Diseño Comentado de Base de Datos para Punto de Venta

1. Tablas de Autenticación y Usuarios

Schema::create('users', function (Blueprint $table) {

$table->id(); // ID único del usuario

$table->string('name'); // Nombre completo del usuario

$table->string('email')->unique(); // Email único para login

$table->timestamp('email\_verified\_at')->nullable(); // Fecha de verificación de email

$table->string('password'); // Contraseña encriptada

$table->string('phone')->nullable(); // Teléfono de contacto

$table->string('address')->nullable(); // Dirección física

$table->boolean('is\_active')->default(true); // Si el usuario está activo/inactivo

$table->rememberToken(); // Token para "recordar" sesión

$table->timestamps(); // created\_at y updated\_at

$table->softDeletes(); // deleted\_at para eliminación suave

});

2. Roles y Permisos (spatie/laravel-permission)

Schema::create('roles', function (Blueprint $table) {

$table->bigIncrements('id'); // ID del rol

$table->string('name')->unique(); // Nombre único (ej: 'admin', 'cajero')

$table->string('guard\_name')->default('web'); // Tipo de autenticación

$table->string('description')->nullable(); // Descripción del rol

$table->timestamps();

});

Schema::create('permissions', function (Blueprint $table) {

$table->bigIncrements('id'); // ID del permiso

$table->string('name')->unique(); // Nombre único (ej: 'edit.orders')

$table->string('guard\_name')->default('web'); // Tipo de autenticación

$table->string('description')->nullable(); // Descripción del permiso

$table->timestamps();

});

Schema::create('role\_has\_permissions', function (Blueprint $table) {

// Tabla pivote que relaciona permisos con roles

$table->unsignedBigInteger('permission\_id'); // ID del permiso

$table->unsignedBigInteger('role\_id'); // ID del rol

// Llaves foráneas

$table->foreign('permission\_id')->references('id')->on('permissions')->onDelete('cascade');

$table->foreign('role\_id')->references('id')->on('roles')->onDelete('cascade');

$table->primary(['permission\_id', 'role\_id']); // Llave primaria compuesta

});

3. Estructura de Sucursales

Schema::create('branches', function (Blueprint $table) {

$table->id(); // ID único de sucursal

$table->string('name'); // Nombre de la sucursal (ej: "Sucursal Centro")

$table->string('code')->unique(); // Código corto único (ej: "CENTRO")

$table->string('address'); // Dirección física completa

$table->string('phone'); // Teléfono de contacto

$table->string('email')->nullable(); // Email de contacto

$table->time('opening\_time'); // Hora de apertura (ej: 08:00:00)

$table->time('closing\_time'); // Hora de cierre (ej: 22:00:00)

$table->boolean('is\_active')->default(true); // Sucursal activa/inactiva

$table->text('description')->nullable(); // Notas adicionales

$table->timestamps();

$table->softDeletes(); // Para desactivar sucursales sin eliminarlas

});

Schema::create('branch\_user', function (Blueprint $table) {

// Tabla pivote para relación muchos-a-muchos entre usuarios y sucursales

$table->foreignId('user\_id')->constrained()->onDelete('cascade'); // ID usuario

$table->foreignId('branch\_id')->constrained()->onDelete('cascade'); // ID sucursal

$table->boolean('is\_active')->default(true); // Si la asignación está activa

$table->date('assigned\_at')->default(now()); // Fecha de asignación

$table->date('unassigned\_at')->nullable(); // Fecha de desasignación

$table->text('assignment\_notes')->nullable(); // Razón del cambio

$table->primary(['user\_id', 'branch\_id']); // Llave primaria compuesta

$table->timestamps();

});

4. Gestión de Cajas Registradoras

Schema::create('cash\_registers', function (Blueprint $table) {

$table->id(); // ID único de caja

$table->foreignId('branch\_id')->constrained()->onDelete('cascade'); // Sucursal

$table->string('name'); // Nombre descriptivo (ej: "Caja Principal")

$table->string('code')->unique(); // Código único (ej: "CAJA1-CENTRO")

$table->decimal('initial\_balance', 12, 2)->default(0); // Saldo inicial al abrir

$table->decimal('current\_balance', 12, 2)->default(0); // Saldo actual en caja

$table->boolean('is\_open')->default(false); // Si la caja está abierta/cerrada

$table->timestamp('last\_opened\_at')->nullable(); // Última apertura

$table->timestamp('last\_closed\_at')->nullable(); // Último cierre

$table->timestamps();

