Projeto Sistema de Farmácia

Schema Prisma Definitivo (Base de Produção)

Inclui: multi-loja + estoque central, lotes/validade, transferências, caixa e COGS gravado no item de venda.

Arquivo gerado

schema.prisma

Conteúdo

```
// ============================================================
// PHARMA — Schema Prisma Definitivo (Base de Produção)
// Multi-loja + Estoque Central + Lotes/Validade + COGS gravado
// ============================================================

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

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

// ------------------------------
// ENUMS
// ------------------------------
enum StoreType {
  CENTRAL
  LOJA
}

enum SaleStatus {
  DRAFT
  CONFIRMED
  PAID
  CANCELED
  REFUNDED
}

enum SaleChannel {
  BALCAO
  DELIVERY
}

enum PaymentMethod {
  DINHEIRO
  PIX
  CARTAO
}

enum CashMovementType {
  RECEBIMENTO
  SANGRIA
  SUPRIMENTO
  ESTORNO
  AJUSTE
}

enum InventoryMovementType {
  IN
```

```
  OUT
  ADJUST_POS
  ADJUST_NEG
  TRANSFER_OUT
  TRANSFER_IN
}

enum TransferStatus {
  DRAFT
  SENT
  RECEIVED
  CANCELED
}

// ------------------------------
// CORE: LOJAS
// ------------------------------
model Store {
  id        String   @id @default(uuid())
  name      String
  type      StoreType
  active    Boolean  @default(true)
  createdAt DateTime @default(now())

  // relações
  accessUsers     StoreUser[]
  sales           Sale[]
  inventoryLots   InventoryLot[]
  inventoryMoves  InventoryMovement[]
  cashSessions    CashSession[]
  deliveries      Delivery[]
  transfersOrigin StockTransfer[] @relation("TransferOrigin")
  transfersDest   StockTransfer[] @relation("TransferDestination")

  @@index([type])
  @@index([active])
}

// Usuário pode operar em múltiplas lojas (multi-loja real)
model StoreUser {
  id        String   @id @default(uuid())
  store     Store    @relation(fields: [storeId], references: [id])
  storeId   String
  user      User     @relation(fields: [userId], references: [id])
  userId    String
  isDefault Boolean  @default(false)
  createdAt DateTime @default(now())

  @@unique([storeId, userId])
  @@index([userId])
  @@index([storeId])
}

// ------------------------------
// AUTH / RBAC (compacto e suficiente)
// ------------------------------
model User {
  id           String   @id @default(uuid())
  name         String
  email        String   @unique
  passwordHash String
  active       Boolean  @default(true)
  createdAt    DateTime @default(now())

  role   Role   @relation(fields: [roleId], references: [id])
  roleId String

  stores StoreUser[]
  audit  AuditLog[]

  @@index([active])
}
```

```
model Role {
  id    String @id @default(uuid())
  name  String @unique

  users RoleUser[]
  perms RolePermission[]
}

// Relação separada para facilitar auditoria/expansão
model RoleUser {
  id     String @id @default(uuid())
  role   Role   @relation(fields: [roleId], references: [id])
  roleId String
  user   User   @relation(fields: [userId], references: [id])
  userId String

  @@unique([roleId, userId])
  @@index([userId])
}

// Permissões por chave (ex.: "cash.close", "inventory.adjust")
model RolePermission {
  id            String @id @default(uuid())
  role          Role   @relation(fields: [roleId], references: [id])
  roleId        String
  permissionKey String

  @@unique([roleId, permissionKey])
  @@index([permissionKey])
}

model AuditLog {
  id        String   @id @default(uuid())
  user      User?    @relation(fields: [userId], references: [id])
  userId    String?
  store     Store?   @relation(fields: [storeId], references: [id])
  storeId   String?
  action    String
  entity    String
  entityId  String?
  payload   Json?
  createdAt DateTime @default(now())

  @@index([createdAt])
  @@index([storeId])
  @@index([userId])
  @@index([entity, entityId])
}

// ------------------------------
// PRODUTOS / PREÇOS
// ------------------------------
model Category {
  id   String @id @default(uuid())
  name String @unique

  products Product[]
}

model Product {
  id         String    @id @default(uuid())
  ean        String?   @unique
  name       String
  brand      String?
  controlled Boolean   @default(false)
  active     Boolean   @default(true)
  createdAt  DateTime  @default(now())

