```
//
------------------------------------------------------------------------
-------
// SVELTO SAAS - SCHEMA DE BANCO DE DADOS (VERSION 2.0 - FULL
COMPLIANCE)
// Baseado no "Manual Definitivo de Engenharia de Dados Financeiros
MP"
//
------------------------------------------------------------------------
-------

generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

//
------------------------------------------------------------------------
-------
// 1. NÚCLEO ADMINISTRATIVO (Auth & Tenancy)
//
------------------------------------------------------------------------
-------

model Tenant {
  id            String   @id @default(uuid())
  name          String
  document      String?  // CNPJ/CPF
  settings      Json?    // Ex: { "fiscalCloseDay": 25 }

  createdAt     DateTime @default(now())
  updatedAt     DateTime @updatedAt

  users         User[]
```

```
  integrations  Integration[]
  transactions  Transaction[]
  settlements   Settlement[]
  payouts       Payout[]        // Novo: Saques para conta bancária
  auditLogs     AuditLog[]

  @@map("tenants")
}

model User {
  id            String   @id @default(uuid())
  tenantId      String

  name          String
  email         String   @unique
  passwordHash  String

  role          UserRole @default(MEMBER)

  createdAt     DateTime @default(now())
  updatedAt     DateTime @updatedAt

  tenant        Tenant     @relation(fields: [tenantId], references:
[id])
  auditLogs     AuditLog[]

  @@index([tenantId])
  @@map("users")
}

enum UserRole {
  ADMIN
  MANAGER
  MEMBER
}
```

```
//
------------------------------------------------------------------------
-------
// 2. INTEGRAÇÕES (Conectores)
//
------------------------------------------------------------------------
-------

model Integration {
  id            String            @id @default(uuid())
  tenantId      String

  name          String
  provider      IntegrationProvider

  // Security: Envelope Encryption (Ciphertext + IV + KeyID)
  credentials   Json

  // Configs: { "plan": "D+14", "autoConciliate": true,
"bankAccountId": 123 }
  settings      Json?

  status        IntegrationStatus @default(ACTIVE)
  lastSyncAt    DateTime?

  createdAt     DateTime          @default(now())
  updatedAt     DateTime          @updatedAt

  tenant        Tenant        @relation(fields: [tenantId],
references: [id])
  transactions  Transaction[]
  payouts       Payout[]

  @@index([tenantId])
  @@map("integrations")
}

enum IntegrationProvider {
```

```
  MERCADOPAGO
  OMIE
}

enum IntegrationStatus {
  ACTIVE
  ERROR
  PAUSED
}

//
------------------------------------------------------------------------
-------
// 3. CORE OPERACIONAL (A Venda / Checkout)
//
------------------------------------------------------------------------
-------

model Transaction {
  id                  String   @id @default(uuid())
  tenantId            String
  integrationId       String

  // --- Identificadores (A Tríade de Ouro) ---
  gatewayId           String   // MP ID (Source of Truth do Gateway)
  externalReference   String?  // ID do Pedido (Sua Loja/ERP).
CRÍTICO.
  authorizationCode   String?  // NSU/Auth Code

  // --- Dados ERP ---
  erpId               String?  // nCodTitulo (Omie)
  erpStatus           String?  // "ABERTO", "LIQUIDADO"

  // --- Datas ---
  dateEvent           DateTime // Data da Venda (Competência)
  dateEstimated       DateTime? // Previsão de Liquidação (Money
Release Date)
```

```
  // --- Engenharia Financeira (A Equação do Líquido) ---
  // V_liq = V_bruto - (MDR + Financiamento + Frete + Impostos) +
Ajustes

  amountGross         Decimal  @db.Decimal(19, 4) // Venda Bruta

  amountMdrFee        Decimal  @default(0) @db.Decimal(19, 4) // Taxa
Adm (mercadopago_fee)
  amountFinancingFee  Decimal  @default(0) @db.Decimal(19, 4) // Custo
Parcelamento (financing_fee)
  amountShippingFee   Decimal  @default(0) @db.Decimal(19, 4) // Custo
Envio (shipping_fee)
  amountTaxes         Decimal  @default(0) @db.Decimal(19, 4) //
Retenções (taxes_amount)
  amountCoupon        Decimal  @default(0) @db.Decimal(19, 4) //
Ajustes/Descontos (coupon_amount)

  amountNetGateway    Decimal  @db.Decimal(19, 4) // O que sobrou (Net
Received)

  // Shadow Accounting (Nosso cálculo para auditoria)
  amountNetCalculated Decimal? @db.Decimal(19, 4)

  // --- Dados do Pagador (Anti-Fraude) ---
  // Usado para bater com o cadastro do ERP. Se divergir -> Alerta.
  payerDocType        String?  // CPF, CNPJ
  payerDocNumber      String?  // Normalizado (apenas números)
  payerEmail          String?
  payerName           String?

