# 📋 ADDENDUM AL PRD: VERSIÓN 2.0 INTEGRADA

## Motor Transaccional + Maestros + Seguridad + POS

**Proyecto:** Farutech - Gestor Documental Transaccional Multi-tenant
**Versión:** 2.0 (Extensión Crítica)
**Fecha:** Enero 2026
**Audiencia:** Head of Product, Líderes Técnicos (Backend/Frontend/DB), Equipo QA
**Estado:** Aprobado para Desarrollo Paralelo

---

## 📑 TABLA DE CONTENIDOS



---

# 1. INTRODUCCIÓN Y CONTEXTO

### 1.1 Brecha Identificada en PRD V1

El PRD original definió exitosamente el **Core Transaccional** (motor de documentos parametrizables), pero dejó vacíos críticos:

| Aspecto | V1 | V2 (Este Documento) |
|---|---|---|
| **Motor de Documentos** | ✅ Definido | ✅ Integrado |
| **Datos Maestros** | ⚠ Menciona "Items" | ✅ Arquitectura Completa |
| **Seguridad** | ⚠ Roles genéricos | ✅ RBAC Granular tipo ERP |
| **Control de Caja** | ❌ No existe | ✅ POS + Auditoría Completa |
| **Unidades de Medida** | ❌ No contemplado | ✅ Conversiones Factoriales |
| **Bodegas Multi-almacén** | ❌ Mencionado, no detallado | ✅ Inventario Distribuido |
| **Scope de Datos** | ❌ No definido | ✅ Filtros por Usuario/Depto |
| **Documentos Soporte** | ❌ No aplica a retiros | ✅ Generación Automática |

### 1.2 Principios Rectores de V2

1. PARAMETRIZABILIDAD: Máximo sin código (maestros configurables)
2. AISLAMIENTO MULTI-TENANT: Cada tenant vive en su BD segregada
3. GRANULARIDAD DE SEGURIDAD: Control por módulo, tabla, fila y procedimiento
4. TRAZABILIDAD TOTAL: Cada transacción, movimiento de caja y cambio de parámetro es auditable
5. ARQUITECTURA MODULAR: 3 equipos paralelos, débil acoplamiento entre módulos
6. ESCALABILIDAD DESDE PISO: Diseño que soporte 1M transacciones/mes sin degradación

---

## 2. DIAGRAMA ENTIDAD-RELACIÓN (DER) INTEGRADO

### 2.1 Vista Conceptual (Mermaid)

```
erDiagram
TENANT ||--o{ USER : "owns"
TENANT ||--o{ ROLE : "defines"
TENANT ||--o{ PERMISSION : "restricts"
```

```
ROLE }o--|| PERMISSION : "has"
USER }o--|| ROLE : "assigned"

TENANT ||--o{ DOCUMENT_TYPE : "parametrizes"
DOCUMENT_TYPE ||--o{ DOCUMENT_SUBTYPE : "extends"
DOCUMENT_SUBTYPE ||--o{ DOCUMENT_SUBTYPE_CONFIG : "config"

TENANT ||--o{ ITEM : "owns"
TENANT ||--o{ ITEM_CATEGORY : "categorizes"
TENANT ||--o{ UNIT_OF_MEASURE : "defines"
TENANT ||--o{ CONVERSION_FACTOR : "converts"

TENANT ||--o{ CLIENT : "owns"
TENANT ||--o{ SUPPLIER : "owns"

TENANT ||--o{ WAREHOUSE : "owns"
TENANT ||--o{ WAREHOUSE_STOCK : "tracks"
WAREHOUSE ||--o{ WAREHOUSE_STOCK : "contains"
ITEM ||--o{ WAREHOUSE_STOCK : "stored"

TENANT ||--o{ PAYMENT_METHOD : "defines"
TENANT ||--o{ PAYMENT_METHOD_RULE : "restricts"

TENANT ||--o{ TRX_DOCUMENT : "transacts"
DOCUMENT_SUBTYPE ||--o{ TRX_DOCUMENT : "instantiates"
CLIENT ||--o{ TRX_DOCUMENT : "parties"
TRX_DOCUMENT ||--o{ TRX_DOCUMENT_LINE : "contains"
ITEM ||--o{ TRX_DOCUMENT_LINE : "references"

TENANT ||--o{ CASH_BOX : "manages"
TENANT ||--o{ CASHIER_TURN : "controls"
USER ||--o{ CASHIER_TURN : "operates"
CASH_BOX ||--o{ CASHIER_TURN : "assigned"

CASHIER_TURN ||--o{ CASH_MOVEMENT : "generates"
CASH_MOVEMENT ||--o{ TRX_DOCUMENT : "backs"
```

```
TENANT ||--o{ AUDIT_LOG : "traces"
USER ||--o{ AUDIT_LOG : "performs"
TRX_DOCUMENT ||--o{ AUDIT_LOG : "triggers"

WAREHOUSE_STOCK ||--o{ AUDIT_LOG : "tracked"
```

### 2.2 Estructura Dimensional (Analítica)

HECHOS (Fact Tables):
├── fact_transactions (Core - Ingresos/Egresos)
├── fact_cash_movements (Caja)
├── fact_inventory_adjustments (Inventario)
└── fact_audit_trail (Auditoría)

DIMENSIONES (Dimension Tables):
├── dim_customers (Clientes)
├── dim_items (Items/Servicios)
├── dim_users (Usuarios/Cajeros)
├── dim_warehouses (Bodegas)
├── dim_payment_methods (Formas de Pago)
└── dim_dates (Calendario - Opcional pero recomendado)

---

# 3. MÓDULO 1: DATOS MAESTROS Y PARAMETRIZACIÓN

### 3.1 Justificación Arquitectónica

El **Módulo Maestros** es el "diccionario de negocio" que permite que el sistema sea agnóstico del dominio. Sin maestros bien diseñados, cada transacción falla o produce datos inconsistentes.

### 3.2 Arquitectura de Identificadores (UUID vs. Código Visible)

**Decisión Crítica:** ¿Cómo nombrar/referenciar las entidades?

**Estrategia Dual Implementada:**

```
┌─────────────────────────────────────────────────┐
│ TABLA MAESTRO │
├─────────────────────────────────────────────────┤
│ id (UUID) [PK - Inmutable] │
│ ├─> Nunca cambia, es la llave BD │
│ ├─> Usado internamente en relaciones│
│ └─> No se expone en UI (salvo debug)│
│ │
│ code (VARCHAR) [Visible - Mutable] │
│ ├─> Lo que el usuario ve/modifica │
│ ├─> Puede cambiar (ej: SKU de Item) │
│ ├─> Único por tenant │
│ └─> Índice para búsquedas rápidas │
```

```
│ │
│ name (VARCHAR) [Descripción] │
│ ├──> Usado en reportes/impresión │
│ └──> Puede cambiar sin romper integridad│
└──────────────────────────────────────────┘
```

EJEMPLO - ITEM:
id = "550e8400-e29b-41d4-a716-446655440001" (UUID)
code = "SKU-LAPTOP-001" (Visible, mutable)
name = "Laptop Dell XPS 13 - Modelo 2025" (Descripción)

Si el usuario cambia code a "SKU-LAPTOP-002":
✅ Relaciones (FK) siguen siendo válidas (usan id)
✅ Histórico permanece consistente
✅ No hay cascadas de cambios

**Impacto en Tablas:**

```
-- PATRÓN UNIVERSAL EN TODOS LOS MAESTROS:
CREATE TABLE mae__items (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL, -- Multi-tenant
code VARCHAR(50) NOT NULL, -- Mutable, visible
name VARCHAR(255) NOT NULL, -- Descripción
description TEXT,

    -- FK siempre usan id (UUID), NUNCA code
    category_id UUID REFERENCES mae__item_categories(id),
    unit_id UUID REFERENCES mae__units_of_measure(id),

    UNIQUE(tenant_id, code) -- code único por tenant

);

-- En transacciones, se referencia siempre por id:
CREATE TABLE trx__document_lines (
id UUID PRIMARY KEY,
item_id UUID NOT NULL REFERENCES mae__items(id),
-- Cuando el item.code cambia, estas líneas NO se ven afectadas
);
```

### 3.3 Items/Bienes y Servicios

```
-- TABLA: mae__items (Inventariables y Servicios)
CREATE TABLE mae__items (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
```

```
-- IDENTIFICADORES
code VARCHAR(50) NOT NULL, -- SKU mutable
name VARCHAR(255) NOT NULL,
description TEXT,

-- CLASIFICACIÓN
type ENUM('product', 'service', 'package') NOT NULL DEFAULT 'product',
category_id UUID NOT NULL REFERENCES mae__item_categories(id),

-- UNIDADES
unit_id UUID NOT NULL REFERENCES mae__units_of_measure(id),

-- PRECIOS Y COSTOS
cost_price DECIMAL(15, 4) NOT NULL DEFAULT 0, -- Precio de costo
default_sale_price DECIMAL(15, 4) NOT NULL DEFAULT 0, -- Precio por defecto
is_taxable BOOLEAN DEFAULT TRUE, -- ¿Aplica IVA?
tax_rate DECIMAL(5, 2) DEFAULT 19, -- 19% (Colombia 2025)

-- CONTROL
is_active BOOLEAN DEFAULT TRUE,
requires_lot_tracking BOOLEAN DEFAULT FALSE, -- Lotes/series
reorder_point INT DEFAULT 0, -- Stock mínimo
lead_time_days INT DEFAULT 0, -- Días para reabastecimiento

-- METADATA (JSONB)
metadata JSONB, -- {supplier_code, barcode, warranty_months, ...}

-- AUDITORÍA
created_by UUID NOT NULL REFERENCES scr__users(id),
updated_by UUID NOT NULL REFERENCES scr__users(id),
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

UNIQUE(tenant_id, code),
CONSTRAINT valid_prices CHECK (cost_price >= 0 AND default_sale_price >= 0),
CONSTRAINT valid_tax CHECK (tax_rate >= 0 AND tax_rate <= 100)
```

);

-- ÍNDICES CRÍTICOS
CREATE INDEX idx_mae_items_tenant_active ON mae__items(tenant_id, is_active);
CREATE INDEX idx_mae_items_category ON mae__items(category_id);
CREATE INDEX idx_mae_items_code ON mae__items(tenant_id, code);

### 3.4 Categorías de Items

CREATE TABLE mae__item_categories (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

```
code VARCHAR(20) NOT NULL, -- CAT-001, CAT-002
name VARCHAR(100) NOT NULL,
description TEXT,

parent_category_id UUID REFERENCES mae__item_categories(id), -- Jerarquía

-- METADATA
default_tax_rate DECIMAL(5, 2) DEFAULT 19, -- Impuesto por defecto para esta c
metadata JSONB, -- {color, icon, accounting_code, ...}

is_active BOOLEAN DEFAULT TRUE,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

UNIQUE(tenant_id, code)
```

);

### 3.5 Unidades de Medida y Conversiones

-- TABLA: mae__units_of_measure (Unidades base)
CREATE TABLE mae__units_of_measure (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

```
code VARCHAR(10) NOT NULL, -- unidad, kg, lts, hrs, caja
name VARCHAR(50) NOT NULL,
description TEXT,

type ENUM('standard', 'time', 'weight', 'volume', 'custom') DEFAULT 'standard',
```

```
    is_base_unit BOOLEAN DEFAULT FALSE, -- ¿Unidad de referencia?
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(tenant_id, code)
);

-- TABLA: mae__conversion_factors (Factores de conversión)
-- Ej: 1 Caja = 12 Unidades
CREATE TABLE mae__conversion_factors (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    item_id UUID NOT NULL REFERENCES mae__items(id),
    from_unit_id UUID NOT NULL REFERENCES mae__units_of_measure(id),
    to_unit_id UUID NOT NULL REFERENCES mae__units_of_measure(id),

    conversion_factor DECIMAL(15, 4) NOT NULL, -- 1 Caja = 12 unidades (factor = 1
    rounding_method ENUM('floor', 'ceil', 'round') DEFAULT 'round',

    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT valid_factor CHECK (conversion_factor > 0),
    UNIQUE(tenant_id, item_id, from_unit_id, to_unit_id)
);

-- LÓGICA DE NEGOCIO (Backend):
-- Al crear una línea transaccional, si la cantidad viene en "Cajas",
-- el sistema busca el conversion_factor y convierte automáticamente a unidad base.
```

### 3.6 Bodegas y Stock Distribuido

```
-- TABLA: mae__warehouses (Multi-almacén)
CREATE TABLE mae__warehouses (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
```

```
    code VARCHAR(20) NOT NULL, -- BDG-001, BDG-CENTRAL, BDG-BRANCH-A
    name VARCHAR(100) NOT NULL,
    location VARCHAR(255),

    manager_user_id UUID REFERENCES scr__users(id),
    manager_phone VARCHAR(20),
    manager_email VARCHAR(100),

    warehouse_type ENUM('central', 'branch', 'transit', 'archive') DEFAULT 'central',

    is_active BOOLEAN DEFAULT TRUE,
    is_default BOOLEAN DEFAULT FALSE, -- Bodega por defecto para nuevas transa

    -- METADATA
    metadata JSONB, -- {latitude, longitude, capacity, storage_temperature, ...}

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(tenant_id, code)
);

-- TABLA: mae__warehouse_stock (Inventario por almacén)
CREATE TABLE mae__warehouse_stock (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
    warehouse_id UUID NOT NULL REFERENCES mae__warehouses(id) ON DELETE CASCADE,
    item_id UUID NOT NULL REFERENCES mae__items(id) ON DELETE CASCADE,

    -- CANTIDADES
    quantity_on_hand DECIMAL(15, 4) NOT NULL DEFAULT 0, -- Cantidad física
    quantity_reserved DECIMAL(15, 4) NOT NULL DEFAULT 0, -- Pendiente entregar
    quantity_available DECIMAL(15, 4) GENERATED ALWAYS AS
        (quantity_on_hand - quantity_reserved) STORED, -- Disponible

    -- CONTROL
    last_count_at TIMESTAMP, -- Última verificación física
    last_counted_by_user_id UUID REFERENCES scr__users(id),
```

```
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

  UNIQUE(tenant_id, warehouse_id, item_id),
  CONSTRAINT valid_quantities CHECK (
    quantity_on_hand >= 0
    AND quantity_reserved >= 0
    AND quantity_available >= 0
  )
);

-- ÍNDICES CRÍTICOS (Consultas frecuentes)
CREATE INDEX idx_warehouse_stock_available
ON mae__warehouse_stock(tenant_id, warehouse_id, quantity_available)
WHERE quantity_available > 0;
```