  category   Category? @relation(fields: [categoryId], references: [id])
  categoryId String?

  prices   ProductPrice[]
```

```
  lots     InventoryLot[]
  saleItems SaleItem[]

  @@index([active])
  @@index([controlled])
  @@index([categoryId])
  @@index([name])
}

model ProductPrice {
  id        String   @id @default(uuid())
  product   Product  @relation(fields: [productId], references: [id])
  productId String

  // Dinheiro: Decimal (evitar Float)
  price     Decimal  @db.Decimal(14, 2)

  // Se no futuro quiser preço por loja, adicione storeId aqui (fase avançada)
  createdAt DateTime @default(now())
  active    Boolean  @default(true)

  @@index([productId, active])
}

// ------------------------------
// ESTOQUE (LOTE/VALIDADE) + MOVIMENTAÇÕES
// ------------------------------
model InventoryLot {
  id         String   @id @default(uuid())
  product    Product  @relation(fields: [productId], references: [id])
  productId  String
  store      Store    @relation(fields: [storeId], references: [id])
  storeId    String

  lotNumber  String
  expiration DateTime
  costUnit   Decimal  @db.Decimal(14, 4) // custo interno com 4 casas
  quantity   Int
  createdAt  DateTime @default(now())
  active     Boolean  @default(true)

  moves      InventoryMovement[]

  @@index([storeId, productId])
  @@index([expiration])
  @@index([active])
  @@unique([storeId, productId, lotNumber, expiration])
}

model InventoryMovement {
  id        String               @id @default(uuid())
  store     Store                @relation(fields: [storeId], references: [id])
  storeId   String
  product   Product              @relation(fields: [productId], references: [id])
  productId String
  lot       InventoryLot?        @relation(fields: [lotId], references: [id])
  lotId     String?

  type      InventoryMovementType
  quantity  Int // sempre positivo; o tipo determina o sentido
  reason    String?
  refType   String?
  refId     String?
  createdAt DateTime             @default(now())

  // Ligações importantes
  sale      Sale?                @relation(fields: [saleId], references: [id])
  saleId    String?

  transfer  StockTransfer?       @relation(fields: [transferId], references: [id])
  transferId String?
  createdBy User?                @relation(fields: [createdById], references: [id])
```

```
  createdById String?

  @@index([storeId, createdAt])
  @@index([productId, createdAt])
  @@index([refType, refId])
  @@index([saleId])
  @@index([transferId])
}

// ------------------------------
// VENDAS / ITENS / PAGAMENTOS (COGS gravado)
// ------------------------------
model Sale {
  id         String      @id @default(uuid())
  number     String
  store      Store       @relation(fields: [storeId], references: [id])
  storeId    String

  customer   Customer?   @relation(fields: [customerId], references: [id])
  customerId String?

  seller     User?       @relation(fields: [sellerId], references: [id])
  sellerId   String?

  status     SaleStatus   @default(DRAFT)
  channel    SaleChannel  @default(BALCAO)

  total      Decimal      @db.Decimal(14, 2)
  discount   Decimal      @db.Decimal(14, 2) @default(0)

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

  items      SaleItem[]
  payments   Payment[]
  movements  InventoryMovement[]
  delivery   Delivery?

  @@index([storeId, createdAt])
  @@index([status])
  @@unique([storeId, number])
}

model SaleItem {
  id        String   @id @default(uuid())
  sale      Sale     @relation(fields: [saleId], references: [id])
  saleId    String

  product   Product  @relation(fields: [productId], references: [id])
  productId String

  // No MVP pode ser 1 lote por item; se precisar multi-lote por item,
  // trocamos para SaleItemAllocation (fase avançada).
  lot       InventoryLot? @relation(fields: [lotId], references: [id])
  lotId     String?

  quantity  Int
  priceUnit Decimal  @db.Decimal(14, 2)
  subtotal  Decimal  @db.Decimal(14, 2)

  // COGS gravado no pagamento (PAID): custo unitário e total do item
  cogsUnit  Decimal? @db.Decimal(14, 4)
  cogsTotal Decimal? @db.Decimal(14, 2)

  @@index([saleId])
  @@index([productId])
  @@index([lotId])
}

model Payment {
  id        String        @id @default(uuid())
  sale      Sale          @relation(fields: [saleId], references: [id])
