Svelto SaaS - Master Blueprint (Versão 2.3 - "The NSU Era")

Última Atualização: 18/01/2026
Status do Projeto: Milestone 2 (Conciliação Avançada) - CONCLUÍDO
Arquitetura: Monorepo NestJS + Next.js + Prisma ORM (PostgreSQL)
Versão Prisma: 5.22.0 (TRAVADA)

1. Visão Estratégica e Filosofia (Refinada)

O Svelto evoluiu de um simples integrador para um **Motor de Auditoria Forense**. A premissa de "Link por Referência Externa" provou-se falha devido à geração de hashes aleatórios pelo Checkout do Mercado Pago.

A Nova Verdade Tríplice

1. **Verdade Operacional (Transaction):**
   - Fonte: API MP /v1/payments/search.
   - Lógica: **Fee Explosion** (Separação granular de taxas, comissões e impostos no momento da ingestão).
   - **Atualização Crítica:** Ingestão via **Batch Processing** (Lotes de 50) para evitar timeouts de banco em grandes volumes.
2. **Verdade Contábil (ErpReceivable):**
   - Fonte: API Omie ListarContasReceber.
   - **Pivô de Ligação:** O campo erpNsu (e não mais o externalReference) é a chave dourada que armazena o ID da transação do Gateway.
3. **Verdade Financeira (Conciliation):**
   - Lógica: O motor de conciliação não confia cegamente. Ele verifica duplicidade, valida conta bancária (String Match) e aplica janelas temporais estendidas (-2/+7 dias).

2. Arquitetura de Dados (Schema v2.3)

2.1. Tabela Transaction (Ingestão Gateway)

- **Chave Primária Lógica:** gatewayId (ID do MP).
- **State Locking:** Registros com status MATCHED, CONCILIATED ou PAID_OUT são imutáveis pelo Sync automático, exceto em caso de **CHARGEBACK**.
- **Tipagem:** Todos os valores monetários são Decimal(19,4).

2.2. Tabela ErpReceivable (Espelho Omie)

- **Propósito:** Espelho local para performance (evita rate-limit da API Omie).
- **Campos Críticos:**
   - erpNsu: Armazena o gatewayId (Ex: "138845366921"). **Tipo: String**.

    - bankAccountId: ID da conta corrente no Omie. **Tipo: String** (Crucial para filtros).
    - amountValue: Valor do título.
- **Sincronização:** Suporta **Full Sync** (Data Fixa/Custom) e **Incremental** (baseado em lastSync).

2.3. Tabela Integration

- **Configuração Bancária:**
    - JSON: { "omieBankAccount": { "nCodCC": 7277285643 } }
    - *Atenção:* O nCodCC no JSON é Number, mas deve ser convertido para String ao filtrar contra ErpReceivable.bankAccountId.

3. Motores de Processamento (Services)

3.1. Engine de Ingestão MP (MercadoPagoService + IntegrationsController)

- **Estratégia:** Fetch paginado na API do MP.
- **Persistência:** Upsert em lotes de 50 registros (prisma.$transaction).
    - *Justificativa:* Evitar deadlocks e timeouts da Railway em cargas iniciais massivas (>2k registros).
- **Auto-Healing:** O Controller detecta disparidade de TenantID e realiza migração automática da integração se necessário.

3.2. Engine de Espelhamento Omie (OmieSyncService)

- **Estratégia:** do...while loop com payload JSON nativo.
- **Filtros Obrigatórios:** filtrar_apenas_alteracao = "S" para capturar inclusões e edições.
- **Mapeamento:**
    - Status CANCELADO se cancelado == 'S'.
    - Status RECEBIDO se valor_saldo_ado == 0 ou valor_recebido >= valor_documento.

3.3. Engine de Conciliação "NSU Hunter" (ConciliationService v15)

O algoritmo antigo de "Hard Match por Referência" foi depreciado. O novo fluxo é:

1. **Fase 1: Hard Match (NSU)**
    - Comparação Estrita: Transaction.gatewayId === ErpReceivable.erpNsu.
    - Sanitização: .trim() obrigatório em ambos os lados.
    - Filtro Bancário: Aplica filtro por bankAccountId (String) se configurado.
    - *Confiabilidade:* 100%.
2. **Fase 2: Fuzzy Match (Fallback)**
    - Critérios:
        - Mesmo Valor Monetário (amountValue).
        - Janela Temporal Estendida: Data do Evento -2 dias a +7 dias (Cobre atrasos de emissão no ERP).
        - Conta Bancária Correta.

- *Regra de Desambiguação:* Só aplica o match se houver **exatamente 1** candidato elegível. Se houver >1, loga aviso de ambiguidade e não concilia.

4. Regras de Negócio e Segurança

RN01 - Tipagem Estrita de Filtros

O PostgreSQL/Prisma não faz coerção automática de Number para String em cláusulas WHERE.

- **Regra:** Todo ID vindo de JSON ou Config deve ser convertido explicitamente (String(id).trim()) antes de ser usado em queries do Prisma.

RN02 - State Locking (Proteção de Trabalho)

Ao rodar um Sync (MP ou Omie):

- Se o registro já existe e tem status final (CONCILIATED), **ignorar atualização de status**.
- Exceção: Se o status novo for CHARGEBACK (Desastre), atualizar imediatamente.

RN03 - Agendamento (Scheduler)

- Implementado SyncScheduler (@nestjs/schedule).
- Execução: A cada 1 hora.
- Modo: Incremental (usa date_last_updated).

5. Guia de Operação (Production)

Comandos de Carga Inicial (PowerShell)

1. Atualizar Token (se 403 Forbidden):

```
$body = @{ token = "APP_USR-NOVO-TOKEN..." } | ConvertTo-Json
Invoke-RestMethod -Uri "http://localhost:3000/integrations/{ID}" -Method Patch -Body $body -ContentType "application/json"
```

2. Carga Inicial MP (Batch 50):

```
$body = @{ initialDate = "2025-01-01" } | ConvertTo-Json
Invoke-RestMethod -Uri "http://localhost:3000/integrations/{ID}/sync" -Method Post -Body $body -ContentType "application/json"
```

3. Executar Matchmaker (NSU Hunter):

```
Invoke-RestMethod -Uri "http://localhost:3000/integrations/{ID}/match" -Method Post
```

6. Próximos Passos (Milestone 3)

1. **Dashboard de Divergências:** UI para mostrar casos onde o Matchmaker falhou (ex: valor diferente por centavos).
2. **Interface de Resolução Manual:** Permitir que o usuário force o vínculo de transações ambíguas.
3. **Baixa Automática:** Implementar a escrita no Omie (LancarRecebimento) para transações com status MATCHED.