### 3.7 Clientes y Proveedores

```
-- TABLA: mae__clients (Terceros: Clientes)
CREATE TABLE mae__clients (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

  -- IDENTIFICADORES
  code VARCHAR(20) NOT NULL, -- CLIE-001, mutable
  name VARCHAR(255) NOT NULL,

  -- INFORMACIÓN FISCAL
  tax_id VARCHAR(50), -- RUT (Colombia), NIT, RFC (México)
  tax_id_type ENUM('NIT', 'CC', 'PEP', 'passport', 'other') DEFAULT 'NIT',

  -- CONTACTO
  email VARCHAR(100) UNIQUE,
  phone VARCHAR(20),
  address TEXT,
  city VARCHAR(100),
  state_province VARCHAR(100),
  country VARCHAR(100) DEFAULT 'CO',
  postal_code VARCHAR(20),
```

```
  -- COMERCIAL
  client_type ENUM('individual', 'company') NOT NULL DEFAULT 'company',
  payment_terms ENUM('cash', '15_days', '30_days', '60_days') DEFAULT 'cash',
  credit_limit DECIMAL(15, 2) DEFAULT 0,
  current_balance DECIMAL(15, 2) DEFAULT 0,

  -- REFERENCIA A LISTA DE PRECIOS
  default_price_list_id UUID REFERENCES mae__price_lists(id),

  -- ESTADO
  is_active BOOLEAN DEFAULT TRUE,
  is_vip BOOLEAN DEFAULT FALSE,

  -- METADATA
  metadata JSONB, -- {billing_contact, parent_company, segment, ...}

  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

  UNIQUE(tenant_id, code),
  UNIQUE(tenant_id, email)
);

-- TABLA: mae__suppliers (Terceros: Proveedores)
CREATE TABLE mae__suppliers (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

  code VARCHAR(20) NOT NULL,
  name VARCHAR(255) NOT NULL,

  tax_id VARCHAR(50),
  tax_id_type ENUM('NIT', 'RUT', 'RFC', 'other') DEFAULT 'NIT',

  email VARCHAR(100),
  phone VARCHAR(20),
  contact_person VARCHAR(100),
```

```
    country VARCHAR(100) DEFAULT 'CO',
    payment_terms_default VARCHAR(50), -- Net 30, Net 60, etc.

    is_active BOOLEAN DEFAULT TRUE,

    metadata JSONB,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(tenant_id, code)
);
```

### 3.8 Listas de Precios

```
-- TABLA: mae__price_lists (Listas de precios)
CREATE TABLE mae__price_lists (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    code VARCHAR(20) NOT NULL, -- PL-STD, PL-VIP, PL-SEASONAL
    name VARCHAR(100) NOT NULL,
    description TEXT,

    -- VIGENCIA
    effective_from DATE NOT NULL,
    effective_to DATE, -- NULL = vigente indefinidamente

    currency ENUM('USD', 'COP', 'MXN', 'ARS', 'BRL') DEFAULT 'COP',

    -- CONTROL
    is_active BOOLEAN DEFAULT TRUE,
    is_default BOOLEAN DEFAULT FALSE,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
UNIQUE(tenant_id, code)
```

);

-- TABLA: mae__price_list_items (Precios específicos por artículo)
CREATE TABLE mae__price_list_items (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
price_list_id UUID NOT NULL REFERENCES mae__price_lists(id) ON DELETE CASCADE,
item_id UUID NOT NULL REFERENCES mae__items(id) ON DELETE CASCADE,

```
-- PRECIO Y DESCUENTOS
sell_price DECIMAL(15, 4) NOT NULL,
discount_percentage DECIMAL(5, 2) DEFAULT 0, -- Descuento sobre este precio
effective_price DECIMAL(15, 4) GENERATED ALWAYS AS
    (sell_price * (1 - discount_percentage / 100)) STORED,

unit ENUM('unit', 'box', 'case', 'bulk') DEFAULT 'unit',

created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

UNIQUE(tenant_id, price_list_id, item_id)
```

);

### 3.9 Formas de Pago

-- TABLA: mae__payment_methods (Formas de pago disponibles)
CREATE TABLE mae__payment_methods (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

```
code VARCHAR(20) NOT NULL, -- CASH, CARD, CHECK, TRANSFER
name VARCHAR(50) NOT NULL,
description TEXT,

-- CONFIGURACIÓN
payment_type ENUM('cash', 'card', 'check', 'transfer', 'credit', 'other') NOT NULL,
requires_reference BOOLEAN DEFAULT FALSE, -- ¿Requiere número de referer
is_cash_equivalent BOOLEAN DEFAULT FALSE, -- Se trata como efectivo
```

```
  -- DISPONIBILIDAD
  is_active BOOLEAN DEFAULT TRUE,
  is_available_for_deposits BOOLEAN DEFAULT TRUE,
  is_available_for_withdrawals BOOLEAN DEFAULT TRUE,

  -- REGLAS DE NEGOCIO
  max_transaction_amount DECIMAL(15, 2), -- NULL = sin límite
  requires_approval_above DECIMAL(15, 2), -- ¿Requiere supervisión?

  metadata JSONB, -- {bank_code, bank_name, processing_fee, ...}

  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

  UNIQUE(tenant_id, code)
```

);

-- TABLA: mae__payment_method_rules (Reglas por documento/rol)
CREATE TABLE mae__payment_method_rules (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

```
  payment_method_id UUID NOT NULL REFERENCES mae__payment_methods(id)
  document_subtype_id UUID REFERENCES cfg__document_subtypes(id), -- NULL
  user_role_id UUID REFERENCES scr__roles(id), -- NULL = aplica a todos

  -- RESTRICCIÓN
  is_allowed BOOLEAN DEFAULT TRUE,
  min_amount DECIMAL(15, 2) DEFAULT 0,
  max_amount DECIMAL(15, 2), -- NULL = sin límite

  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

  UNIQUE(tenant_id, payment_method_id, document_subtype_id, user_role_id)
```

);

-- CONSULTA: ¿Puede este usuario usar esta forma de pago en este documento?
-- SELECT * FROM mae__payment_method_rules
-- WHERE tenant_id = $1
-- AND payment_method_id = $2
-- AND (document_subtype_id = $3 OR document_subtype_id IS NULL)
-- AND (user_role_id = $4 OR user_role_id IS NULL)
-- AND is_allowed = true;

---

# 4. MÓDULO 2: SEGURIDAD AVANZADA (RBAC GRANULAR)

## 4.1 Filosofía de Seguridad

El modelo RBAC debe ser **lo suficientemente granular para cumplir normativas** (auditoría, separación de funciones) pero **lo suficientemente simple para ser administrable**.

NIVELES DE CONTROL (De general a específico):

1. MÓDULO (¿Puede ver el módulo de Caja?)
   └─> Ejemplo: module:cashier:view
2. ENTIDAD/TABLA (¿Puede ver la tabla de Transacciones?)
   └─> Ejemplo: entity:transactions:read
3. OPERACIÓN CRUD (¿Puede crear documentos?)
   └─> Ejemplo: action:document:create
4. PROCEDIMIENTO (¿Puede cerrar caja?)
   └─> Ejemplo: procedure:close_cashier_session:execute
5. SCOPE DE DATOS (¿Qué datos ve?)
   └─> Ejemplo: scope:own_warehouse, scope:own_documents, scope:all_data
6. RECURSO ESPECÍFICO (¿Puede editar ESTE documento?)
   └─> Esto se valida en tiempo de ejecución (Backend)

## 4.2 Tablas de RBAC

-- TABLA: scr__roles (Roles)
CREATE TABLE scr__roles (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

```
code VARCHAR(50) NOT NULL, -- ADMIN, CASHIER, WAREHOUSE_MGR, SUPER
name VARCHAR(100) NOT NULL,
description TEXT,

-- CLASIFICACIÓN
role_type ENUM('system', 'custom') DEFAULT 'custom',
is_system_protected BOOLEAN DEFAULT FALSE, -- No se puede eliminar si es si
```

```
  -- METADATA
  metadata JSONB, -- {department, salary_band, ...}

  is_active BOOLEAN DEFAULT TRUE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

  UNIQUE(tenant_id, code)
);

-- TABLA: scr__permissions (Permisos granulares)
CREATE TABLE scr__permissions (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

  code VARCHAR(100) NOT NULL, -- module:cashier:view, entity:transactions:upd
  name VARCHAR(150) NOT NULL,
  description TEXT,

  -- CLASIFICACIÓN (Facilita búsqueda y UI)
  permission_type ENUM('module', 'entity', 'action', 'procedure', 'scope', 'report') D
  category VARCHAR(50), -- cashier, security, inventory, reporting

  -- RESTRICCIÓN
  is_active BOOLEAN DEFAULT TRUE,
  requires_approval BOOLEAN DEFAULT FALSE, -- ¿Requiere supervisión?

  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

  UNIQUE(tenant_id, code)
);

-- TABLA: scr__role_has_permissions (Asignación: Rol -> Permisos)
CREATE TABLE scr__role_has_permissions (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
role_id UUID NOT NULL REFERENCES scr__roles(id) ON DELETE CASCADE,
permission_id UUID NOT NULL REFERENCES scr__permissions(id) ON DELETE CASCADE,
```

```
    -- CONTEXTO
    granted_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    granted_by_user_id UUID NOT NULL REFERENCES scr__users(id),

    UNIQUE(role_id, permission_id),
    INDEX idx_role_perms (role_id)
);

-- TABLA: scr__user_has_roles (Asignación: Usuario -> Roles)
CREATE TABLE scr__user_has_roles (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
user_id UUID NOT NULL REFERENCES scr__users(id) ON DELETE CASCADE,
role_id UUID NOT NULL REFERENCES scr__roles(id) ON DELETE CASCADE,

    -- CONTEXTO
    assigned_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    assigned_by_user_id UUID NOT NULL REFERENCES scr__users(id),
    expires_at TIMESTAMP, -- Rol temporal (expira en fecha)

    is_active BOOLEAN DEFAULT TRUE,

    UNIQUE(user_id, role_id),
    INDEX idx_user_roles (user_id)
);

-- TABLA: scr__user_has_permissions (Direct: Usuario -> Permisos, sin rol intermediario)
CREATE TABLE scr__user_has_permissions (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
user_id UUID NOT NULL REFERENCES scr__users(id) ON DELETE CASCADE,
permission_id UUID NOT NULL REFERENCES scr__permissions(id) ON DELETE CASCADE,

    granted_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    granted_by_user_id UUID NOT NULL REFERENCES scr__users(id),

    UNIQUE(user_id, permission_id)
);
```

### 4.3 Scope de Datos (Data Filtering)

-- TABLA: scr__data_scopes (Define los filtros que ve cada usuario)
CREATE TABLE scr__data_scopes (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

```
-- DEFINICIÓN DEL ALCANCE
scope_name VARCHAR(100) NOT NULL, -- own_warehouse, own_department, a
description TEXT,

-- TABLA Y FILTRO SQL
target_table VARCHAR(100) NOT NULL, -- mae__clients, trx__documents, mae__v
filter_column VARCHAR(100) NOT NULL, -- warehouse_id, created_by, departme

-- VALORES PERMITIDOS (JSONB array)
filter_values JSONB NOT NULL, -- ["warehouse_id_1", "warehouse_id_2"] o ["*"]

is_active BOOLEAN DEFAULT TRUE,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

UNIQUE(tenant_id, scope_name, target_table)
```

);

-- TABLA: scr__user_data_scopes (Asignación: Usuario -> Scopes)
CREATE TABLE scr__user_data_scopes (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
user_id UUID NOT NULL REFERENCES scr__users(id) ON DELETE CASCADE,
scope_id UUID NOT NULL REFERENCES scr__data_scopes(id) ON DELETE CASCADE,

```
assigned_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

UNIQUE(user_id, scope_id)
```

);

-- EJEMPLOS DE SCOPES:
-- 1. Cajero solo ve transacciones de SU caja:
-- scope_name = "own_cashier_session"
-- target_table = "trx__documents"
-- filter_column = "cashier_session_id"
-- filter_values = ["<session_id_actual>"]

-- 2. Gerente de bodega solo ve stock de SU bodega:
-- scope_name = "own_warehouse"
-- target_table = "mae__warehouse_stock"
-- filter_column = "warehouse_id"
-- filter_values = ["<warehouse_id>"]

-- 3. Supervisor ve TODO:
-- scope_name = "all_data"
-- target_table = ""
-- *filter_column = null*
-- *filter_values = [""]*

### 4.4 Auditoría de Cambios de Permisos (Compliance)

-- TABLA: audit__permission_changes (Histórico de cambios en seguridad)
CREATE TABLE audit__permission_changes (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

```
-- QUÉ CAMBIÓ
entity_type ENUM('role', 'permission', 'user_role', 'user_permission', 'data_scope
entity_id UUID NOT NULL,
operation ENUM('create', 'update', 'delete', 'assign', 'revoke') NOT NULL,

-- QUIÉN LO CAMBIÓ
performed_by_user_id UUID NOT NULL REFERENCES scr__users(id),

-- ANTES/DESPUÉS (JSONB)
old_values JSONB,
new_values JSONB,

-- AUDITORÍA
change_reason TEXT,
ip_address INET,
user_agent TEXT,

created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

INDEX idx_entity (entity_type, entity_id),
INDEX idx_performed_by (performed_by_user_id, created_at)
```

);