```

```
  saleId    String
  method    PaymentMethod
  amount    Decimal       @db.Decimal(14, 2)
  createdAt DateTime      @default(now())

  @@index([saleId])
  @@index([createdAt])
}

// ------------------------------
// CAIXA (SESSÃO + MOVIMENTOS)
// ------------------------------
model CashSession {
  id          String    @id @default(uuid())
  store       Store     @relation(fields: [storeId], references: [id])
  storeId     String
  openedBy    User?     @relation(fields: [openedById], references: [id])
  openedById  String?
  openedAt    DateTime  @default(now())
  closedBy    User?     @relation(fields: [closedById], references: [id])
  closedById  String?
  closedAt    DateTime?
  initialCash Decimal   @db.Decimal(14, 2)
  finalCash   Decimal?  @db.Decimal(14, 2)
  note        String?

  movements   CashMovement[]

  @@index([storeId, openedAt])
  @@index([closedAt])
}

model CashMovement {
  id         String           @id @default(uuid())
  session    CashSession      @relation(fields: [sessionId], references: [id])
  sessionId  String
  type       CashMovementType
  method     PaymentMethod?
  amount     Decimal          @db.Decimal(14, 2)
  reason     String?
  refType    String?
  refId      String?
  createdAt  DateTime         @default(now())

  createdBy  User?            @relation(fields: [createdById], references: [id])
  createdById String?

  @@index([sessionId, createdAt])
  @@index([refType, refId])
}

// ------------------------------
// CLIENTES + ENDEREÇOS
// ------------------------------
model Customer {
  id        String   @id @default(uuid())
  name      String
  document  String?
  phone     String?
  email     String?
  createdAt DateTime @default(now())

  addresses Address[]
  sales     Sale[]

  @@index([name])
  @@index([document])
}

model Address {
  id         String   @id @default(uuid())
  customer   Customer @relation(fields: [customerId], references: [id])
```

```
  customerId String
  street     String
  number     String
  district   String
  city       String
  state      String
  zipCode    String
  complement String?
  reference  String?

  @@index([customerId])
}

// ------------------------------
// DELIVERY (gancho; fase 2)
// ------------------------------
model Delivery {
  id          String   @id @default(uuid())
  sale        Sale     @relation(fields: [saleId], references: [id])
  saleId      String   @unique
  store       Store    @relation(fields: [storeId], references: [id])
  storeId     String
  status      String
  fee         Decimal  @db.Decimal(14, 2) @default(0)
  deliveredAt DateTime?
  createdAt   DateTime @default(now())

  @@index([storeId, createdAt])
}

// ------------------------------
// TRANSFERÊNCIAS CENTRAL ↔ LOJA
// ------------------------------
model StockTransfer {
  id               String         @id @default(uuid())
  originStore      Store          @relation("TransferOrigin", fields: [originStoreId], references: [id])
  originStoreId    String
  destinationStore Store          @relation("TransferDestination", fields: [destinationStoreId], referenc
  destinationStoreId String

  status           TransferStatus @default(DRAFT)
  createdAt        DateTime       @default(now())
  createdBy        User?          @relation(fields: [createdById], references: [id])
  createdById      String?
  sentAt           DateTime?
  receivedAt       DateTime?
  receivedBy       User?          @relation(fields: [receivedById], references: [id])
  receivedById     String?
  note             String?

  items            StockTransferItem[]
  movements        InventoryMovement[]

  @@index([originStoreId, createdAt])
  @@index([destinationStoreId, createdAt])
  @@index([status])
}

model StockTransferItem {
  id         String        @id @default(uuid())
  transfer   StockTransfer @relation(fields: [transferId], references: [id])
  transferId String

  product    Product       @relation(fields: [productId], references: [id])
  productId  String

  // lote da origem (quando existir)
  originLot  InventoryLot? @relation(fields: [originLotId], references: [id])
  originLotId String?

  quantity   Int
  costUnit   Decimal       @db.Decimal(14, 4) // custo preservado
```

```
  @@index([transferId])
  @@index([productId])
  @@index([originLotId])
}
```