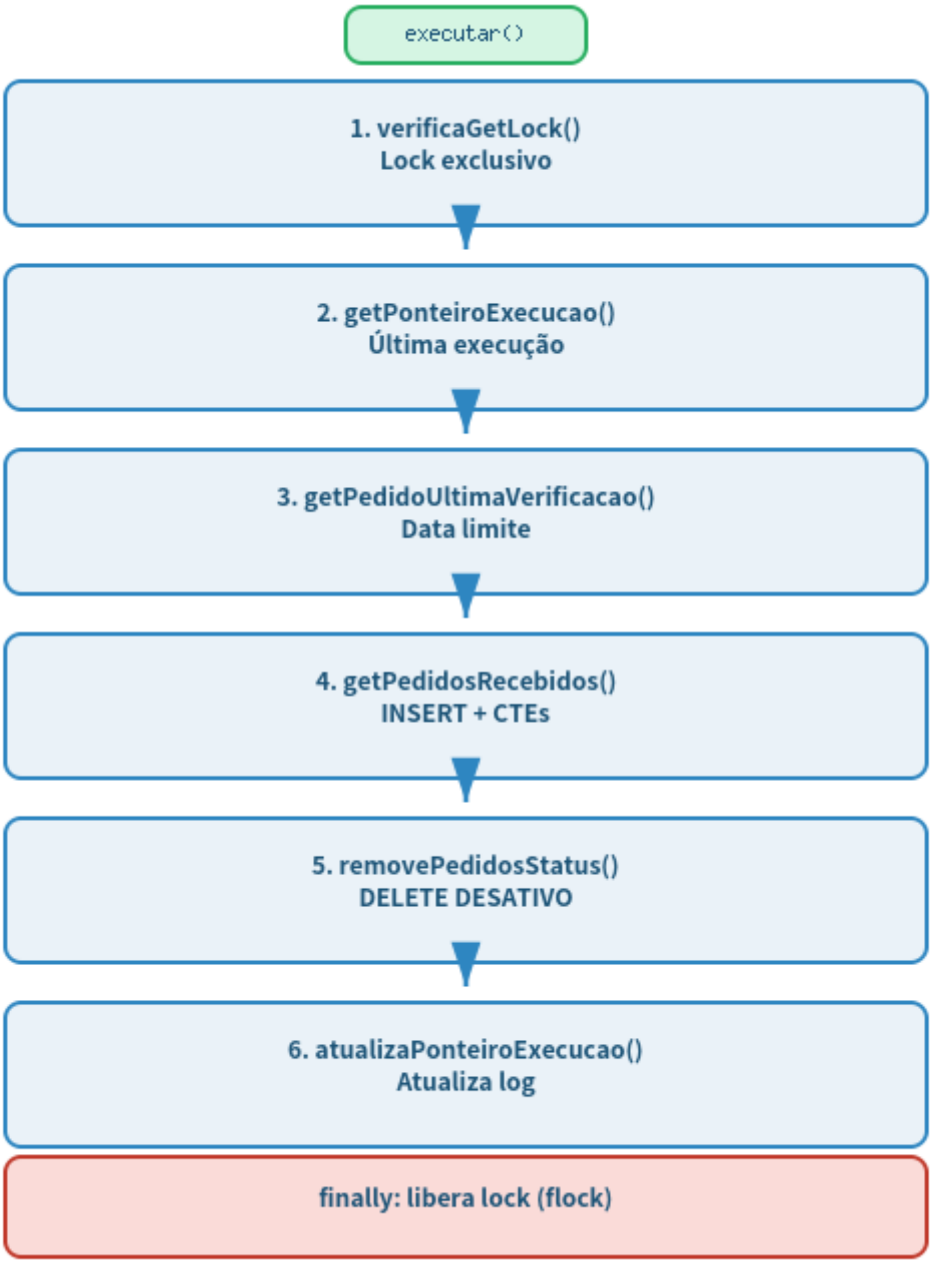


Figura 7 — Fluxograma completo do método executar(), com 5 etapas e liberação do lock no finally.