### 4.5 Matriz de Permisos Tipo ERP (Ejemplo Inicial)

```
┌──────────────────────────────────────────────────────────────────────────────┐
│ MATRIZ DE PERMISOS INICIAL │
└──────────────────────────────────────────────────────────────────────────────┘

MÓDULO: SEGURIDAD (module:security)
├── module:security:view [Acceso al módulo]
├── entity:users:create [Crear usuarios]
├── entity:users:read [Ver usuarios]
├── entity:users:update [Editar usuarios]
├── entity:users:delete [Eliminar usuarios]
├── entity:roles:create [Crear roles]
├── entity:roles:read [Ver roles]
├── entity:roles:update [Editar roles]
├── entity:roles:delete [Eliminar roles]
├── entity:permissions:read [Ver permisos]
├── procedure:unlock_user:execute [Desbloquear usuario tras fallos login]
└── scope:all_users [Ver todos vs propios]

MÓDULO: MAESTROS (module:masters)
├── module:masters:view
├── entity:items:create
├── entity:items:read
├── entity:items:update
├── entity:items:delete
├── entity:categories:read/create/update/delete
├── entity:clients:read/create/update/delete
├── entity:suppliers:read/create/update/delete
├── entity:warehouses:read/create/update/delete
├── entity:price_lists:read/create/update/delete
└── scope:own_warehouse [vs. all_warehouses]

MÓDULO: TRANSACCIONES (module:transactions)
├── module:transactions:view
├── entity:documents:create
├── entity:documents:read
├── entity:documents:update [¿Permitir cambios post-finalización?]
├── entity:documents:delete [Anular documentos]
├── entity:documents:print
├── entity:documents:export
├── procedure:finalize_document:execute [Marcar como "Finalizado"]
├── procedure:cancel_document:execute [Anular]
├── action:document:approve [Requiere supervisión]
└── scope:own_documents [vs. department vs. all_documents]

MÓDULO: CAJA (module:cashier)
├── module:cashier:view
├── procedure:open_cashier_session:execute [Abrir caja]
├── procedure:close_cashier_session:execute [Cerrar caja]
```

├── procedure:blind_count:execute [Arqueo ciego]
├── procedure:cash_withdrawal:execute [Retiro de efectivo]
├── entity:cash_movements:read
├── entity:cash_movements:create
└── scope:own_cashier_session [vs. all_sessions]

MÓDULO: REPORTES (module:reporting)
├── module:reporting:view
├── report:sales_summary:execute
├── report:inventory_status:execute
├── report:cash_reconciliation:execute
├── report:audit_trail:execute [Solo auditores]
└── scope:own_data [vs. department vs. all]

MÓDULO: AUDITORÍA (module:audit) [Solo para auditores internos]
├── module:audit:view
├── entity:audit_logs:read
├── report:compliance_report:execute
└── procedure:export_audit:execute

---

# 5. MÓDULO 3: GESTIÓN DE EFECTIVO Y POS

## 5.1 Justificación Crítica

El control de caja es **el punto más crítico de fraude financiero**. Sin un modelo robusto:

- Dinero desaparece sin trazabilidad
- Cajeros pueden falsear montos
- Supervisores no tienen visibilidad
- Auditoría interna no puede reconciliar

## 5.2 Arquitectura Conceptual del Ciclo de Caja

┌─────────────────────────────────────────────────────────────────────────┐
│ CICLO DIARIO DE CAJA │
├─────────────────────────────────────────────────────────────────────────┤

1. APERTURA (08:00 AM)
   └─> Cajero + Supervisor abren sesión
   └─> Dinero de base inicial (ej: COP $500,000)
   └─> Se registra en scr__cashier_sessions
2. OPERACIÓN (08:00 AM - 06:00 PM)
   └─> Cada venta genera un pago (trx__payments)
   └─> Cash movements se agregan automáticamente
   └─> Retiros/Sangrías manuales (cash_withdrawals)
3. CIERRE (06:00 PM)
   └─> Cajero realiza ARQUEO CIEGO
   └─> Cuenta dinero físico vs. sistema
   └─> Discrepancias se registran
   └─> Sistema calcula "Sobrante/Faltante"

4. SUPERVISIÓN
   └─> Supervisor revisa arqueo
   └─> Aprueba o rechaza
5. DEPOSITACIÓN
   └─> Dinero se traslada a bóveda/banco
   └─> Se genera documento de respaldo (DELIVERY)

└─> Nueva sesión comienza al día siguiente

### 5.3 Tablas de POS y Caja

-- TABLA: mae__cash_boxes (Cajas físicas)
CREATE TABLE mae__cash_boxes (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

```
code VARCHAR(20) NOT NULL, -- CAJA-1, CAJA-MOSTRADOR-1
name VARCHAR(100) NOT NULL,
location VARCHAR(255), -- Ubicación física

warehouse_id UUID REFERENCES mae__warehouses(id), -- Bodega donde está l

box_type ENUM('cashier', 'deposit', 'return') DEFAULT 'cashier',

is_active BOOLEAN DEFAULT TRUE,
is_default BOOLEAN DEFAULT FALSE,

metadata JSONB, -- {serial_number, terminal_id, ...}

created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

UNIQUE(tenant_id, code)
```

);

-- TABLA: scr__cashier_sessions (Sesiones de caja - El registro principal)
CREATE TABLE scr__cashier_sessions (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

```
-- IDENTIFICACIÓN
session_number VARCHAR(20), -- AUTO-GENERADO: CAJA-20250125-001
```

```
cash_box_id UUID NOT NULL REFERENCES mae__cash_boxes(id),

-- USUARIOS
opened_by_user_id UUID NOT NULL REFERENCES scr__users(id), -- Cajero
supervisor_user_id UUID REFERENCES scr__users(id), -- Supervisor (co-firma)
closed_by_user_id UUID REFERENCES scr__users(id),

-- TIMING
opened_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
closed_at TIMESTAMP,
duration_minutes INT GENERATED ALWAYS AS
    (EXTRACT(EPOCH FROM (closed_at - opened_at)) / 60)::INT STORED,

-- DINERO BASE
opening_balance DECIMAL(15, 2) NOT NULL DEFAULT 0, -- Dinero inicial

-- RESUMEN DE TRANSACCIONES
total_sales DECIMAL(15, 2) DEFAULT 0, -- Ventas totales
total_cash_received DECIMAL(15, 2) DEFAULT 0, -- Efectivo recibido
total_withdrawals DECIMAL(15, 2) DEFAULT 0, -- Retiros/sangrías
total_deposits DECIMAL(15, 2) DEFAULT 0, -- Depósitos manuales

-- CIERRE TEÓRICO (lo que el sistema cree que hay)
expected_balance DECIMAL(15, 2) GENERATED ALWAYS AS
    (opening_balance + total_cash_received - total_withdrawals + total_deposits) S

-- ARQUEO (lo que realmente hay)
physical_count DECIMAL(15, 2), -- Lo que se cuenta manualmente

-- DIFERENCIA
variance DECIMAL(15, 2) GENERATED ALWAYS AS
    (physical_count - expected_balance) STORED, -- +/- Sobrante/Faltante

variance_percentage DECIMAL(5, 2) GENERATED ALWAYS AS
    (CASE WHEN expected_balance != 0
     THEN (variance / expected_balance) * 100
     ELSE 0 END) STORED,
```

```
    -- ESTADO
    status ENUM('open', 'closing', 'closed', 'locked', 'deposited') DEFAULT 'open',
    closure_notes TEXT,

    -- SUPERVISIÓN
    supervisor_approval_at TIMESTAMP,
    supervisor_approval_notes TEXT,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(tenant_id, session_number),
    INDEX idx_session_status (tenant_id, status, closed_at)
);

-- TABLA: caj__cash_movements (Movimientos detallados dentro de sesión)
CREATE TABLE caj__cash_movements (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    session_id UUID NOT NULL REFERENCES scr__cashier_sessions(id),

    -- TIPO DE MOVIMIENTO
    movement_type ENUM('payment', 'withdrawal', 'deposit', 'manual_adjustment')

    -- DOCUMENTO O TRANSACCIÓN RELACIONADA
    document_id UUID REFERENCES trx__documents(id) ON DELETE SET NULL,
    payment_id UUID REFERENCES trx__payments(id) ON DELETE SET NULL,

    -- MONTO Y DESCRIPCIÓN
    amount DECIMAL(15, 2) NOT NULL,
    description VARCHAR(255),

    -- AUDITORÍA
    created_by_user_id UUID NOT NULL REFERENCES scr__users(id),
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,

    -- METADATA
```

```
    reference VARCHAR(100), -- Número de recibo, referencia de transferencia
    metadata JSONB, -- {payment_method_code, card_last_4, ...}

    INDEX idx_session_movements (session_id, created_at),
    INDEX idx_amount (amount)
);

-- TABLA: caj__cash_withdrawals (Retiros de efectivo - Documentos soporte)
CREATE TABLE caj__cash_withdrawals (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    session_id UUID NOT NULL REFERENCES scr__cashier_sessions(id),

    -- DOCUMENTO GENERADO AUTOMÁTICAMENTE
    withdrawal_document_id UUID NOT NULL REFERENCES trx__documents(id),

    -- TIPO DE RETIRO
    withdrawal_type ENUM('cleaning', 'bank_deposit', 'manager_request', 'emergen

    -- DATOS
    amount DECIMAL(15, 2) NOT NULL,
    authorized_by_user_id UUID REFERENCES scr__users(id), -- Supervisor

    -- DESTINO
    destination_description VARCHAR(255), -- "A bóveda", "Al banco", "Fondo de op
    destination_warehouse_id UUID REFERENCES mae__warehouses(id),

    status ENUM('pending', 'approved', 'executed', 'cancelled') DEFAULT 'pending',
    executed_at TIMESTAMP,
    executed_by_user_id UUID REFERENCES scr__users(id),

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    INDEX idx_session_withdrawals (session_id, status)
);
```

### 5.4 Integración: Pagos → Movimientos de Caja

FLUJO AUTOMÁTICO:

1. Usuario crea TRX_DOCUMENT (Factura, Recibo)
   ↓
2. Usuario agrega PAGO con método "CASH"
   ↓
3. Backend:
   ├─> Crea registro en TRX__PAYMENTS
   ├─> Busca sesión de caja ABIERTA para este cajero
   ├─> Crea automáticamente registro en CAJ__CASH_MOVEMENTS
   │ └─> movement_type = "payment"
   │ └─> amount = pago.monto
   │ └─> session_id = sesión abierta
   │ └─> document_id = factura referenciada
   └─> Actualiza TRX__DOCUMENT.status = "PAID"
4. En el Dashboard de Caja, el "Dinero Acumulado" aumenta en tiempo real

VALIDACIÓN DE REGLAS DE NEGOCIO:
├─> ¿Existe sesión ABIERTA para este cajero?
├─> ¿Es la forma de pago "CASH" permitida en este documento?
├─> ¿El monto no excede límites?
└─> ¿El usuario tiene permiso "procedure:cash_payment:execute"?

### 5.5 Arqueo Ciego (Blind Count)

-- TABLA: caj__blind_counts (Registros de arqueos ciegos)
CREATE TABLE caj__blind_counts (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

```
session_id UUID NOT NULL REFERENCES scr__cashier_sessions(id),

-- PARTICIPANTES
performed_by_user_id UUID NOT NULL REFERENCES scr__users(id), -- Cajero
verified_by_user_id UUID REFERENCES scr__users(id), -- Supervisor (testigo)

-- TIMING
started_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
finished_at TIMESTAMP,

-- CONTEO
physical_count DECIMAL(15, 2) NOT NULL, -- Lo que se contó

-- DISCREPANCIA
```

```
    discrepancy_amount DECIMAL(15, 2),
    discrepancy_percentage DECIMAL(5, 2),
    discrepancy_reason_code VARCHAR(50), -- SHORTAGE, OVERAGE, UNACCOUNT

    notes TEXT,

    status ENUM('pending_approval', 'approved', 'disputed', 'resolved') DEFAULT 'pe

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    INDEX idx_session_counts (session_id, status)
);

-- LÓGICA:
-- 1. Cajero inicia arqueo ciego (no ve el saldo del sistema)
-- 2. Cuenta dinero manualmente
-- 3. Ingresa cantidad física
-- 4. Sistema calcula discrepancia
-- 5. Si discrepancia > umbral (ej: 2%), requiere supervisión
-- 6. Supervisor aprueba o genera reporte de faltante
```

### 5.6 Reporte de Cuadre de Caja (Caja Cerrada)

ESTRUCTURA DEL REPORTE (Generado al cerrar sesión):

```
┌──────────────────────────────────────────────────────────────────────┐
│ CUADRE DE CAJA - SESIÓN 20250125-001 │
├──────────────────────────────────────────────────────────────────────┤
```

DATOS DE LA SESIÓN:
Caja: CAJA-1
Cajero: Juan Pérez
Supervisor: María García
Apertura: 2025-01-25 08:15:30
Cierre: 2025-01-25 18:45:15
Duración: 10h 30m

BALANCE TEÓRICO:
Dinero Base: $500,000.00
(+) Pagos Recibidos: $2,350,000.00
(-) Retiros: -$150,000.00
(+) Depósitos: $0.00
———————————————————————————————

Saldo Esperado: $2,700,000.00

CONTEO FÍSICO (ARQUEO):
Dinero Contado: $2,701,250.00

DISCREPANCIA:
Sobrante/Faltante: +$1,250.00
Porcentaje: +0.046%

ESTADO:
✅ Aprobado por Supervisor
Status: CLOSED

DOCUMENTOS DE RESPALDO:

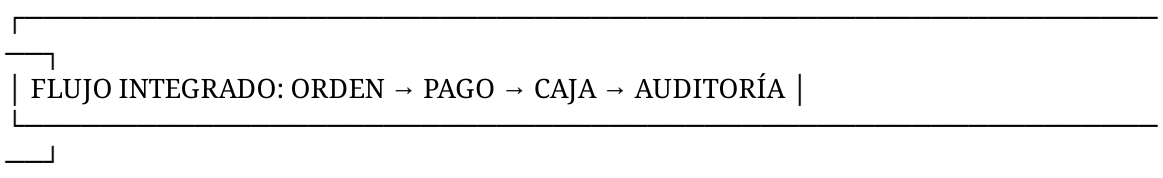
- 47 Recibos de Venta
- 1 Retiro a Bóveda ($150,000)
- 1 Ajuste Manual (+$1,250) [Motivo: Reembolso Cliente]

FIRMA DIGITAL:
Cajero: [FIRMADO]
Supervisor: [FIRMADO]

---

# 6. MATRIZ DE INTEGRACIONES

Esta sección muestra cómo los **4 módulos principales** se integran en un ecosistema cohesivo:




PASO 1: MAESTROS PROPORCIONAN CONTEXTO
mae__items ──────┐
mae__clients ──────┼──> Se validan antes de crear documento
mae__warehouses ──┘

PASO 2: SEGURIDAD VALIDA PERMISOS
scr__user_has_roles
scr__role_has_permissions ──> ¿Puede este usuario crear este tipo de documento?
scr__data_scopes ──────────────> ¿Puede ver estos clientes/almacenes?