  // --- Classificação & Status ---
  paymentMethod       String   // credit_card, pix, boleto
  installments        Int      @default(1)

  status              TransactionStatus @default(PENDING)
  statusDetail        String?           // Ex:
"cc_rejected_high_risk", "accredited"
```

```
  // --- Gestão de Disputas (Chargebacks) ---
  isDisputed          Boolean  @default(false) // Se caiu em
contestação
  disputeCoverage     Boolean  @default(false) // Se o MP cobriu o
prejuízo (coverage_applied)

  createdAt           DateTime @default(now())
  updatedAt           DateTime @updatedAt

  tenant              Tenant      @relation(fields: [tenantId],
references: [id])
  integration         Integration @relation(fields: [integrationId],
references: [id])

  reconciliations     Reconciliation[]

  @@unique([integrationId, gatewayId])
  @@index([tenantId, dateEvent])
  @@index([tenantId, externalReference])
  @@index([tenantId, payerDocNumber]) // Busca rápida por fraudadores
  @@map("transactions")
}

enum TransactionStatus {
  PENDING
  MATCHED       // Vinculado ao ERP
  AUDITED       // Taxas conferidas
  CONCILIATED   // Dinheiro liberado na Conta Virtual (Settled)
  PAID_OUT      // Dinheiro sacado para o Banco (Payout)
  DIVERGENT
  IGNORED
  CHARGEBACK    // Bloqueado por disputa
}

//
------------------------------------------------------------------------
-------
// 4. CORE FINANCEIRO (O Dinheiro)
```

```
//
------------------------------------------------------------------------
-------

// Nível 1: Liberação na Conta Virtual (Settlement / Release Report)
// Representa o dinheiro saindo do "Limbo" e ficando "Disponível" no
MP.
model Settlement {
  id               String   @id @default(uuid())
  tenantId         String

  gatewayReference String?  // ID do Lote/Relatório no MP
  dateSettled      DateTime // Data da liberação

  amountTotal      Decimal  @db.Decimal(19, 4)

  tenant           Tenant           @relation(fields: [tenantId],
references: [id])
  reconciliations  Reconciliation[] // Quais vendas compõem este lote?
  payoutId         String?          // Esse dinheiro já foi sacado?
  payout           Payout?          @relation(fields: [payoutId],
references: [id])

  createdAt        DateTime @default(now())

  @@index([tenantId, dateSettled])
  @@map("settlements")
}

// Nível 2: Saque para o Banco (Withdrawal / Payout)
// Representa a transferência da Conta Virtual MP para o
Itaú/Bradesco.
// É ISSO que bate com o OFX do banco.
model Payout {
  id               String   @id @default(uuid())
  tenantId         String
  integrationId    String
```

```
  externalReference String? // ID do Payout no MP (withdrawal_id)

  dateEvent        DateTime // Data do saque
  amount           Decimal  @db.Decimal(19, 4) // Valor sacado

  bankAccount      String?  // "Itaú **** 1234" (Metadado)

  status           PayoutStatus @default(PENDING)

  tenant           Tenant      @relation(fields: [tenantId],
references: [id])
  integration      Integration @relation(fields: [integrationId],
references: [id])

  // Um saque é composto por vários Settlements (liberações
acumuladas)
  settlements      Settlement[]

  createdAt        DateTime @default(now())
  updatedAt        DateTime @updatedAt

  @@map("payouts")
}

enum PayoutStatus {
  PENDING
  CONFIRMED_BANK // Batido com o OFX
  ERROR
}

// Tabela Pivô (Venda -> Liberação)
model Reconciliation {
  id              String   @id @default(uuid())
  transactionId   String
  settlementId    String

  amountCovered   Decimal  @db.Decimal(19, 4)
```

```
  transaction     Transaction @relation(fields: [transactionId],
references: [id])
  settlement      Settlement  @relation(fields: [settlementId],
references: [id])

  createdAt       DateTime @default(now())

  @@map("reconciliations")
}

//
------------------------------------------------------------------------
-------
// 5. AUDITORIA (JSONB Diff)
//
------------------------------------------------------------------------
-------

model AuditLog {
  id          String   @id @default(uuid())
  tenantId    String
  userId      String?

  action      String
  resource    String
  resourceId  String

  diff        Json?

  createdAt   DateTime @default(now())

  tenant      Tenant   @relation(fields: [tenantId], references: [id])
  user        User?    @relation(fields: [userId], references: [id])

  @@index([tenantId, createdAt])
  @@map("audit_logs")
}
```