Detalhamento de cada etapa

#	Método	O que faz (simples)	O que faz (técnico)
1	`verificaLock()`	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 e status	INSERT com CTEs + ON DUPLICATE KEY UPDATE
5	`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 `executar()` 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.
- **"Erro ao inserir/atualizar pedidos recebidos na franquia_1beat."** — falha no INSERT/UPDATE principal.

Tratamento de erros: exceções no `executar()` são capturadas e enviadas a `error_byexception()`. Os métodos intermediários (`getPedidosRecebidos`, `atualizaPonteiroExecucao`) também possuem try/catch interno que retorna `null` em falha. O lock é sempre liberado no bloco `finally`.

Como os dados são processados

O método `getPedidosRecebidos()` executa um único comando SQL complexo que usa **CTEs** (Common Table Expressions — subconsultas nomeadas) para organizar os dados em etapas.

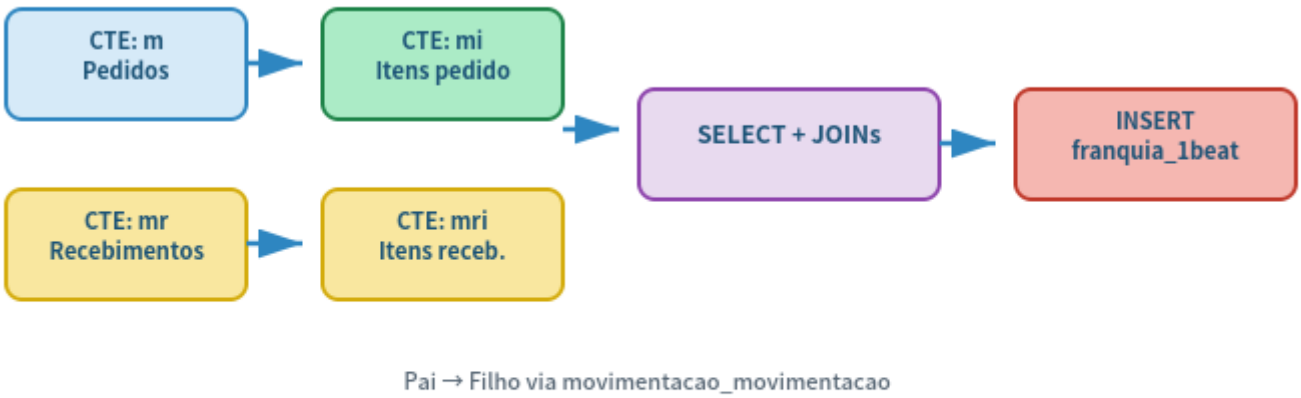


Figura 9 — As CTEs organizam pedidos, itens e recebimentos antes do INSERT final.

Etapa por etapa

1. CTE `m` — Pedidos no período

Seleciona movimentações do tipo PEDIDO alteradas entre o ponteiro e a data limite. Faz LEFT JOIN com `stage_pedido_lx` para integração LX.

2. CTE `mi` — Itens do pedido

Para cada pedido, soma as quantidades de cada produto (grade) nos itens ativos do pedido. Também captura o flag `pedido_corte_item` (usado no cálculo de QTDE_CORTADA e

QTDE_PENDENTE).

3. CTE `mr` — Recebimentos vinculados

Busca movimentações filhas (recebimentos) ligadas ao pedido pai, apenas com situação `Finalizado`.

4. CTE `mri` — Itens recebidos

Soma as quantidades recebidas de cada produto em cada recebimento vinculado ao pedido.

5. SELECT final — Montagem do registro

Junta pedido + itens pedidos + itens recebidos + dados da filial (nome, CNPJ, código).