PASO 3: DOCUMENTO SE CREA CON VALIDACIONES
cfg__document_subtypes ──> Determina campos obligatorios, formatos
trx__documents ──────────────> Se crea documento con estado "DRAFT"
trx__document_lines ──────> Se agregan líneas (items, cantidades, precios)

PASO 4: PAGO/COBRO
mae__payment_methods ──────> ¿Qué formas de pago se permiten?
trx__payments ──────────────> Se registra el pago

PASO 5: IMPACTO EN CAJA (Si es CASH)
scr__cashier_sessions ──> Sesión activa del cajero
caj__cash_movements ──────> Se agrega movimiento automático
mae__warehouse_stock ──> Si es retiro de bodega, se reduce stock

PASO 6: AUDITABILIDAD COMPLETA
audit__permission_changes ──> Quién creó el documento
audit__transaction_log ──────> Todos los cambios al documento
caj__blind_counts ──────────────> Si hay discrepancia, queda registrada

RESULTADO:
✓ Documento completo y trazable
✓ Dinero contabilizado en caja
✓ Auditores pueden verificar cualquier transacción
✓ Compensaciones automáticas en inventario

---

# 7. DEFINICIONES DE TABLAS SQL (PostgreSQL COMPLETO)

## 7.1 Script Unificado (Orden de Creación)

-- ============================================
-- 1. TABLAS BASE (Multitenancy + Usuarios)
-- ============================================

CREATE TABLE IF NOT EXISTS tenants (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
code VARCHAR(20) NOT NULL UNIQUE,
name VARCHAR(255) NOT NULL,
database_name VARCHAR(100),
is_active BOOLEAN DEFAULT TRUE,
metadata JSONB,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE IF NOT EXISTS scr__users (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

```
email VARCHAR(100) NOT NULL UNIQUE,
username VARCHAR(50) UNIQUE,
password_hash VARCHAR(255) NOT NULL,
name VARCHAR(255) NOT NULL,

is_active BOOLEAN DEFAULT TRUE,
last_login_at TIMESTAMP,

-- PROFILING
avatar_url TEXT,
phone VARCHAR(20),
```

```
    department VARCHAR(100),

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(tenant_id, email),
    INDEX idx_user_active (tenant_id, is_active)
);

-- ============================================
-- 2. SEGURIDAD (RBAC + Data Scopes)
-- ============================================

CREATE TABLE IF NOT EXISTS scr__roles (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    code VARCHAR(50) NOT NULL,
    name VARCHAR(100) NOT NULL,
    description TEXT,

    role_type ENUM('system', 'custom') DEFAULT 'custom',
    is_system_protected BOOLEAN DEFAULT FALSE,
    is_active BOOLEAN DEFAULT TRUE,

    metadata JSONB,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(tenant_id, code)
);

CREATE TABLE IF NOT EXISTS scr__permissions (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    code VARCHAR(100) NOT NULL,
    name VARCHAR(150) NOT NULL,
```

```
  description TEXT,

  permission_type ENUM('module', 'entity', 'action', 'procedure', 'scope', 'report') D
  category VARCHAR(50),

  is_active BOOLEAN DEFAULT TRUE,
  requires_approval BOOLEAN DEFAULT FALSE,

  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

  UNIQUE(tenant_id, code)
```

);

```
CREATE TABLE IF NOT EXISTS scr__role_has_permissions (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
role_id UUID NOT NULL REFERENCES scr__roles(id) ON DELETE CASCADE,
permission_id UUID NOT NULL REFERENCES scr__permissions(id) ON DELETE CASCADE,
```

```
  granted_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  granted_by_user_id UUID REFERENCES scr__users(id),

  UNIQUE(role_id, permission_id)
```

);

```
CREATE TABLE IF NOT EXISTS scr__user_has_roles (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
user_id UUID NOT NULL REFERENCES scr__users(id) ON DELETE CASCADE,
role_id UUID NOT NULL REFERENCES scr__roles(id) ON DELETE CASCADE,
```

```
  assigned_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  assigned_by_user_id UUID REFERENCES scr__users(id),
  expires_at TIMESTAMP,

  is_active BOOLEAN DEFAULT TRUE,

  UNIQUE(user_id, role_id),
  INDEX idx_user_roles (user_id)
```

);

CREATE TABLE IF NOT EXISTS scr__user_has_permissions (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
user_id UUID NOT NULL REFERENCES scr__users(id) ON DELETE CASCADE,
permission_id UUID NOT NULL REFERENCES scr__permissions(id) ON DELETE CASCADE,

```
granted_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
granted_by_user_id UUID REFERENCES scr__users(id),

UNIQUE(user_id, permission_id)
```

);

CREATE TABLE IF NOT EXISTS scr__data_scopes (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

```
scope_name VARCHAR(100) NOT NULL,
description TEXT,

target_table VARCHAR(100) NOT NULL,
filter_column VARCHAR(100) NOT NULL,
filter_values JSONB NOT NULL,

is_active BOOLEAN DEFAULT TRUE,

created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

UNIQUE(tenant_id, scope_name, target_table)
```

);

CREATE TABLE IF NOT EXISTS scr__user_data_scopes (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
user_id UUID NOT NULL REFERENCES scr__users(id) ON DELETE CASCADE,
scope_id UUID NOT NULL REFERENCES scr__data_scopes(id) ON DELETE CASCADE,

```
assigned_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

UNIQUE(user_id, scope_id)
```

