Svelto SaaS - Master Blueprint (Versão 2.0 - Full Spec)

Última Atualização: 10/01/2026
Status do Projeto: Milestone 0 (Fundação) Concluído.
Versão do Prisma: 5.22.0 (LTS Travada)
Arquitetura: Monorepo NestJS + Next.js

1. Visão Estratégica e Filosofia

O Svelto é um middleware de Engenharia Financeira Forense.
Diferente de integradores comuns que apenas copiam dados de A para B, o Svelto audita a integridade do dinheiro.
A Verdade Tríplice (Triple Source of Truth):

1. **Verdade Operacional (Venda):** "O cliente passou o cartão."
   - *Fonte:* API MP /v1/payments.
   - *Entidade Svelto:* Transaction.
2. **Verdade Financeira (Liquidação):** "O MP liberou o dinheiro na conta virtual."
   - *Fonte:* Relatório MP /v1/settlement/report.
   - *Entidade Svelto:* Settlement.
3. **Verdade Bancária (Caixa):** "O dinheiro caiu no Itaú."
   - *Fonte:* Extrato OFX e Payouts MP.
   - *Entidade Svelto:* Payout.

2. Dicionário de Dados e Mapeamento (Deep Dive)

2.1. Tabela Transaction (A Venda)

Representa o fato gerador. É imutável após a criação, exceto pelo status.

Campo Svelto	Fonte Mercado Pago (JSON)	Descrição Técnica
gatewayId	id	ID único da transação (ex: 123456789). Chave de busca.
externalReference	external_reference	**CRÍTICO.** O ID do pedido no ERP. Sem isso, não há conciliação automática.
authorizationCode	authorization_code	Código NSU da adquirente.

		Usado para disputas judiciais.
dateEvent	date_created	Data da competência contábil (DRE). Convertida para UTC.
dateEstimated	money_release_date	Previsão de quando o dinheiro *deveria* cair. Usado para Fluxo de Caixa Projetado.
amountGross	transaction_amount	Valor que o cliente pagou. Base de cálculo.
amountNetGateway	transaction_details.net_received_amount	O valor líquido prometido pelo MP na hora da venda.
amountMdrFee	fee_details[type='mercado pago_fee'].amount	Taxa administrativa/comissão.
amountFinancingFee	fee_details[type='financing_fee'].amount	Custo do parcelamento "sem juros" (absorvido pelo lojista).
payerDocNumber	payer.identification.number	CPF/CNPJ do comprador. Usado para regra Anti-Fraude (Cross-check com ERP).

2.2. Tabela Settlement (O Extrato Virtual)

Representa o crédito efetivo na conta Mercado Pago.

Campo Svelto	**Fonte Mercado Pago (Relatório)**	**Descrição Técnica**
gatewayReference	SOURCE_ID (em linhas de Liquidação)	ID do lote ou referência interna do movimento financeiro.
dateSettled	DATE_CREATED (do	Data real da

	relatório)	disponibilização do recurso.
amountTotal	NET_CREDIT_AMOUNT	Valor somado que entrou na conta virtual.

2.3. Tabela Reconciliation (O Vínculo)

Tabela pivô N:N que resolve o problema do parcelamento.

- *Cenário:* Venda de R$ 1.000,00 em 10x.
- *Resultado:* Teremos 1 Transaction ligada a 10 Settlements diferentes através de 10 registros na tabela Reconciliation.

3. Máquina de Estados (Workflow)

O campo status da tabela Transaction rege o comportamento do sistema.

1. **PENDING (Inicial):**
   - A transação foi importada via Webhook ou API /v1/payments.
   - *Ação:* Sistema busca no ERP (Omie) um pedido com externalReference.
2. **MATCHED (Vínculo Operacional):**
   - Encontramos o pedido no ERP (nCodTitulo).
   - *Ação:* Shadow Accounting calcula as taxas esperadas.
3. **AUDITED (Conferência):**
   - O cálculo bateu (Diferença < R$ 0,05).
   - *Ação:* Aguardando o dinheiro cair (Liquidação).
4. **CONCILIATED (Liquidação Virtual):**
   - Chegou o Relatório de Liquidação confirmando o crédito.
   - *Ação:* O Svelto envia o comando de **Baixa** para o Omie (BaixarContaReceber).
5. **PAID_OUT (Saque Real):**
   - O dinheiro foi transferido para o banco físico.
   - *Ação:* Conciliação Bancária finalizada.
6. **DIVERGENT (Erro):**
   - Taxas não batem ou valor líquido diferente. Exige intervenção humana.
7. **CHARGEBACK (Disputa):**
   - Dinheiro bloqueado preventivamente.

4. Protocolos de Segurança (Security Specs)

4.1. Envelope Encryption (Credenciais)

Não salvamos chaves de API limpas.

1. **DEK (Data Encryption Key):** Geramos uma chave aleatória única para cada registro.

2. **Cifragem:** Usamos a DEK para cifrar o Token do cliente (AES-256-GCM).
3. **KEK (Key Encryption Key):** Usamos uma chave Mestra (variável de ambiente MASTER_KEY) para cifrar a DEK.
4. **Persistência:** Salvamos no banco { ciphertext, encrypted_dek, iv, tag }.

4.2. Idempotência (Omie)

Para evitar baixar o mesmo título duas vezes:

1. **Check:** Chamar ConsultarContaReceber passando nCodTitulo.
2. **Verify:** Se status for "LIQUIDADO", parar e marcar sucesso.
3. **Act:** Se status "ABERTO", enviar BaixarContaReceber.

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>"

6. Roteiro de Implementação (Roadmap)

Fase 1: Ingestão e Segurança (Atual)

- [ ] **Service de Criptografia:** Implementar AES-256.
- [ ] **Connector MercadoPago:** Autenticação e busca (/v1/payments/search).
- [ ] **Webhook Receiver:** Receber notificações de novas vendas.

Fase 2: Auditoria e ERP (Próxima)

- [ ] **Engine de Regras:** Calcular taxas esperadas.
- [ ] **Connector Omie:** Implementar Consultar e Baixar.
- [ ] **Conciliador:** Job que cruza Vendas x Lotes.