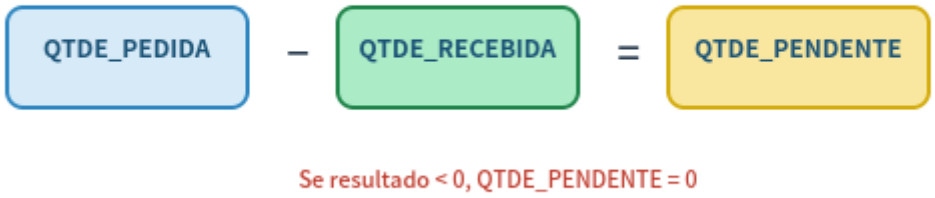


Figura 10 — Cálculo de pendente e cortada: depende do flag `pedido_corte_item` no item.

Campos calculados

Campo destino	Como é calculado
`QTDE_PEDIDA`	Soma dos itens do pedido (`mi.qtde`)
`QTDE_RECEBIDA`	Soma dos itens recebidos (`COALESCE(mri.qtde, 0)`)
`QTDE_CORTADA`	Se `pedido_corte_item = 'S'`: `MAX(0, pedida - recebida)`; senão: `0`
`QTDE_PENDENTE`	Se `pedido_corte_item = 'S'`: `0`; senão: `MAX(0, pedida - recebida)`
`ENTRADA_CONFIRMADA`	`1` se situação = FINALIZADO, senão `0`

`INATIVO`	`1` se situação = Cancelado, senão `0`
`CONFIRMADO`	`0` se Cancelado ou Aberto; `1` nas demais situações
`DATA_EXCLUSAO_MOVIMENTACAO`	`m.data_exclusao` se Cancelado; senão `NULL`
`DATA_TRANSFERENCIA`	`NOW()` no momento da gravação
`SKU`	Código do produto (`mi.codigo`)
`CHAVE_NFE`	Chave da nota fiscal do pedido

Flag `pedido_corte_item`: vem de `movimentacao_detalhe.pedido_corte_item` (prioriza o valor do recebimento, se existir). Quando igual a `'S'`, a diferença entre pedido e recebido vai para `QTDE_CORTADA` em vez de `QTDE_PENDENTE`.

Atualização de registros existentes

O INSERT usa `ON DUPLICATE KEY UPDATE`, ou seja:

- Se o registro **não existe** (chave primária nova) → insere.
- Se o registro **já existe** → atualiza: QTDE_RECEBIDA, QTDE_PEDIDA, QTDE_CORTADA, QTDE_PENDENTE, ENTRADA_CONFIRMADA, DATA_TRANSFERENCIA, INATIVO, CONFIRMADO e DATA_EXCLUSAO_MOVIMENTACAO (esta última só quando INATIVO = 1).

Chave primária: `ID_PRODUTO_GRADE + COD_FILIAL + ID_FILIAL + ID_MOVIMENTACAO_PAI + CNPJ_FILIAL`

Filtros aplicados

Filtro	Valor	Motivo
`modulo`	'PEDIDO'	Apenas movimentações de pedido
`tipo_estoque`	'PEDIDO'	Confirma tipo de estoque
`m.data`	> ponteiro AND <= limite	Janela incremental
`md.situacao`	'ATIVO'	Itens não excluídos
`md.id_produto_grade`	IS NOT NULL	Produtos com grade definida
`mr.situacao`	'Finalizado'	Recebimentos concluídos

Dicionário da coluna CONFIRMADO

Situação no ERP	CONFIRMADO
Cancelado ou Aberto	0
Demais situações (Confirmado, Recebido, Finalizado, etc.)	1