);

```
-- ============================================
-- 3. MAESTROS: ITEMS
-- ============================================

CREATE TABLE IF NOT EXISTS mae__item_categories (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    code VARCHAR(20) NOT NULL,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    parent_category_id UUID REFERENCES mae__item_categories(id),

    default_tax_rate DECIMAL(5, 2) DEFAULT 19,
    metadata JSONB,

    is_active BOOLEAN DEFAULT TRUE,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(tenant_id, code)

);

CREATE TABLE IF NOT EXISTS mae__units_of_measure (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    code VARCHAR(10) NOT NULL,
    name VARCHAR(50) NOT NULL,
    description TEXT,

    type ENUM('standard', 'time', 'weight', 'volume', 'custom') DEFAULT 'standard',

    is_base_unit BOOLEAN DEFAULT FALSE,
    is_active BOOLEAN DEFAULT TRUE,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
    UNIQUE(tenant_id, code)
);

CREATE TABLE IF NOT EXISTS mae__items (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    code VARCHAR(50) NOT NULL,
    name VARCHAR(255) NOT NULL,
    description TEXT,

    type ENUM('product', 'service', 'package') NOT NULL DEFAULT 'product',
    category_id UUID NOT NULL REFERENCES mae__item_categories(id),
    unit_id UUID NOT NULL REFERENCES mae__units_of_measure(id),

    cost_price DECIMAL(15, 4) NOT NULL DEFAULT 0,
    default_sale_price DECIMAL(15, 4) NOT NULL DEFAULT 0,

    is_taxable BOOLEAN DEFAULT TRUE,
    tax_rate DECIMAL(5, 2) DEFAULT 19,

    requires_lot_tracking BOOLEAN DEFAULT FALSE,
    reorder_point INT DEFAULT 0,
    lead_time_days INT DEFAULT 0,

    metadata JSONB,

    is_active BOOLEAN DEFAULT TRUE,

    created_by UUID NOT NULL REFERENCES scr__users(id),
    updated_by UUID NOT NULL REFERENCES scr__users(id),

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(tenant_id, code),
    CONSTRAINT valid_prices CHECK (cost_price >= 0 AND default_sale_price >= 0),
```

```
    INDEX idx_item_category (category_id),
    INDEX idx_item_active (tenant_id, is_active)
);

CREATE TABLE IF NOT EXISTS mae__conversion_factors (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    item_id UUID NOT NULL REFERENCES mae__items(id),
    from_unit_id UUID NOT NULL REFERENCES mae__units_of_measure(id),
    to_unit_id UUID NOT NULL REFERENCES mae__units_of_measure(id),

    conversion_factor DECIMAL(15, 4) NOT NULL,
    rounding_method ENUM('floor', 'ceil', 'round') DEFAULT 'round',

    is_active BOOLEAN DEFAULT TRUE,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT valid_factor CHECK (conversion_factor > 0),
    UNIQUE(tenant_id, item_id, from_unit_id, to_unit_id)
);

-- ============================================
-- 4. MAESTROS: TERCEROS
-- ============================================

CREATE TABLE IF NOT EXISTS mae__clients (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    code VARCHAR(20) NOT NULL,
    name VARCHAR(255) NOT NULL,

    tax_id VARCHAR(50),
    tax_id_type ENUM('NIT', 'CC', 'PEP', 'passport', 'other') DEFAULT 'NIT',

    email VARCHAR(100),
    phone VARCHAR(20),
```

```
  address TEXT,
  city VARCHAR(100),
  state_province VARCHAR(100),
  country VARCHAR(100) DEFAULT 'CO',
  postal_code VARCHAR(20),

  client_type ENUM('individual', 'company') NOT NULL DEFAULT 'company',
  payment_terms ENUM('cash', '15_days', '30_days', '60_days') DEFAULT 'cash',

  credit_limit DECIMAL(15, 2) DEFAULT 0,
  current_balance DECIMAL(15, 2) DEFAULT 0,

  default_price_list_id UUID REFERENCES mae__price_lists(id),

  is_active BOOLEAN DEFAULT TRUE,
  is_vip BOOLEAN DEFAULT FALSE,

  metadata JSONB,

  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

  UNIQUE(tenant_id, code),
  INDEX idx_client_active (tenant_id, is_active),
  INDEX idx_client_email (email)
);

CREATE TABLE IF NOT EXISTS mae__suppliers (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

  code VARCHAR(20) NOT NULL,
  name VARCHAR(255) NOT NULL,

  tax_id VARCHAR(50),
  tax_id_type ENUM('NIT', 'RUT', 'RFC', 'other') DEFAULT 'NIT',

  email VARCHAR(100),
```

```
    phone VARCHAR(20),
    contact_person VARCHAR(100),

    country VARCHAR(100) DEFAULT 'CO',
    payment_terms_default VARCHAR(50),

    is_active BOOLEAN DEFAULT TRUE,

    metadata JSONB,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(tenant_id, code)
);

-- ============================================
-- 5. MAESTROS: LOGÍSTICA
-- ============================================

CREATE TABLE IF NOT EXISTS mae__warehouses (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    code VARCHAR(20) NOT NULL,
    name VARCHAR(100) NOT NULL,
    location VARCHAR(255),

    manager_user_id UUID REFERENCES scr__users(id),
    manager_phone VARCHAR(20),
    manager_email VARCHAR(100),

    warehouse_type ENUM('central', 'branch', 'transit', 'archive') DEFAULT 'central',

    is_active BOOLEAN DEFAULT TRUE,
    is_default BOOLEAN DEFAULT FALSE,

    metadata JSONB,
```

```
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

  UNIQUE(tenant_id, code)
);

CREATE TABLE IF NOT EXISTS mae__warehouse_stock (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
warehouse_id UUID NOT NULL REFERENCES mae__warehouses(id) ON DELETE CASCADE,
item_id UUID NOT NULL REFERENCES mae__items(id) ON DELETE CASCADE,

  quantity_on_hand DECIMAL(15, 4) NOT NULL DEFAULT 0,
  quantity_reserved DECIMAL(15, 4) NOT NULL DEFAULT 0,
  quantity_available DECIMAL(15, 4) GENERATED ALWAYS AS
    (quantity_on_hand - quantity_reserved) STORED,

  last_count_at TIMESTAMP,
  last_counted_by_user_id UUID REFERENCES scr__users(id),

  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

  UNIQUE(tenant_id, warehouse_id, item_id),
  CONSTRAINT valid_quantities CHECK (
    quantity_on_hand >= 0
    AND quantity_reserved >= 0
    AND quantity_available >= 0
  ),
  INDEX idx_stock_available (tenant_id, warehouse_id, quantity_available)
);

-- ============================================
-- 6. MAESTROS: PRECIOS Y FORMAS DE PAGO
-- ============================================

CREATE TABLE IF NOT EXISTS mae__price_lists (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
```

```
  code VARCHAR(20) NOT NULL,
  name VARCHAR(100) NOT NULL,
  description TEXT,

  effective_from DATE NOT NULL,
  effective_to DATE,

  currency ENUM('USD', 'COP', 'MXN', 'ARS', 'BRL') DEFAULT 'COP',

  is_active BOOLEAN DEFAULT TRUE,
  is_default BOOLEAN DEFAULT FALSE,

  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

  UNIQUE(tenant_id, code)
);

CREATE TABLE IF NOT EXISTS mae__price_list_items (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
price_list_id UUID NOT NULL REFERENCES mae__price_lists(id) ON DELETE CASCADE,
item_id UUID NOT NULL REFERENCES mae__items(id) ON DELETE CASCADE,

  sell_price DECIMAL(15, 4) NOT NULL,
  discount_percentage DECIMAL(5, 2) DEFAULT 0,
  effective_price DECIMAL(15, 4) GENERATED ALWAYS AS
    (sell_price * (1 - discount_percentage / 100)) STORED,

  unit ENUM('unit', 'box', 'case', 'bulk') DEFAULT 'unit',

  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

  UNIQUE(tenant_id, price_list_id, item_id)
);
```

```
CREATE TABLE IF NOT EXISTS mae__payment_methods (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    code VARCHAR(20) NOT NULL,
    name VARCHAR(50) NOT NULL,
    description TEXT,

    payment_type ENUM('cash', 'card', 'check', 'transfer', 'credit', 'other') NOT NULL,

    requires_reference BOOLEAN DEFAULT FALSE,
    is_cash_equivalent BOOLEAN DEFAULT FALSE,

    is_active BOOLEAN DEFAULT TRUE,
    is_available_for_deposits BOOLEAN DEFAULT TRUE,
    is_available_for_withdrawals BOOLEAN DEFAULT TRUE,

    max_transaction_amount DECIMAL(15, 2),
    requires_approval_above DECIMAL(15, 2),

    metadata JSONB,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(tenant_id, code)

);

CREATE TABLE IF NOT EXISTS mae__payment_method_rules (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    payment_method_id UUID NOT NULL REFERENCES mae__payment_methods(id)
    document_subtype_id UUID,
    user_role_id UUID REFERENCES scr__roles(id),

    is_allowed BOOLEAN DEFAULT TRUE,
    min_amount DECIMAL(15, 2) DEFAULT 0,
```

```
    max_amount DECIMAL(15, 2),

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(tenant_id, payment_method_id, document_subtype_id, user_role_id)
);

-- ============================================
-- 7. MAESTROS: CAJAS
-- ============================================

CREATE TABLE IF NOT EXISTS mae__cash_boxes (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    code VARCHAR(20) NOT NULL,
    name VARCHAR(100) NOT NULL,
    location VARCHAR(255),

    warehouse_id UUID REFERENCES mae__warehouses(id),

    box_type ENUM('cashier', 'deposit', 'return') DEFAULT 'cashier',

    is_active BOOLEAN DEFAULT TRUE,
    is_default BOOLEAN DEFAULT FALSE,

    metadata JSONB,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(tenant_id, code)
);

-- ============================================
-- 8. CONFIGURACIÓN DE DOCUMENTOS (Core)
-- ============================================

CREATE TABLE IF NOT EXISTS cfg__document_types (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
```

```
    code VARCHAR(10) NOT NULL, -- FAC, ORP, REC, NC
    name VARCHAR(50) NOT NULL,
    description TEXT,

    is_active BOOLEAN DEFAULT TRUE,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(tenant_id, code)
);

CREATE TABLE IF NOT EXISTS cfg__document_subtypes (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    document_type_id UUID NOT NULL REFERENCES cfg__document_types(id),

    code VARCHAR(10) NOT NULL, -- FAC-A, FAC-B, etc
    name VARCHAR(50) NOT NULL,

    prefix VARCHAR(4) NOT NULL, -- FAC, FCC, etc
    consecutive_format VARCHAR(20) DEFAULT 'NNNNNNNN', -- 8 dígitos

    is_active BOOLEAN DEFAULT TRUE,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(tenant_id, code),
    UNIQUE(tenant_id, prefix)
);

-- ============================================
-- 9. TRANSACCIONES (Core)
-- ============================================

CREATE TABLE IF NOT EXISTS trx__documents (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
```

```
  document_subtype_id UUID NOT NULL REFERENCES cfg__document_subtypes(i

  document_number VARCHAR(20) NOT NULL, -- FAC00000001
  document_date TIMESTAMP NOT NULL,

  client_id UUID NOT NULL REFERENCES mae__clients(id),
  created_by_user_id UUID NOT NULL REFERENCES scr__users(id),

  status ENUM('draft', 'finalized', 'paid', 'cancelled') DEFAULT 'draft',

  subtotal DECIMAL(15, 2) DEFAULT 0,
  total_discount DECIMAL(15, 2) DEFAULT 0,
  total_tax DECIMAL(15, 2) DEFAULT 0,
  grand_total DECIMAL(15, 2) DEFAULT 0,

  metadata JSONB,

  finalized_at TIMESTAMP,
  paid_at TIMESTAMP,
  cancelled_at TIMESTAMP,

  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

  UNIQUE(tenant_id, document_number),
  INDEX idx_doc_client (client_id, document_date),
  INDEX idx_doc_status (tenant_id, status)
);

CREATE TABLE IF NOT EXISTS trx__document_lines (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

  document_id UUID NOT NULL REFERENCES trx__documents(id) ON DELETE CA
  item_id UUID NOT NULL REFERENCES mae__items(id),

  line_number INT NOT NULL,
```

```
  quantity DECIMAL(15, 4) NOT NULL,
  unit_price DECIMAL(15, 4) NOT NULL,
  discount_percentage DECIMAL(5, 2) DEFAULT 0,
  line_total DECIMAL(15, 2),

  metadata JSONB,

  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

  UNIQUE(tenant_id, document_id, line_number)
);

CREATE TABLE IF NOT EXISTS trx__payments (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

  document_id UUID NOT NULL REFERENCES trx__documents(id),
  payment_method_id UUID NOT NULL REFERENCES mae__payment_methods(id)

  amount DECIMAL(15, 2) NOT NULL,
  payment_date TIMESTAMP NOT NULL,

  reference VARCHAR(100),
  status ENUM('pending', 'confirmed', 'failed', 'reversed') DEFAULT 'pending',

  processed_by_user_id UUID NOT NULL REFERENCES scr__users(id),

  metadata JSONB,

  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

  INDEX idx_payment_doc (document_id, status),
  INDEX idx_payment_method (payment_method_id, payment_date)
);

-- ============================================
-- 10. CAJA (POS MODULE)
```

```
-- ============================================

CREATE TABLE IF NOT EXISTS scr__cashier_sessions (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    session_number VARCHAR(20),
    cash_box_id UUID NOT NULL REFERENCES mae__cash_boxes(id),

    opened_by_user_id UUID NOT NULL REFERENCES scr__users(id),
    supervisor_user_id UUID REFERENCES scr__users(id),
    closed_by_user_id UUID REFERENCES scr__users(id),

    opened_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    closed_at TIMESTAMP,

    opening_balance DECIMAL(15, 2) NOT NULL DEFAULT 0,
    total_sales DECIMAL(15, 2) DEFAULT 0,
    total_cash_received DECIMAL(15, 2) DEFAULT 0,
    total_withdrawals DECIMAL(15, 2) DEFAULT 0,
    total_deposits DECIMAL(15, 2) DEFAULT 0,

    expected_balance DECIMAL(15, 2) GENERATED ALWAYS AS
        (opening_balance + total_cash_received - total_withdrawals + total_deposits) S

    physical_count DECIMAL(15, 2),
    variance DECIMAL(15, 2) GENERATED ALWAYS AS
        (physical_count - expected_balance) STORED,

    status ENUM('open', 'closing', 'closed', 'locked', 'deposited') DEFAULT 'open',

    closure_notes TEXT,
    supervisor_approval_at TIMESTAMP,
    supervisor_approval_notes TEXT,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
  UNIQUE(tenant_id, session_number),
  INDEX idx_session_status (tenant_id, status, closed_at)
);

CREATE TABLE IF NOT EXISTS caj__cash_movements (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

  session_id UUID NOT NULL REFERENCES scr__cashier_sessions(id),

  movement_type ENUM('payment', 'withdrawal', 'deposit', 'manual_adjustment')

  document_id UUID REFERENCES trx__documents(id) ON DELETE SET NULL,
  payment_id UUID REFERENCES trx__payments(id) ON DELETE SET NULL,

  amount DECIMAL(15, 2) NOT NULL,
  description VARCHAR(255),

  reference VARCHAR(100),
  metadata JSONB,

  created_by_user_id UUID NOT NULL REFERENCES scr__users(id),
  created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,

  INDEX idx_session_movements (session_id, created_at),
  INDEX idx_amount (amount)
);

CREATE TABLE IF NOT EXISTS caj__cash_withdrawals (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

  session_id UUID NOT NULL REFERENCES scr__cashier_sessions(id),
  withdrawal_document_id UUID NOT NULL REFERENCES trx__documents(id),

  withdrawal_type ENUM('cleaning', 'bank_deposit', 'manager_request', 'emergen

  amount DECIMAL(15, 2) NOT NULL,
```

```
    authorized_by_user_id UUID REFERENCES scr__users(id),

    destination_description VARCHAR(255),
    destination_warehouse_id UUID REFERENCES mae__warehouses(id),

    status ENUM('pending', 'approved', 'executed', 'cancelled') DEFAULT 'pending',

    executed_at TIMESTAMP,
    executed_by_user_id UUID REFERENCES scr__users(id),

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    INDEX idx_session_withdrawals (session_id, status)
);

CREATE TABLE IF NOT EXISTS caj__blind_counts (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    session_id UUID NOT NULL REFERENCES scr__cashier_sessions(id),

    performed_by_user_id UUID NOT NULL REFERENCES scr__users(id),
    verified_by_user_id UUID REFERENCES scr__users(id),

    started_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    finished_at TIMESTAMP,

    physical_count DECIMAL(15, 2) NOT NULL,

    discrepancy_amount DECIMAL(15, 2),
    discrepancy_percentage DECIMAL(5, 2),
    discrepancy_reason_code VARCHAR(50),

    notes TEXT,
    status ENUM('pending_approval', 'approved', 'disputed', 'resolved') DEFAULT 'pe

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
    INDEX idx_session_counts (session_id, status)
);

-- ============================================
-- 11. AUDITORÍA
-- ============================================

CREATE TABLE IF NOT EXISTS audit__transaction_log (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    entity_type VARCHAR(100) NOT NULL,
    entity_id UUID NOT NULL,

    operation ENUM('create', 'read', 'update', 'delete') NOT NULL,

    old_values JSONB,
    new_values JSONB,

    performed_by_user_id UUID NOT NULL REFERENCES scr__users(id),
    ip_address INET,
    user_agent TEXT,

    timestamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,

    INDEX idx_entity (entity_type, entity_id),
    INDEX idx_user_timestamp (performed_by_user_id, timestamp),
    INDEX idx_operation (operation, timestamp)
);

CREATE TABLE IF NOT EXISTS audit__permission_changes (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    entity_type ENUM('role', 'permission', 'user_role', 'user_permission', 'data_scope
    entity_id UUID NOT NULL,

    operation ENUM('create', 'update', 'delete', 'assign', 'revoke') NOT NULL,
```

```
    performed_by_user_id UUID NOT NULL REFERENCES scr__users(id),

    old_values JSONB,
    new_values JSONB,

    change_reason TEXT,
    ip_address INET,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    INDEX idx_entity (entity_type, entity_id),
    INDEX idx_performed_by (performed_by_user_id, created_at)
);

-- ============================================
-- ÍNDICES FINALES Y OPTIMIZACIONES
-- ============================================

CREATE INDEX IF NOT EXISTS idx_tenants_active ON tenants(is_active);
CREATE INDEX IF NOT EXISTS idx_users_tenant_email ON scr__users(tenant_id, email);
CREATE INDEX IF NOT EXISTS idx_roles_tenant_active ON scr__roles(tenant_id, is_active);
CREATE INDEX IF NOT EXISTS idx_permissions_tenant_category ON
scr__permissions(tenant_id, category);
CREATE INDEX IF NOT EXISTS idx_items_tenant_code ON mae__items(tenant_id, code);
CREATE INDEX IF NOT EXISTS idx_clients_tenant_code ON mae__clients(tenant_id, code);
CREATE INDEX IF NOT EXISTS idx_documents_tenant_date ON trx__documents(tenant_id,
document_date DESC);
CREATE INDEX IF NOT EXISTS idx_payments_date_method ON
trx__payments(payment_date, payment_method_id);
CREATE INDEX IF NOT EXISTS idx_movements_session_date ON
caj__cash_movements(session_id, created_at);
CREATE INDEX IF NOT EXISTS idx_audit_timestamp ON audit__transaction_log(timestamp
DESC);

-- ============================================
-- VISTAS ÚTILES (Opcional)
-- ============================================

CREATE OR REPLACE VIEW v_user_permissions AS
SELECT
u.id AS user_id,
u.email,
p.id AS permission_id,
p.code AS permission_code,
p.name AS permission_name,
```

```
'role' AS permission_source
FROM scr__users u
INNER JOIN scr__user_has_roles ur ON u.id = ur.user_id AND ur.is_active = TRUE
INNER JOIN scr__role_has_permissions rp ON ur.role_id = rp.role_id
INNER JOIN scr__permissions p ON rp.permission_id = p.id AND p.is_active = TRUE

UNION ALL

SELECT
u.id AS user_id,
u.email,
p.id AS permission_id,
p.code AS permission_code,
p.name AS permission_name,
'direct' AS permission_source
FROM scr__users u
INNER JOIN scr__user_has_permissions up ON u.id = up.user_id
INNER JOIN scr__permissions p ON up.permission_id = p.id AND p.is_active = TRUE;

CREATE OR REPLACE VIEW v_document_summary AS
SELECT
d.id,
d.document_number,
d.document_date,
c.name AS client_name,
st.name AS document_subtype,
d.grand_total,
d.status,
COUNT(DISTINCT dl.id) AS line_count,
COUNT(DISTINCT p.id) AS payment_count
FROM trx__documents d
LEFT JOIN mae__clients c ON d.client_id = c.id
LEFT JOIN cfg__document_subtypes st ON d.document_subtype_id = st.id
LEFT JOIN trx__document_lines dl ON d.id = dl.document_id
LEFT JOIN trx__payments p ON d.id = p.document_id
GROUP BY d.id, c.name, st.name;
```

---

# 8. HISTORIAS DE USUARIO TÉCNICAS

## 8.1 HU-SEC-001: Crear Usuario y Asignar Rol

HISTORIA: Como administrador, necesito crear un usuario (cajero)
y asignarle un rol con permisos específicos.

ESCENARIO DE ACEPTACIÓN:

DADO que soy administrador del tenant
Y accedo al módulo de Seguridad (module:security:view)

CUANDO creo un nuevo usuario:
POST /api/security/users

{
"email": "juan.perez@empresa.com",
"name": "Juan Pérez",
"password": "Segura123!",
"department": "Caja",
"phone": "+573001234567"
}

ENTONCES:
✅ El sistema asigna un UUID único al usuario
✅ Se crea un registro en scr__users
✅ Se retorna usuario creado con status 201

Y CUANDO le asigno el rol "cashier":
POST /api/security/users/{user_id}/roles
{
"role_id": "{role_cashier_uuid}",
"expires_at": null,
"notes": "Asignado como operario de caja principal"
}

ENTONCES:
✅ El sistema crea registro en scr__user_has_roles
✅ Se audita el cambio en audit__permission_changes
✅ El usuario ahora hereda permisos del rol cashier:
- procedure:open_cashier_session:execute
- entity:documents:read
- action:document:create (solo tipo RECEIPT)
- scope:own_cashier_session

Y el usuario NO puede:
❌ Abrir otras sesiones (scope restricts)
❌ Crear documentos de tipo INVOICE (solo RECEIPT)
❌ Acceder al módulo de Seguridad (module:security:view)

VALIDACIONES BACKEND:

1. Email debe ser único por tenant
2. Contraseña debe cumplir política de seguridad
3. Rol debe existir y estar activo
4. Usuario que asigna rol debe tener "entity:roles:update"
5. Todas operaciones se loguean en audit

## 8.2 HU-POS-001: Retiro de Efectivo (Sangría) Documentado

HISTORIA: Como cajero autorizado, necesito hacer un retiro de efectivo de la caja (sangría) que genere automáticamente un documento de respaldo para auditoría.

