```
//
// SVELTO SAAS SCHEMA DE BANCO DE DADOS (VERSION 2.1 - COM MÓDULO
OMIE)
// 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[]
```

```
  auditLogs       AuditLog[]
  erpReceivables  ErpReceivable[] // Novo: Títulos do Omie

  @@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: { "omieBankAccount": { "nCodCC": 123 }, "plan": "D+14" }
  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[]
  erpReceivables ErpReceivable[]

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

enum IntegrationProvider {
  MERCADOPAGO
  OMIE
}

enum IntegrationStatus {
  ACTIVE
  ERROR
  PAUSED
```

```
}

//
================================================================================
=======
// 3. CORE OPERACIONAL (A Venda / Checkout - Mercado Pago)
//
================================================================================
=======

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 (Links)
  erpId              String? // nCodTitulo (Omie) - Se já conciliado
  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
  amountFinancingFee Decimal  @default(0) @db.Decimal(19, 4) // Custo
Parcelamento
```

```
  amountShippingFee  Decimal  @default(0) @db.Decimal(19, 4) // Custo
Envio
  amountTaxes        Decimal  @default(0) @db.Decimal(19, 4) //
Retenções
  amountCoupon       Decimal  @default(0) @db.Decimal(19, 4) //
Ajustes/Descontos

  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)
  payerDocType    String? // CPF, CNPJ
  payerDocNumber  String? // Normalizado
  payerEmail      String?
  payerName       String?

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

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

  // Gestão de Disputas
  isDisputed      Boolean @default(false)
  disputeCoverage Boolean @default(false)

  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])
  @@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)
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[]

  payoutId         String?
  payout           Payout?  @relation(fields: [payoutId], references:
[id])

  createdAt DateTime @default(now())

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

// Nível 2: Saque para o Banco (Withdrawal / Payout)
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)
  bankAccount       String?  // "Itaú **** 1234" (Metadado)

  status            PayoutStatus @default(PENDING)

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

  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. MÓDULO ERP (Espelhamento Omie)
//
================================================================================
=======

// Espelho dos Títulos do ERP (Omie)
model ErpReceivable {
  id              String   @id @default(uuid())
  tenantId        String
  integrationId   String   // Link com a integração Omie

  // Chaves do Omie
```

```
  erpId           String   // codigo_lancamento_omie
  erpDocNumber    String?  // numero_documento_fiscal ou cNumTitulo
  erpExternalRef  String?  // numero_pedido (Chave de ligação com MP)

  // Valores
  amountValue     Decimal  @db.Decimal(19, 4) // valor_documento

  // Datas
  dateDue         DateTime // data_vencimento
  dateEmission    DateTime // data_emissao
  datePrevision   DateTime // data_previsao

  // Status
  status          String   // "EM ABERTO", "LIQUIDADO", "CANCELADO"

  // Dados do Cliente (Para inteligência)
  customerId      String?  // codigo_cliente_fornecedor
  customerName    String?  // (Opcional, se quisermos cachear)

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

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

  @@unique([integrationId, erpId]) // Evita duplicidade do mesmo
título
  @@index([tenantId, erpExternalRef]) // Índice vital para o Match
  @@map("erp_receivables")
}

//
================================================================================
=======
// 6. 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")
}
```