Estrutura da tabela de destino

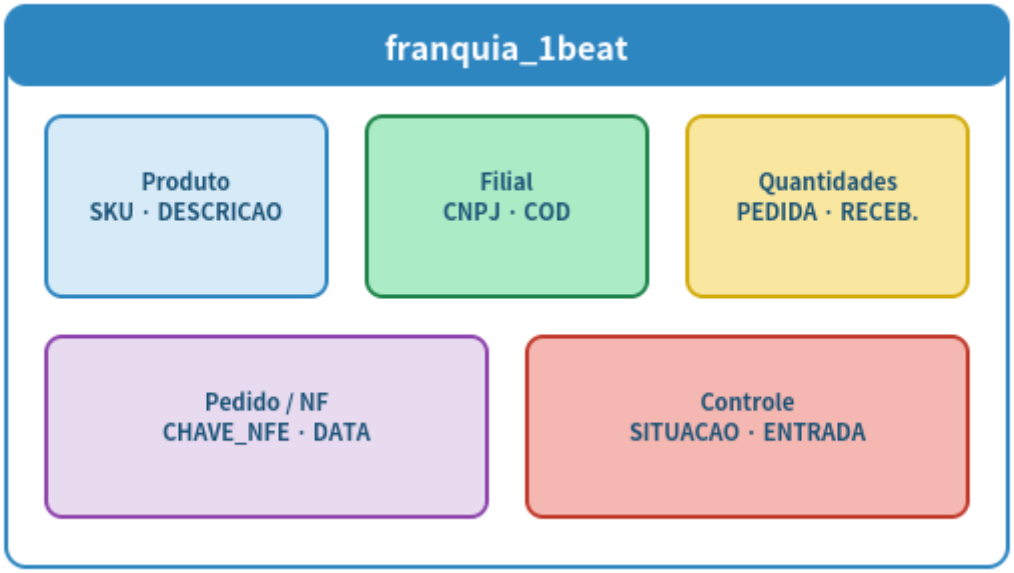


Figura 11 — Campos da tabela organizados por grupo: produto, filial, quantidades, pedido e controle.

Todos os campos

Campo	Tipo	Significado (simples)	Detalhe técnico
`SKU`	varchar(45)	Código do produto	Vem de `movimentacao_detalhe.codigo`
`CHAVE_NFE`	varchar(45)	Número da nota fiscal	Chave da NF-e do pedido
`ID_FILIAL`	int	Identificador da loja	FK para `entidade.id`
`NOME_FILIAL`	varchar(45)	Nome da loja	`entidade.nome`
`CNPJ_FILIAL`	varchar(50)	CNPJ da loja	`juridica.cnpj` — parte da PK

`COD_FILIAL`	varchar(45)	Código interno da loja	`entidade.codigo` — parte da PK
`QTDE_PEDIDA`	decimal(10,2)	Quanto foi pedido	Soma dos itens do pedido
`QTDE_CORTADA`	decimal(10,2)	Quanto foi cortado (não será mais entregue)	Quando `pedido_corte_item = 'S'`: pedida − recebida (mín. 0)
`QTDE_RECEBIDA`	decimal(10,2)	Quanto já chegou	Soma dos recebimentos finalizados
`QTDE_PENDENTE`	decimal(10,2)	Quanto falta	pedida − recebida (mín. 0), ou 0 se item cortado
`DESCRICAO`	longtext	Nome/descrição do produto	Texto do item do pedido
`ID_MOVIMENTACAO_PAI`	int	Número do pedido no sistema	ID da movimentação pai — parte da PK
`ID_PRODUTO_GRADE`	int	Identificador do produto com tamanho/cor	Parte da PK
`ENTRADA_CONFIRMADA`	int	Pedido já foi finalizado? (1=sim, 0=não)	Baseado em `m.situacao = 'FINALIZADO'`
`DATA_EMISSAO`	timestamp	Data em que o pedido foi emitido	`movimentacao.data_emissao`
`DATA_TRANSFERENCIA`	timestamp	Quando os dados foram enviados ao OneBeat	`NOW()` na gravação
`INATIVO`	int	Registro inativo? (1=sim, 0=não)	`1` se pedido Cancelado; substitui o antigo `SITUACAO`
`CONFIRMADO`	int	Pedido confirmado para o OneBeat? (1=sim, 0=não)	`0` se Cancelado/Aberto; `1` demais
`DATA_EXCLUSAO_MOVIMENTACAO`	timestamp	Data em que o pedido foi cancelado/excluído	`movimentacao.data_exclusao` se Cancelado
`EXTRA`	json	Campo reservado para dados extras	Não preenchido pelo módulo atualmente
`DATA_CRIACAO`	timestamp	Quando o registro foi criado	DEFAULT CURRENT_TIMESTAMP

Chave primária e índices



Figura 12 — Chave primária composta por 5 campos garante um registro único por produto/pedido/filial.