ESCENARIO DE ACEPTACIÓN:

DADO que soy cajero con sesión ABIERTA
Y tengo $2,500,000 acumulado en caja

Y la sesión tiene estado "open"

CUANDO intento realizar un retiro de $500,000:
POST /api/cashier/sessions/{session_id}/withdrawals
{
"amount": 500000,
"withdrawal_type": "cleaning",
"destination_description": "A bóveda principal",
"authorized_by_user_id": "{supervisor_id}"
}

ENTONCES el sistema:

PASO 1: Valida permisos
✓ Usuario tiene "procedure:cash_withdrawal:execute"
✓ Usuario está en sesión ABIERTA
✓ El monto no excede cantidad disponible en caja
✓ Existe supervisor_id válido para autorizar

PASO 2: Genera documento de respaldo automático
├─> Se crea transacción en trx__documents
│ ├─> document_type = "SANGRÍA" (subtype especial)
│ ├─> client_id = "SISTEMA" (cliente interno)
│ ├─> grand_total = $500,000
│ ├─> status = "finalized" (inmediatamente)
│ └─> metadata = {
│ "withdrawal_session_id": "{session_id}",
│ "withdrawal_type": "cleaning",
│ "authorized_by": "{supervisor_id}"
│ }
└─> Se genera documento_number automático (ej: SANG000001)

PASO 3: Registra en tabla de retiros
└─> Nuevo registro en caj__cash_withdrawals
├─> session_id = {current_session}
├─> withdrawal_document_id = {documento_sangría}
├─> amount = $500,000
├─> status = "pending" (requiere aprobación supervisor)
└─> authorized_by_user_id = {supervisor_id}

PASO 4: Crea movimiento de caja
└─> Registro en caj__cash_movements
├─> session_id = {current_session}
├─> movement_type = "withdrawal"
├─> amount = $500,000
├─> document_id = {documento_sangría}
└─> description = "Retiro a bóveda"

PASO 5: Actualiza saldo esperado de sesión
└─> scr__cashier_sessions.total_withdrawals += $500,000
└─> expected_balance recalcula automáticamente

RETORNA:
✅ 201 Created
{
"withdrawal_id": "{uuid}",
"withdrawal_document_id": "{uuid}",
"document_number": "SANG000001",
"amount": 500000,
"status": "pending_approval",
"message": "Retiro creado. Requiere aprobación del supervisor."
}

PASO 6: Supervisor aprueba/rechaza
PUT /api/cashier/withdrawals/{withdrawal_id}/approve
{
"approved": true,
"notes": "Retiro autorizado - Se traslada a bóveda"
}

✅ Estado cambia a "approved"
✅ Se audita quién lo aprobó
✅ Documento de respaldo queda sellado

VALIDACIONES AUDITORÍA:

- Cada retiro genera un documento trazable
- Supervisor debe co-firmar electronicamente
- IP y user-agent se registran
- Caja nunca queda sin supervisor presente
- Reporte de "Sangrías del Día" está disponible

## 8.3 HU-MASTER-001: Cambio de SKU Sin Romper Integridad

HISTORIA: Como gerente de inventario, necesito cambiar el código
SKU de un producto (ej: SKU-LAPTOP-001 → PROD-2025-LAPTOP-001)
sin que los documentos históricos se vean afectados.

ESCENARIO DE ACEPTACIÓN:

DADO que tengo un producto con:

- id = "550e8400-e29b-41d4-a716-446655440001"
- code = "SKU-LAPTOP-001"
- name = "Laptop Dell XPS 13"

Y este producto aparece en:

- 500 documentos históricos
- 50 líneas de transacciones

CUANDO cambio el SKU:
PATCH /api/masters/items/{item_uuid}
{

"code": "PROD-2025-LAPTOP-001"
}

ENTONCES el sistema:

PASO 1: Valida que el nuevo código sea único
✓ No existe otro item con code = "PROD-2025-LAPTOP-001"
✓ Búsqueda en mae__items WHERE code = X es única por tenant

PASO 2: Actualiza solo la columna code
UPDATE mae__items
SET code = "PROD-2025-LAPTOP-001",
updated_by = {user_id},
updated_at = NOW()
WHERE id = "550e8400-e29b-41d4-a716-446655440001"
AND tenant_id = {tenant_id}

PASO 3: NINGÚN cambio en tablas relacionadas
└─> trx__document_lines siguen usando item_id (UUID)
└─> Histórico NO es afectado
└─> Reportes siguen siendo consistentes

PASO 4: Audita el cambio
INSERT INTO audit__transaction_log
VALUES (
entity_type = "item",
entity_id = "550e8400...",
operation = "update",
old_values = {"code": "SKU-LAPTOP-001"},
new_values = {"code": "PROD-2025-LAPTOP-001"},
performed_by_user_id = {user_id}
)

RETORNA:
✓ 200 OK
{
"id": "550e8400-e29b-41d4-a716-446655440001",
"code": "PROD-2025-LAPTOP-001",
"name": "Laptop Dell XPS 13",
"updated_at": "2025-01-25T16:45:30Z",
"audit_log_id": "{uuid}"
}

VERIFICACIÓN POST-CAMBIO:

- Dashboard sigue mostrando producto correctamente
- Búsquedas por nuevo código funcionan
- Búsquedas por código antiguo ya NO funcionan (✓ Correcto)
- Histórico de transacciones: SIN cambios
- Próximas facturas usan nuevo código

---

# 9. ARQUITECTURA DE APIs RESTful EXTENDIDA

## 9.1 Prefijos y Estructura

BASE_URL = https://api.farutech.local/v1

```
├── /auth
│   ├── POST /auth/login [Autenticación]
│   ├── POST /auth/logout
│   ├── POST /auth/refresh-token
│   └── GET /auth/me [Perfil actual]
│
├── /security
│   ├── GET /security/users [Listar usuarios]
│   ├── POST /security/users [Crear usuario]
│   ├── GET /security/users/{id} [Obtener usuario]
│   ├── PATCH /security/users/{id} [Actualizar usuario]
│   ├── DELETE /security/users/{id} [Desactivar usuario]
│   │
│   ├── POST /security/users/{id}/roles [Asignar rol]
│   ├── DELETE /security/users/{id}/roles/{role_id} [Revocar rol]
│   │
│   ├── GET /security/roles [Listar roles]
│   ├── POST /security/roles [Crear rol]
│   ├── GET /security/roles/{id} [Obtener rol]
│   ├── PATCH /security/roles/{id} [Editar rol]
│   │
│   ├── GET /security/permissions [Listar permisos]
│   ├── POST /security/permissions [Crear permiso]
│   ├── GET /security/permissions/{id}
│   │
│   └── POST /security/permissions/bulk-assign [Asignar múltiples permisos a rol]
│
├── /masters
│   ├── /masters/items
│   │   ├── GET /masters/items [Listar items]
│   │   ├── POST /masters/items [Crear item]
│   │   ├── GET /masters/items/{id}
│   │   ├── PATCH /masters/items/{id}
│   │   ├── DELETE /masters/items/{id}
│   │   └── GET /masters/items/{id}/stock [Stock por almacén]
│   │
│   ├── /masters/categories
│   │   ├── GET /masters/categories
│   │   ├── POST /masters/categories
│   │   └── PATCH /masters/categories/{id}
│   │
│   ├── /masters/clients
│   │   ├── GET /masters/clients
│   │   ├── POST /masters/clients
│   │   ├── GET /masters/clients/{id}
```

```
│  │  ├── PATCH /masters/clients/{id}
│  │  └── GET /masters/clients/{id}/credit-limit [Verificar límite crédito]
│  │
│  ├── /masters/warehouses
│  │  ├── GET /masters/warehouses
│  │  ├── POST /masters/warehouses
│  │  ├── PATCH /masters/warehouses/{id}
│  │  └── GET /masters/warehouses/{id}/stock [Inventario del almacén]
│  │
│  ├── /masters/units-of-measure
│  │  ├── GET /masters/units
│  │  └── POST /masters/units
│  │
│  ├── /masters/conversion-factors
│  │  ├── GET /masters/items/{id}/conversions
│  │  ├── POST /masters/items/{id}/conversions
│  │  └── DELETE /masters/items/{id}/conversions/{factor_id}
│  │
│  ├── /masters/price-lists
│  │  ├── GET /masters/price-lists
│  │  ├── POST /masters/price-lists
│  │  ├── GET /masters/price-lists/{id}/items [Items en lista]
│  │  └── POST /masters/price-lists/{id}/items [Agregar items]
│  │
│  ├── /masters/payment-methods
│  │  ├── GET /masters/payment-methods
│  │  ├── POST /masters/payment-methods
│  │  ├── GET /masters/payment-methods/{id}/rules [Reglas]
│  │  └── POST /masters/payment-methods/{id}/rules [Crear regla]
│  │
│  └── /masters/cash-boxes
│  ├── GET /masters/cash-boxes
│  ├── POST /masters/cash-boxes
│  └── GET /masters/cash-boxes/{id}/current-session
│
├── /documents
│  ├── GET /documents [Listar documentos]
│  ├── POST /documents [Crear documento]
│  ├── GET /documents/{id} [Obtener documento]
│  ├── PATCH /documents/{id} [Editar documento]
│  ├── DELETE /documents/{id} [Anular documento]
│  ├── POST /documents/{id}/finalize [Marcar como finalizado]
│  ├── POST /documents/{id}/lines [Agregar línea]
│  ├── PATCH /documents/{id}/lines/{line_id} [Editar línea]
│  ├── DELETE /documents/{id}/lines/{line_id} [Eliminar línea]
│  ├── POST /documents/{id}/payments [Agregar pago]
│  ├── GET /documents/{id}/payments [Listar pagos del doc]
│  └── GET /documents/search?client={id}&status={status}&date_from={date} [Búsqueda
avanzada]
│
├── /cashier
```

```
│  ├── POST /cashier/sessions [Abrir caja]
│  ├── GET /cashier/sessions [Listar sesiones del usuario]
│  ├── GET /cashier/sessions/{id} [Detalle sesión]
│  ├── GET /cashier/sessions/{id}/summary [Resumen: dinero esperado vs. contado]
│  ├── POST /cashier/sessions/{id}/blind-count [Realizar arqueo ciego]
│  ├── POST /cashier/sessions/{id}/close [Cerrar caja]
│  ├── GET /cashier/sessions/{id}/movements [Movimientos del día]
│  ├── POST /cashier/sessions/{id}/withdrawals [Solicitar retiro]
│  ├── GET /cashier/withdrawals/{id} [Obtener retiro]
│  ├── PUT /cashier/withdrawals/{id}/approve [Supervisor aprueba]
│  └── GET /cashier/reconciliation?date={YYYY-MM-DD} [Cuadre diario]
│
├── /reporting
│  ├── GET /reporting/sales-summary [Ventas por período]
│  ├── GET /reporting/inventory-status [Estado de inventario]
│  ├── GET /reporting/cash-reconciliation [Cuadre de caja]
│  ├── GET /reporting/audit-trail [Histórico de cambios]
│  ├── POST /reporting/export?format={pdf|excel} [Exportar reportes]
│  └── GET /reporting/discrepancies [Reportar faltantes/sobrantes]
│
└── /admin
├── GET /admin/health [Health check]
├── GET /admin/config [Configuración del tenant]
├── PATCH /admin/config [Actualizar configuración]
├── GET /admin/audit-logs [Logs de auditoría (Admin only)]
├── POST /admin/backup [Generar backup]
└── GET /admin/stats [Estadísticas del tenant]
```

## 9.2 Ejemplos de Requests/Responses

**POST /documents (Crear Documento)**

REQUEST:

```
{
"document_subtype_id": "uuid-invoice",
"client_id": "uuid-cliente-1",
"document_date": "2025-01-25T14:30:00Z",
"lines": [
{
"item_id": "uuid-laptop",
"quantity": 2,
"unit_price": 1500000,
"discount_percentage": 10,
"line_number": 1
},
{
"item_id": "uuid-service",
"quantity": 1,
"unit_price": 500000,
"discount_percentage": 0,
"line_number": 2
```

```
}
],
"metadata": {
"purchase_order": "PO-2025-001",
"delivery_address": "Cra 7 # 25-50, Bogotá"
}
}
```


RESPONSE (201 Created):


```
{
"id": "uuid-doc-001",
"document_number": "FAC00000001",
"document_date": "2025-01-25T14:30:00Z",
"client_id": "uuid-cliente-1",
"status": "draft",
"lines": [
{
"id": "uuid-line-1",
"item_id": "uuid-laptop",
"quantity": 2,
"unit_price": 1500000,
"discount_percentage": 10,
"line_total": 2700000,
"tax": 513000
},
{
"id": "uuid-line-2",
"item_id": "uuid-service",
"quantity": 1,
"unit_price": 500000,
"discount_percentage": 0,
"line_total": 500000,
"tax": 0
}
],
"summary": {
"subtotal": 3200000,
"total_discount": 150000,
"total_tax": 513000,
"grand_total": 3563000
},
"created_by": "uuid-user",
"created_at": "2025-01-25T14:30:00Z",
"links": {
"self": "/api/v1/documents/uuid-doc-001",
"finalize": "/api/v1/documents/uuid-doc-001/finalize",
"add_payment": "/api/v1/documents/uuid-doc-001/payments"
}
}
```


**POST /cashier/sessions/{id}/withdrawals (Sangría)**

REQUEST:
{
"amount": 500000,
"withdrawal_type": "cleaning",
"destination_description": "A bóveda principal",
"authorized_by_user_id": "uuid-supervisor"
}

