Svelto SaaS - Master Blueprint (Versão 2.2 - "Connected Era")

Última Atualização: 15/01/2026
Status do Projeto: Milestone 1 (Concluído) | Milestone 2 (Em Execução)
Arquitetura: Monorepo NestJS + Next.js + Prisma ORM (PostgreSQL)

1. Visão Estratégica e Filosofia

O Svelto é um middleware de Engenharia Financeira Forense. Ele não apenas conecta pontas; ele espelha, audita e reconcilia dados entre Gateways (Mercado Pago) e ERPs (Omie).

A Verdade Tríplice (Triple Source of Truth)

1. **Verdade Operacional (Venda):**
   - Fonte: API MP /v1/payments.
   - Armazenamento: Tabela Transaction.
   - Lógica: "Fee Explosion" (Separação de MDR, Financing, Shipping).
2. **Verdade Contábil (ERP):**
   - Fonte: API Omie ListarContasReceber.
   - Armazenamento: Tabela ErpReceivable (Novo - Espelhamento).
   - Lógica: O ERP é a autoridade fiscal. O Svelto deve encontrar o par correspondente aqui.
3. **Verdade Financeira (Liquidação):**
   - Fonte: Relatório MP /v1/settlement/report.
   - Armazenamento: Tabela Settlement.
   - Lógica: Confirmação do crédito bancário efetivo.

2. Arquitetura de Dados (Schema v2.1)

2.1. Tabela Transaction (Ingestão Gateway)

Representa o fato gerador no Gateway.

- **Chave Única:** integrationId + gatewayId (MP ID).
- **State Locking:** Registros com status CONCILIATED ou PAID_OUT são imutáveis pelo Sync automático, exceto em caso de CHARGEBACK.
- **Colunas Críticas:**
  - amountGross: Valor pago pelo cliente.
  - amountMdrFee: Taxa administrativa explícita.
  - amountFinancingFee: Custo de parcelamento (lojista).
  - amountNetGateway: O líquido esperado.
  - externalReference: O ID do Pedido (Elo de ligação).

2.2. Tabela ErpReceivable (Ingestão ERP) - *NOVO*

Espelho local dos títulos do Omie para performance de conciliação.

- **Chave Única:** integrationId + erpId (nCodTitulo).
- **Propósito:** Permitir algoritmos de "Match" local (SQL) sem depender da latência da API do Omie.
- **Colunas Críticas:**
  - erpExternalRef: Número do Pedido (Chave de busca).
  - amountValue: Valor do título.
  - status: "EM ABERTO" ou "RECEBIDO".

2.3. Tabela Integration (Configurações)

- **Security:** credentials armazena tokens cifrados com AES-256 (Envelope Encryption).
- **Settings:** JSON que armazena o "De-Para" bancário.

```
{
  "omieBankAccount": {
    "nCodCC": 123456,
    "descricao": "Banco Itaú"
  }
}
```

3. Motores de Processamento (Services)

3.1. Engine de Ingestão (Smart Sync)

- **Provider:** MercadoPagoService.
- **Lógica:** Paginação automática (loop while hasMore).
- **Inteligência:** Antes de salvar, verifica o status atual no banco. Se o usuário conciliou manualmente, o Sync respeita e não sobrescreve (exceto desastres).

3.2. Engine de Espelhamento (Omie Sync)

- **Provider:** OmieSyncService.
- **Lógica:** Busca títulos "A VENCER" e "VENCIDOS" recentes.
- **Objetivo:** Manter o Svelto atualizado com a realidade do ERP (se alguém baixou um título manualmente no Omie, o Svelto deve saber).

3.3. Engine de Conciliação (MatchMaker) - *EM CONSTRUÇÃO*

- **Algoritmo:**
  1. **Match Perfeito:** Transaction.externalReference == ErpReceivable.erpExternalRef AND Valor == Valor.
  2. **Match Inteligente:** Mesma data (janela de 3 dias) + Mesmo Valor + Mesmo Cliente (Busca por CPF/CNPJ).

- **Ação:** Quando confirmado, chama OmieService.BaixarContaReceber.

4. Segurança e Infraestrutura

- **Criptografia:** SecurityService utiliza crypto nativo do Node.js com algoritmo aes-256-gcm. Chaves nunca são salvas em texto plano.
- **API Omie:** Todas as chamadas são POST (RPC). O Svelto trata os erros lógicos (faultstring) que vêm com status HTTP 200.
- **Rate Limiting:** O uso de *Data Mirroring* protege nossa cota de API no Omie, pois as consultas pesadas são feitas no banco local.

5. Guia de Setup e Comandos (Dev Environment)

Infraestrutura Local:

● Node.js: v20 (LTS).

● Gerenciador: npm.

● Banco: PostgreSQL (via Railway).

Comandos de Rotina:

● Atualizar Schema do Banco:

○ Cuidado: No Windows, evite npx global.

○ Comando: .\node_modules\.bin\prisma db push

● Ver Dados (Prisma Studio):

○ Comando: .\node_modules\.bin\prisma studio

● Rodar Backend (API):

○ Pasta: apps/api

○ Comando: npm run start:dev

○ Porta: http://localhost:3000

● Rodar Frontend (Web):

○ Pasta: apps/web

○ Comando: npm run dev

○ Porta: http://localhost:3001

Variáveis de Ambiente (.env na API):

DATABASE_URL="postgresql://postgres:gEEaiOzJTaNBdjAMnVjmXCcxBzeqavJL@shortline.proxy.rlwy.net:19521/railway"

REDIS_URL="redis://default:RgptNHHWRPPeUoJMFncseqxbMCPWhOAv@caboose.proxy.rlwy.net:29678" MASTER_KEY="<GERAR_UMA_HASH_DE_32_BYTES_HEX>"