Índice adicional: `IDX_1BEAT_SKU` no campo `SKU` para consultas rápidas por código de produto.

Regras de negócio e situações

O módulo lê a situação do pedido na tabela `movimentacao` e traduz para o campo `SITUACAO` da tabela destino.

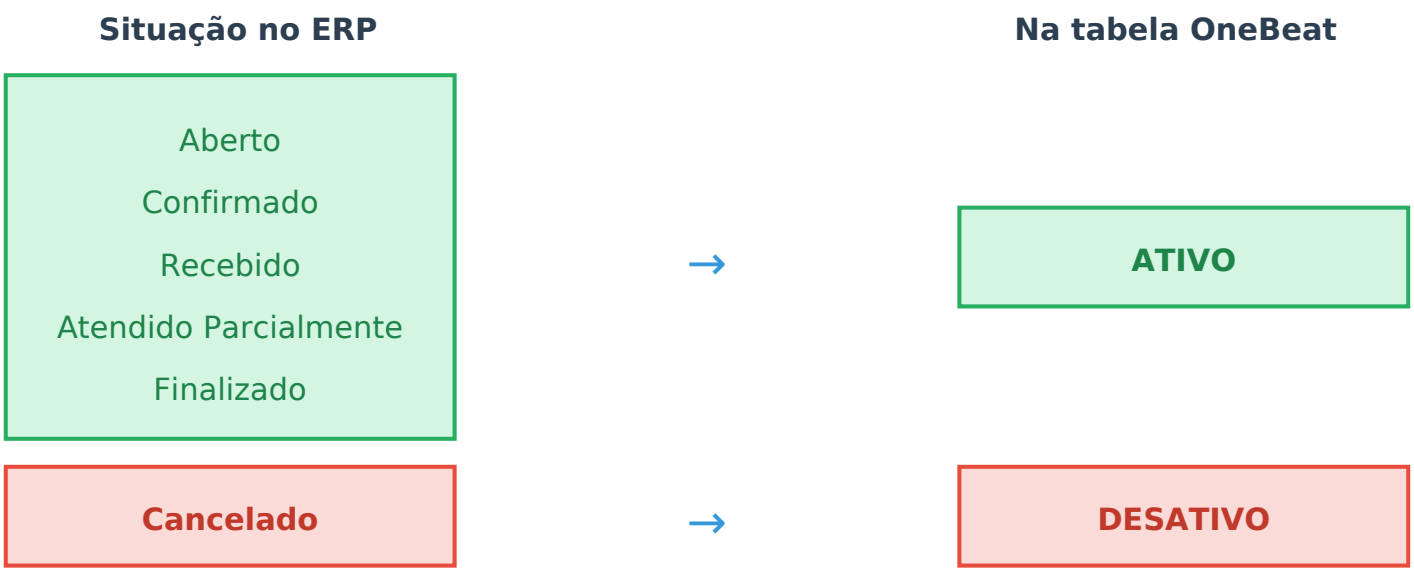


Figura 13 — Pedidos abertos e cancelados viram DESATIVO e são removidos da tabela após a sincronização.

Regra da quantidade pendente

Regra simples: o que falta = o que foi pedido menos o que já chegou. Se por algum motivo o recebido for maior que o pedido, o sistema considera que **nada falta** (zero).