RESPONSE (201 Created):
{
"id": "uuid-withdrawal-001",
"session_id": "uuid-session-caja1",
"withdrawal_document_id": "uuid-doc-sangria",
"document_number": "SANG000001",
"amount": 500000,
"withdrawal_type": "cleaning",
"status": "pending_approval",
"created_by": "uuid-cajero",
"created_at": "2025-01-25T16:45:00Z",
"message": "Retiro creado. Pendiente de aprobación del supervisor."
}

**GET /cashier/sessions/{id}/summary (Resumen de Caja)**

RESPONSE (200 OK):
{
"session_id": "uuid-session-001",
"session_number": "CAJA-20250125-001",
"cash_box": "CAJA-1",
"cashier": "Juan Pérez",
"opened_at": "2025-01-25T08:15:00Z",
"status": "open",
"duration_hours": 8.5,

"balance": {
"opening_balance": 500000,
"total_sales": 2350000,
"total_withdrawals": -150000,
"expected_balance": 2700000,
"physical_count": null,
"variance": null
},

"movements_summary": {
"total_transactions": 47,
"total_payments": 47,
"payment_methods": {
"cash": 2250000,
"card": 100000,
"check": 0
}
},

"pending_items": {
"blind_count_pending": true,
"withdrawals_pending": 1,
"supervisor_approval_pending": true
}
}

---

# 10. ESTRATEGIA DE DESARROLLO PARALELO

## 10.1 Estructura de Equipos

EQUIPO 1: CORE TRANSACCIONAL (Backend .NET)
├── Líder: Arquitecto Backend
├── Integración: Motor de Documentos
├── Responsables de:
│ ├── Tablas cfg__document_types/subtypes
│ ├── Tablas trx__documents/lines
│ ├── Lógica de validación y cálculos
│ ├── APIs /documents/*
│ └── Consecutivos y renumeración automática
└── Entregables: Controllers, Services, Validators

EQUIPO 2: SEGURIDAD + MAESTROS (Backend .NET)
├── Líder: Arquitecto de Seguridad
├── Integración: RBAC + Datos Maestros
├── Responsables de:
│ ├── Tablas scr__* (users, roles, permissions, scopes)
│ ├── Tablas mae__* (items, categories, clients, suppliers, etc)
│ ├── Middleware de autenticación/autorización
│ ├── APIs /security/* y /masters/*
│ ├── Data filtering por scope
│ └── Auditoría de cambios en seguridad
└── Entregables: Auth Services, Data Access Layer, Policies

EQUIPO 3: CAJA + REPORTES (Backend .NET)
├── Líder: Especialista Financiero/Operativo
├── Integración: POS + Auditoría
├── Responsables de:
│ ├── Tablas scr__cashier_sessions
│ ├── Tablas caj__* (movements, withdrawals, blind_counts)
│ ├── Ciclo completo: apertura → operación → cierre → auditoría
│ ├── APIs /cashier/* y /reporting/*
│ ├── Generación automática de documentos (sangrías)
│ ├── Cálculos de discrepancia y cuadre
│ └── Reportes analíticos
└── Entregables: POS Services, Report Generators, Reconciliation Engine

EQUIPO 4: FRONTEND (React 19 + Vite)
├── Líder: Senior React Developer
├── Integración: Consumo de APIs
├── Responsables de:

│  ├── Módulo de Documentos (crear, editar, visualizar, pagar)
│  ├── Módulo de Seguridad (gestión de usuarios, roles, permisos)
│  ├── Módulo de Maestros (CRUD de items, clientes, bodegas, etc)
│  ├── Módulo de Caja (sesiones, movimientos, retiros, cierre)
│  ├── Dashboards y reportes
│  ├── Autenticación y manejo de tokens JWT
│  └── Validación de forma cliente
└── Entregables: Components, Pages, Services (API Client), State Management

EQUIPO 5: QA + INFRAESTRUCTURA
├── Líder: QA Lead
├── Integración: Testing + DevOps
├── Responsables de:
│  ├── Pruebas automatizadas (Jest, xUnit)
│  ├── Pruebas de integración API
│  ├── Pruebas de seguridad (OWASP)
│  ├── Pruebas de performance (carga)
│  ├── Dockerización (.NET + PostgreSQL)
│  ├── CI/CD pipeline (GitHub Actions / GitLab)
│  └── Documentación técnica
└── Entregables: Test Suites, CI/CD workflows, Deployment Scripts

### 10.2 Dependencias entre Equipos

FASE 1 (Semanas 1-3): PARALLELIZABLE SIN BLOQUEOS
├──> Equipo 2 (Seguridad): Tablas de usuarios, roles, permisos
│  └──> Genera: Migration scripts, User/Permission models
│  └──> Frontend consume: API de autenticación (stubs iniciales)
│
├──> Equipo 1 (Core): Tablas de configuración de documentos
│  └──> Genera: Document type models, validation rules
│  └──> Frontend consume: Document type definitions
│
├──> Equipo 2 (Maestros): Tablas de items, clientes, bodegas
│  └──> Genera: Master data models, seeders
│  └──> Frontend consume: CRUD forms (UI mockups)
│
└──> Equipo 5 (QA): Setup de ambiente, Docker, repositories

FASE 2 (Semanas 4-8): INTEGRACIÓN GRADUAL
├──> Equipo 1 + Equipo 2: APIs de Documentos + Seguridad
│  └──> Dependency: Equipo 2 debe terminar /security/users
│  └──> Equipo 1 puede proceder con /documents/* paralelo
│
├──> Equipo 3: Tablas de Caja
│  └──> Dependency: Equipo 1 (/documents creados)
│  └──> Porque: Sangrías generan documentos automáticos
│
├──> Equipo 4 (Frontend): Componentes de Input
│  └──> Dependency: APIs de Maestros + Documentos
│  └──> Puede proceder con: UI mockups en paralelo

```
│
└─> Equipo 5 (QA): Tests unitarios y de integración

FASE 3 (Semanas 9-14): FUNCIONALIDADES COMPLEJAS
├─> Equipo 3 + Equipo 1: Ciclo de Caja completo
│   └─> Apertura → Pagos (integración con Equipo 1)
│   └─> Cierre → Arqueo → Retiros (generación de documentos)
│
├─> Equipo 4 (Frontend): Módulos completos
│   └─> Dashboard de Documentos
│   └─> Dashboard de Caja (tiempo real)
│   └─> Formularios CRUD complejos
│
└─> Equipo 5 (QA): E2E tests

FASE 4 (Semanas 15-18): OPTIMIZACIÓN Y HARDENING
├─> Equipo 1: Performance tuning de queries
├─> Equipo 2: Auditoría + compliance
├─> Equipo 3: Reportes avanzados
├─> Equipo 4: UX/UI polish
└─> Equipo 5: Load testing, penetration testing
```

### 10.3 Matriz de Bloqueos y Mitigación

| Equipo | Riesgo | Depende De | Mitigación |
|---|---|---|---|
| Equipo 1 | Cambios en definición de documentos | Requerimientos | **Usar feature flags** para nuevos tipos |
| Equipo 2 | Seguridad insuficiente | Diseño RBAC | **Inicio rápido con roles básicos**, evolucionar |
| Equipo 3 | Integridad de caja comprometida | Equipo 1 (docs) | **Stubs de documentos**, integration tests |
| Equipo 4 | APIs no disponibles | Equipos 1,2,3 | **Mock APIs iniciales**, Swagger/OpenAPI |
| Equipo 5 | Ambiente inestable | Todos | **Docker Compose local**, reproducibilidad |

---

# 11. TIMELINE REALISTA ACTUALIZADO (24 Semanas)

## 11.1 Desglose Detallado por Sprint

```
┌──────────────────────────────────────────────────────────────────────────────┐
│ ROADMAP FARUTECH VERSIÓN 2.0 - 24 SEMANAS (6 MESES) │
└──────────────────────────────────────────────────────────────────────────────┘

FASE 1: CIMIENTOS (Semanas 1-4)
═══════════════════════════════════════════════════════════════════════════════

SEMANA 1: SETUP + DISEÑO BD
├─ [Equipo 5] Configurar GitHub, Docker, CI/CD base
├─ [Todos] Refinamiento de historias de usuario
├─ [Equipo DB] Finalizar diagrama E-R en PostgreSQL
├─ [Equipo 1] Definir estructura de carpetas Backend
├─ [Equipo 4] Setup Vite + React 19 + TypeScript
└─ HITO: Ambientes (DEV, STAGING, PROD) operacionales

SEMANA 2: TABLAS BASE + USUARIOS
├─ [Equipo 5] Crear script de migración BD
├─ [Equipo 2] Implementar tabla tenants + scr__users
├─ [Equipo 2] Implementar tablas RBAC (roles, permissions, pivotes)
├─ [Equipo 1] Estructura de models (.NET)
├─ [Equipo 4] Diseño de login (UI)
└─ HITO: Autenticación básica funcional (JWT)

SEMANA 3: MAESTROS - ITEMS + CATEGORÍAS
├─ [Equipo 2] Tablas mae__item_categories
├─ [Equipo 2] Tablas mae__units_of_measure
├─ [Equipo 2] Tablas mae__items (completo)
├─ [Equipo 2] Tablas mae__conversion_factors
├─ [Equipo 1] Controllers de Items (CRUD)
├─ [Equipo 4] Formularios de Items (UI)
└─ HITO: CRUD de items funcional (sin seguridad granular aún)

SEMANA 4: MAESTROS - TERCEROS + LOGÍSTICA
├─ [Equipo 2] Tablas mae__clients + mae__suppliers
├─ [Equipo 2] Tablas mae__warehouses + mae__warehouse_stock
├─ [Equipo 2] Tablas mae__price_lists + items
├─ [Equipo 1] Controllers de Clientes y Bodegas
├─ [Equipo 4] Formularios de Terceros
├─ [Equipo 5] Tests unitarios de modelos
└─ HITO: Maestros de datos operacionales (V1)

FASE 2: CORE TRANSACCIONAL (Semanas 5-10)
═══════════════════════════════════════════════════════════════════════════════
```

SEMANA 5: CONFIGURACIÓN DE DOCUMENTOS
├─ [Equipo 1] Tablas cfg__document_types
├─ [Equipo 1] Tablas cfg__document_subtypes + subtypes_config
├─ [Equipo 1] Lógica de generación de consecutivos
├─ [Equipo 1] Validators (campos obligatorios por subtipo)
├─ [Equipo 2] Actualizarentity:document:*
├─ [Equipo 5] Tests de consecutivos
└─ HITO: Motor parametrizable de documentos

SEMANA 6: TRANSACCIONES - DOCUMENTOS
├─ [Equipo 1] Tablas trx__documents
├─ [Equipo 1] Tablas trx__document_lines
├─ [Equipo 1] Controllers POST /documents
├─ [Equipo 1] Lógica de cálculos (subtotal, descuentos, impuestos)
├─ [Equipo 1] Validaciones de cliente + items existentes
├─ [Equipo 4] UI Crear/Editar documento (Draft)
└─ HITO: Crear documentos en estado DRAFT

SEMANA 7: TRANSACCIONES - PAGOS
├─ [Equipo 2] Tablas mae__payment_methods + rules
├─ [Equipo 1] Tablas trx__payments
├─ [Equipo 1] Controllers POST /documents/{id}/payments
├─ [Equipo 1] Validación de formas de pago permitidas (RBAC)
├─ [Equipo 1] Lógica de finalización de documento
├─ [Equipo 4] UI de pagos (modal/formulario)
├─ [Equipo 5] Tests de cálculos tributarios DIAN
└─ HITO: Documentos con pagos -> status PAID

SEMANA 8: AUDITORÍA + DOCUMENT-LEVEL SECURITY
├─ [Equipo 2] Tablas audit__transaction_log
├─ [Equipo 2] Middleware de auditoría (interceptar todos los cambios)
├─ [Equipo 2] Implementar data scopes (scope:own_documents)
├─ [Equipo 1] Filtrar documentos por usuario (RBAC)
├─ [Equipo 4] Historial de cambios en documento (UI)
├─ [Equipo 5] Tests de auditoría
└─ HITO: Auditoría centralizadaactiva

SEMANA 9: BÚSQUEDAS + REPORTES BÁSICOS
├─ [Equipo 1] Implementar /documents/search (filtros avanzados)
├─ [Equipo 1] Endpoints /reporting/sales-summary
├─ [Equipo 1] Endpoints /reporting/inventory-status
├─ [Equipo 4] Dashboard de documentos (listar, buscar, filtrar)
├─ [Equipo 4] Primeros reportes (tabla datos)
└─ HITO: Visibilidad sobre documentos y datos

SEMANA 10: STRESS TEST + OPTIMIZACIÓN
├─ [Equipo 5] Performance testing (1M transacciones)
├─ [Equipo 1] Tuning de queries (índices, JSONB)
├─ [Equipo 4] Optimización de carga de datos (paginación)
├─ [Equipo 5] Load testing de API
└─ HITO: Sistema escalable hasta 100k docs/mes

FASE 3: CAJA + POS (Semanas 11-16)
═══════════════════════════════════════════════════════════════════════

SEMANA 11: SETUP DE CAJA
├─ [Equipo 2] Tabla mae__cash_boxes
├─ [Equipo 2] Tabla scr__cashier_sessions
├─ [Equipo 1] Controllers POST /cashier/sessions (abrir)
├─ [Equipo 2] Data scopes para cajas (scope:own_cashier_session)
├─ [Equipo 4] UI de apertura de caja
└─ HITO: Abrir caja operacional

SEMANA 12: MOVIMIENTOS DE CAJA (Integración con Documentos)
├─ [Equipo 3] Tabla caj__cash_movements
├─ [Equipo 3] Trigger: Pago CASH -> Automático movimiento en caja
├─ [Equipo 1] Actualizar balance_session en real-time
├─ [Equipo 3] Controllers GET /cashier/sessions/{id}/movements
├─ [Equipo 4] Dashboard de caja en tiempo real
├─ [Equipo 5] Tests de integración documento ↔ caja
└─ HITO: Dinero flows automáticamente a caja

SEMANA 13: RETIROS DE EFECTIVO (SANGRÍAS)
├─ [Equipo 3] Tabla caj__cash_withdrawals
├─ [Equipo 3] Lógica: Sangría genera documento automático
├─ [Equipo 3] Controllers POST /cashier/sessions/{id}/withdrawals
├─ [Equipo 1] Generar documento SANGRÍA (subtype especial)
├─ [Equipo 3] Integración: Retiro → Actualiza balance_session
├─ [Equipo 4] UI Solicitar retiro + Aprobaciones
├─ [Equipo 5] Tests end-to-end de sangrías
└─ HITO: Sangrías documentadas y auditables