Pedido	Recebido	Pendente	Explicação
10	7	3	Caso normal: faltam 3 unidades
10	10	0	Pedido completo
10	0	10	Nada recebido ainda

5	8	0	Recebido a mais: pendente nunca fica negativo

Entrada confirmada

O campo `ENTRADA_CONFIRMADA` indica se o pedido foi finalizado no ERP:

Valor	Significado	Condição
1	Entrada confirmada	`m.situacao = 'FINALIZADO'`
0	Entrada não confirmada	Qualquer outra situação

Recebimentos considerados

Para contar como "recebido", uma movimentação filha deve atender **todos** estes critérios:

- Estar vinculada ao pedido pai via `movimentacao_movimentacao`
- Ter situação `'Finalizado'` (com F maiúsculo)
- Ter itens ativos com `id_produto_grade` preenchido

Recebimentos em andamento ou cancelados **não entram** no cálculo de QTDE_RECEBIDA.

Processamento incremental



Figura 14 — O módulo foi projetado para rodar repetidamente, processando apenas o que mudou desde a última vez.

Concorrência — apenas uma execução por vez

Regra de segurança: se duas execuções tentarem rodar ao mesmo tempo, a segunda será bloqueada com a mensagem "*Ja tem outro processo em execucao, aguarde o termino*". Isso evita dados duplicados ou corrompidos.

O que acontece quando não há pedidos novos?

Se não existir nenhum pedido com data maior que o ponteiro, o método `getPedidoUltimaVerificacao()` lança uma exceção: "*Não foram encontrados pedidos para atualizar*". A execução é interrompida, o lock é liberado, e nada é alterado na tabela destino.

Ponteiro inicial

Na primeira execução (sem registro em `log`), o ponteiro padrão é `2000-01-01 00:00:00`, ou seja, todos os pedidos desde essa data serão considerados.

Glossário e referência rápida

Termo	Definição
OneBeat	Sistema externo de gestão de estoque e abastecimento que consome os dados desta tabela.
ERP / ISNAPP	Sistema principal da empresa onde os pedidos são criados e os recebimentos registrados.
Pedido faturado	Pedido de produtos emitido pela matriz para uma loja, com nota fiscal associada.
Recebimento	Registro de que a loja recebeu (ou parte de) um pedido. Fica ligado ao pedido original.
Filial / Loja	Unidade da franquia que faz o pedido e recebe os produtos.
SKU	Código único do produto (como um "RG" do item no estoque).
Grade	Variação do produto (tamanho, cor). Exemplo: camiseta azul tamanho M.
Ponteiro de execução	Data/hora da última sincronização bem-sucedida. O sistema só busca pedidos alterados depois dessa data.
Lock (trava)	Mecanismo que impede duas cópias do programa de rodar ao mesmo tempo.
CTE (Common Table Expression)	Técnica SQL que organiza a consulta em etapas nomeadas (m, mi, mr, mri).
ON DUPLICATE KEY UPDATE	Comando SQL que insere um registro novo ou atualiza se ele já existir (pela chave primária).
Biblioteca ISNAPP	Módulo extensível do ERP, identificado por um token UUID único.
Tenant / Banco dinâmico	Cada cliente/franquia tem seu próprio banco de dados ({banco}), separado dos demais.
Stage (integrador)	Área intermediária de integração com sistemas externos, como o LX.
NF-e / CHAVE_NFE	Nota Fiscal Eletrônica. A chave é o número de 44 dígitos que identifica a nota.
INATIVO	Flag int (0/1) que indica se o registro está inativo. Substitui o antigo campo `SITUACAO`. Valor 1 para pedidos cancelados.

CONFIRMADO	Flag int (0/1) que indica se o pedido está apto para consumo pelo OneBeat. Zero para pedidos cancelados ou abertos.
QTDE_CORTADA	Quantidade cortada de um item — parte do pedido que não será mais entregue.
pedido_corte_item	Flag nos itens do pedido (`movimentacao_detalhe`). Quando igual a `'S'`, a diferença entre pedido e recebido vai para QTDE_CORTADA em vez de QTDE_PENDENTE.
DATA_EXCLUSAO_MOVIMENTACAO	Data em que o pedido foi cancelado ou excluído no ERP. Preenchida apenas para pedidos cancelados.

URLs do módulo

Função	URL
Tela visual	`/bibliotecas/6b5b2af8-ab25-4135-9551-b3cb3e055f1a/onebeat_pedidos_faturados`
Execução (executar)	`/bibliotecas/6b5b2af8-ab25-4135-9551-b3cb3e055f1a/onebeat_pedidos_faturados/executar`
Instalação	`/bibliotecas/6b5b2af8-ab25-4135-9551-b3cb3e055f1a/onebeat_pedidos_faturados/install`

Checklist de implantação (T.I.)

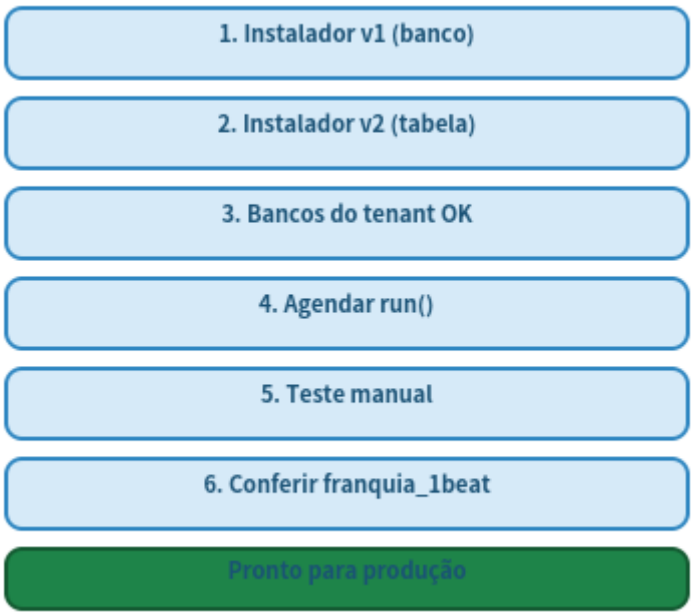


Figura 15 — Checklist recomendado para implantação do módulo.

Resumo dos arquivos

Arquivo	Classe	Responsabilidade
`controle.php`	`onebeat_pedidos_faturados`	Sincronização de pedidos
`instalador.php`	`instalador`	Criação de banco e tabela
`visual.html`	—	Mensagem de carregamento

Identificadores do módulo

Propriedade	Valor
Token	`6b5b2af8-ab25-4135-9551-b3cb3e055f1a`
Classe	`onebeat_pedidos_faturados`
CoreVersion	3
Banco destino	`franquia_osklen_relatorio`
Tabela destino	`franquia_1beat`
Procedure no log	`franquia_1beat`

Dúvidas frequentes:

- "*O sistema não atualizou meu pedido*" — verifique se o agendamento do `executar()` está ativo e se o pedido tem data maior que o ponteiro.
- "*Apareceu erro de processo em execução*" — aguarde a execução anterior terminar ou verifique se há lock órfão em `/tmp/{banco}.onebeat_pedidos_faturados_executar.lock`.
- "*Pedido cancelado ainda aparece*" — na próxima execução ele será marcado com `INATIVO = 1` e `CONFIRMADO = 0`. O registro permanece na tabela com a data de exclusão preenchida.
- "*Pedido aberto aparece no OneBeat*" — o registro existe na tabela, mas com `CONFIRMADO = 0`, indicando que ainda não está apto para consumo.

Índice da documentação

#	Capítulo	Arquivo
01	Introdução	`01_introducao.html`

02	O que faz	`02_o_que_faz.html`
03	Instalação	`03_instalacao.html`
04	Arquitetura	`04_arquitetura.html`
05	Fluxo de execução	`05_fluxo_execucao.html`
06	Processamento de dados	`06_processamento_dados.html`
07	Estrutura da tabela	`07_estrutura_tabela.html`
08	Regras de negócio	`08_regras_negocio.html`
09	Glossário	`09_glossario.html`