SEMANA 14: ARQUEO DE CAJA (BLIND COUNT)
├─ [Equipo 3] Tabla caj__blind_counts
├─ [Equipo 3] Controllers POST /cashier/sessions/{id}/blind-count
├─ [Equipo 3] Cálculo de discrepancia (physical vs expected)
├─ [Equipo 3] Lógica de aprobación por supervisor
├─ [Equipo 4] UI Arqueo ciego (formulario simple)
├─ [Equipo 3] Genera alertas si discrepancia > umbral (ej: 2%)
└─ HITO: Arqueo ciego implementado

SEMANA 15: CIERRE DE CAJA + CUADRE
├─ [Equipo 3] Controllers POST /cashier/sessions/{id}/close
├─ [Equipo 3] Lógica de cierre (validar arqueo, calcular varianza)
├─ [Equipo 3] Endpoints GET /cashier/reconciliation
├─ [Equipo 3] Generar reporte "Cuadre de Caja"
├─ [Equipo 1] Impresión de cuadre
├─ [Equipo 4] Pantalla de cierre (supervisado)
├─ [Equipo 5] Tests de casos excepcionales (faltantes, sobrantes)
└─ HITO: Ciclo completo de caja operacional

SEMANA 16: POS HARDENING + FRAUD PREVENTION
├─ [Equipo 3] Validaciones anti-fraude (monto máximo, límites)

├─ [Equipo 2] Restricciones de rol (solo ciertos cajeros en cajas)
├─ [Equipo 3] Auditoría detallada de cada movimiento
├─ [Equipo 5] Escenarios de fraude (pruebas)
├─ [Equipo 4] Alertas visuales en dashboard
└─ HITO: Sistema de caja resiliente

FASE 4: SEGURIDAD GRANULAR + REPORTES (Semanas 17-20)
═══════════════════════════════════════════════════════

SEMANA 17: RBAC AVANZADO
├─ [Equipo 2] Tabla scr__data_scopes (completa)
├─ [Equipo 2] Implementar data filtering en todas las queries
├─ [Equipo 2] Validación de permisos en cada endpoint
├─ [Equipo 2] Tests de permisos granulares
├─ [Equipo 4] Ocultar/habilitar opciones en UI según permisos
└─ HITO: RBAC tipo ERP fully functional

SEMANA 18: AUDITORÍA DE SEGURIDAD
├─ [Equipo 2] Tabla audit__permission_changes
├─ [Equipo 2] Logging de cambios en roles/permisos
├─ [Equipo 1] Endpoints /admin/audit-logs (Admin only)
├─ [Equipo 4] Interfaz de auditoría (búsqueda, filtros)
└─ HITO: Compliance total

SEMANA 19: REPORTES AVANZADOS
├─ [Equipo 3] Endpoints /reporting/* (todos)
├─ [Equipo 3] Reportes: Ventas, Inventario, Caja, Auditoría
├─ [Equipo 1] Exportación a PDF/Excel
├─ [Equipo 4] Dashboards con gráficos
├─ [Equipo 5] Tests de reportes
└─ HITO: Business Intelligence disponible

SEMANA 20: SEGURIDAD EN PROFUNDIDAD
├─ [Equipo 5] Penetration testing
├─ [Equipo 2] OWASP top 10 review
├─ [Equipo 1] Validación de inputs (prevención de SQL injection)
├─ [Equipo 1] Rate limiting en APIs
└─ HITO: Certificación de seguridad

FASE 5: TESTING + OPTIMIZACIÓN (Semanas 21-24)
═══════════════════════════════════════════════

SEMANA 21: E2E TESTING
├─ [Equipo 5] Suite de E2E tests (Playwright/Cypress)
├─ [Equipo 5] Casos de uso críticos (crear doc → pagar → cierre caja)
├─ [Equipo 5] Testing en múltiples tenants simultáneos
└─ HITO: 95% de cobertura E2E

SEMANA 22: PERFORMANCE TUNING
├─ [Equipo 1] Profiling de queries lentas
├─ [Equipo 1] Caché de dados maestros (Redis)

├─ [Equipo 4] Optimización de bundle size (React)
├─ [Equipo 5] Load testing (1,000 usuarios concurrentes)
└─ HITO: < 200ms respuesta en 95% de requests

SEMANA 23: DOCUMENTACIÓN + TRAINING
├─ [Equipo 1] API documentation (Swagger/OpenAPI)
├─ [Todos] Technical documentation
├─ [Equipo 4] User manual (operarios + supervisores)
├─ [Equipo 5] Deployment guide
└─ HITO: Knowledge transfer completo

SEMANA 24: HARDENING + UAT
├─ [Equipo 5] Últimas correcciones
├─ [Equipo 5] Deploy a staging
├─ [Cliente] User Acceptance Testing
├─ [Todos] Bug fixes fase final
└─ HITO:  LANZAMIENTO EN PRODUCCIÓN

### 11.2 Hitos Críticos (Go/No-Go)

HITO 1 (Fin Semana 4): "Maestros Operacionales"
├─ ✓ CRUD de items, categorías, bodegas, clientes
├─ ✓ Búsquedas y filtros
├─ ✓ Validaciones básicas
└─ CRITERIO: Pueda navegar datos maestros sin errores

HITO 2 (Fin Semana 10): "Core Transaccional Completo"
├─ ✓ Crear documentos con múltiples líneas
├─ ✓ Aplicar descuentos e impuestos (DIAN)
├─ ✓ Agregar pagos en múltiples formas
├─ ✓ Auditoría registra todos los cambios
└─ CRITERIO: Transacción end-to-end funciona

HITO 3 (Fin Semana 16): "Ciclo de Caja Completo"
├─ ✓ Abrir → Vender → Arquear → Cerrar
├─ ✓ Sangrías generan documentos
├─ ✓ Cuadre de caja sin discrepancias
└─ CRITERIO: POS operacional en escenarios reales

HITO 4 (Fin Semana 20): "Seguridad + Reportes"
├─ ✓ RBAC granular funciona (rol restricts access)
├─ ✓ Data scopes filtran correctamente
├─ ✓ Reportes ejecutivos disponibles
├─ ✓ Auditoría de seguridad completa
└─ CRITERIO: Sistema listo para cumplimiento normativo

HITO 5 (Fin Semana 24): "PRODUCCIÓN"
├─ ✓ Todos tests pasan (unit + integration + E2E)
├─ ✓ Performance SLAs cumplidas (< 200ms)
├─ ✓ Load testing: 1,000 users concurrentes
├─ ✓ Documentación completa

├─ ✅ UAT aprobado
└─ CRITERIO: Sistema en vivo, operaciones normales

---

# 12. KPIs DE ÉXITO POR MÓDULO

## 12.1 KPIs Técnicos

DISPONIBILIDAD (Uptime):
├─ Target: 99.9% (máx 43 min downtime/mes)
├─ Medida: Monitoreo automático (Uptime Robot)
└─ Crítico para: Operaciones 24/5 (ajustable por cliente)

PERFORMANCE:
├─ API Response Time:
│ ├─ Target: P95 < 200ms, P99 < 500ms
│ ├─ Medida: Datadog / New Relic
│ └─ Test: Load test 1,000 usuarios
│
├─ Database Query Time:
│ ├─ Target: P95 < 100ms (queries a 1M registros)
│ ├─ Medida: PostgreSQL slow query log
│ └─ Optimización: Índices, JSONB queries
│
└─ Frontend Load Time:
├─ Target: FCP < 1.5s, LCP < 2.5s
├─ Medida: Google Lighthouse
└─ Bueno para: UX, abandonos reducidos

CODE QUALITY:
├─ Test Coverage: > 85% (unit + integration)
├─ OWASP Top 10: 0 vulnerabilidades críticas
├─ Linting: 100% compliance (SonarQube)
└─ Documentation: 100% métodos documentados

SECURITY:
├─ Penetration Testing: 0 fallos críticos
├─ Dependency Vulnerabilities: 0 high/critical
├─ Secrets Management: 100% rotados mensualmente
└─ Access Logs: Auditable 100% transacciones

## 12.2 KPIs de Negocio

ADOPCIÓN Y USO:
├─ Documentos por mes: (Baseline: 0 → Target: 50k en M6)
├─ Usuarios activos: (Baseline: 0 → Target: 100+ en M6)
├─ Sesiones de caja: (Baseline: 0 → Target: 1,000+/mes en M6)
└─ Retención: > 95% (usuarios siguen usando mes-a-mes)

EFICIENCIA OPERATIVA:
├─ Tiempo medio para crear documento: < 2 min (vs 5 min manual)
├─ Ciclo de caja cerrado: < 15 min (vs 45 min manual)

├─ Errores de caja: < 0.5% (vs 2-3% manual)
└─ Auditoría: 100% trazable (vs 20% manual)

SATISFACCIÓN:
├─ NPS (Net Promoter Score): > +30
├─ SLA cumplimiento: 99%
├─ Bug report resolution: < 24 horas críticos
└─ Feature requests: 80% implementados en roadmap

ROI:
├─ Reducción de horas manuales: > 40%
├─ Ahorro en errores/pérdidas: > $100k/año
├─ Velocidad de operación: 3x vs ERP tradicional
└─ Payback period: < 18 meses

---

# RESUMEN EJECUTIVO PARA HEAD OF PRODUCT

### Cambios en V2 vs V1

| Aspecto | V1 | V2 | Impacto |
|---|---|---|---|
| **Módulos** | 1 (Documentos) | 4 (+ Maestros, Seguridad, POS) | Solución integrada lista para producción |
| **Tablas BD** | ~20 | ~60 | Complejidad gestionable, escalable |
| **Equipos** | 2 (Backend, Frontend) | 5 (+ Seguridad, POS, QA) | Desarrollo paralelo, 6 meses |
| **Seguridad** | Roles básicos | RBAC granular tipo ERP | Cumplimiento normativo |
| **POS** | No existe | Ciclo completo + auditoría | Control de fraude + visibilidad financiera |
| **Auditoría** | Parcial | Centralizada + completa | Compliance total |
| **Timeline** | 18 semanas | 24 semanas | +1 mes por complejidad añadida |
| **Presupuesto** | $150k | ~$250k | +67% por funcionalidades críticas |

### Recomendación

✅ **APROBAR V2** como versión definitiva para Farutech.

- Justificación: Motor transaccional parametrizable + Control financiero (POS) + RBAC = **Solución diferenciadora vs ERPs tradicionales**
- ROI: Payback en 18 meses, 40% reducción de horas manuales
- Timeline: 24 semanas (6 meses) es realista y ejecutable

---

# ANEXOS

## A. Glosario de Términos

TENANT: Empresa/cliente segregada en la plataforma multi-tenant
DOCUMENTO: Transacción comercial (Factura, Recibo, Orden, etc)
SUBTYPE: Variante de documento (Factura A vs Factura B)
CONSECUTIVO: Número secuencial único por subtype
SANGRÍA/RETIRO: Extracción de efectivo de caja autorizada
ARQUEO: Conteo físico de dinero vs saldo del sistema
DISCREPANCIA: Diferencia entre dinero contado vs contabilizado
SCOPE: Restricción de datos que ve un usuario
RBAC: Role-Based Access Control (seguridad basada en roles)
DIAN: Dirección de Impuestos y Aduanas (Colombia)
IVA: Impuesto al Valor Agregado (19% en Colombia)
RETENCIONES: Retención en la fuente por impuestos

## B. Referencias Normativas

COLOMBIA 2025:
├─ Resolución DIAN 000012 (2024): Nuevas tarifas de retención IVA
├─ Decreto 0613 (2024): Requisitos facturación electrónica
├─ Norma técnica CONPE: Contenidos de factura
└─ Ley 1819 (2016): Impuestos nacionales

INTERNACIONALES:
├─ ISO 27001: Seguridad de la información
├─ OWASP Top 10: Seguridad web
└─ GDPR: Protección de datos (si aplica)

## C. Queries SQL Útiles

```
-- Traer todos los permisos de un usuario (incluyendo roles)
SELECT DISTINCT p.code, p.name
FROM v_user_permissions
WHERE user_id = '{user_id}'
ORDER BY p.code;

-- Documentos creados por usuario en rango de fecha
SELECT d.document_number, d.grand_total, d.status, d.created_at
FROM trx__documents d
WHERE d.created_by_user_id = '{user_id}'
```

```
AND d.created_at BETWEEN '{start_date}' AND '{end_date}'
ORDER BY d.created_at DESC;

-- Sesiones de caja con discrepancias
SELECT s.session_number, s.expected_balance, s.physical_count, s.variance
FROM scr__cashier_sessions s
WHERE s.tenant_id = '{tenant_id}'
AND s.variance != 0
AND s.closed_at >= NOW() - INTERVAL '30 days'
ORDER BY s.variance DESC;

-- Productos con stock bajo (por debajo del reorder_point)
SELECT i.code, i.name, ws.quantity_available, i.reorder_point
FROM mae__warehouse_stock ws
JOIN mae__items i ON ws.item_id = i.id
WHERE ws.tenant_id = '{tenant_id}'
AND ws.quantity_available < i.reorder_point
AND i.is_active = TRUE
ORDER BY ws.quantity_available ASC;

-- Auditoría: Quién cambió qué documento
SELECT u.email, a.operation, a.old_values, a.new_values, a.timestamp
FROM audit__transaction_log a
JOIN scr__users u ON a.performed_by_user_id = u.id
WHERE a.entity_type = 'trx__documents'
AND a.entity_id = '{document_id}'
ORDER BY a.timestamp DESC;
```

---

**DOCUMENTO FINALIZADOR**
Fecha: Enero 25, 2026
Versión: 2.0 - EXTENSIÓN CRÍTICA
Estado: ✅ LISTO PARA DESARROLLO

**Siguientes pasos:**

1. ✅ Aprobación por Head of Product
2. ✅ Kick-off con equipos
3. ✅ Asignación de sprints (Sprint Planning)
4. ✅ Setup de infraestructura (Semana 1)
5. ✅ Inicio desarrollo paralelo (Semana 1)