});

Schema::create('cash\_register\_movements', function (Blueprint $table) {

$table->id(); // ID único de movimiento

$table->foreignId('cash\_register\_id')->constrained()->onDelete('cascade'); // Caja relacionada

$table->foreignId('user\_id')->constrained()->onDelete('cascade'); // Usuario que realizó

$table->enum('type', [

'open', // Apertura de caja

'close', // Cierre de caja

'income', // Ingreso manual (ej: ajuste positivo)

'expense', // Egreso manual (ej: pago proveedor)

'sale', // Venta normal

'sale\_cancel',// Cancelación de venta

'adjustment', // Ajuste de caja

'transfer\_in', // Transferencia desde otra caja

'transfer\_out' // Transferencia a otra caja

]);

$table->decimal('amount', 12, 2); // Monto del movimiento

$table->string('reference')->nullable(); // Número de factura/ref externa

$table->text('description')->nullable(); // Detalle del movimiento

$table->json('details')->nullable(); // Datos adicionales estructurados

$table->timestamps();

});

5. Catálogo de Productos y Variantes

Schema::create('categories', function (Blueprint $table) {

$table->id(); // ID categoría

$table->string('name'); // Nombre (ej: "Hamburguesas")

$table->string('slug')->unique(); // URL amigable (ej: "hamburguesas")

$table->text('description')->nullable(); // Descripción

$table->string('image')->nullable(); // Ruta de imagen

$table->integer('order')->default(0); // Orden de visualización

$table->boolean('is\_active')->default(true); // Activa/inactiva

$table->timestamps();

});

Schema::create('products', function (Blueprint $table) {

$table->id(); // ID producto

$table->foreignId('category\_id')->constrained()->onDelete('cascade'); // Categoría

$table->string('name'); // Nombre (ej: "Hamburguesa Clásica")

$table->string('slug')->unique(); // URL amigable

$table->text('description')->nullable(); // Descripción/ingredientes

$table->string('image')->nullable(); // Ruta de imagen

$table->enum('product\_type', [

'food', // Alimentos

'beverage', // Bebidas

'combo', // Combos

'other' // Otros

])->default('food');

$table->boolean('has\_variants')->default(false); // Si tiene variantes (tamaños, extras)

$table->boolean('is\_combo')->default(false); // Si es un combo de productos

$table->boolean('track\_stock')->default(false); // Si se controla inventario

$table->boolean('is\_active')->default(true); // Producto activo/inactivo

$table->timestamps();

$table->softDeletes(); // Para desactivar sin eliminar

});

Schema::create('product\_branch', function (Blueprint $table) {

// Precios y stock específicos por sucursal

$table->foreignId('product\_id')->constrained()->onDelete('cascade'); // ID producto

$table->foreignId('branch\_id')->constrained()->onDelete('cascade'); // ID sucursal

$table->decimal('price', 12, 2); // Precio en ESTA sucursal

$table->integer('stock')->default(0)->nullable(); // Cantidad en stock

$table->integer('min\_stock')->default(0)->nullable(); // Mínimo antes de alerta

$table->boolean('is\_available')->default(true); // Disponible en esta sucursal

$table->primary(['product\_id', 'branch\_id']); // Llave compuesta

$table->timestamps();

});

Schema::create('variants', function (Blueprint $table) {

$table->id(); // ID variante

$table->string('name'); // Nombre (ej: "Porción grande")

$table->string('type'); // Tipo (ej: "size", "extra", "addition")

$table->boolean('is\_active')->default(true); // Variante activa/inactiva

$table->timestamps();

});

Schema::create('product\_variant', function (Blueprint $table) {

// Relación productos con sus variantes y precios

$table->foreignId('product\_id')->constrained()->onDelete('cascade'); // ID producto

$table->foreignId('variant\_id')->constrained()->onDelete('cascade'); // ID variante

$table->decimal('price', 12, 2); // Precio adicional por esta variante

$table->integer('order')->default(0); // Orden de visualización

$table->primary(['product\_id', 'variant\_id']); // Llave compuesta

$table->timestamps();

});

6. Gestión de Inventario y Compras

Schema::create('suppliers', function (Blueprint $table) {

$table->id(); // ID proveedor

$table->string('name'); // Nombre o razón social

$table->string('contact\_person'); // Persona de contacto

$table->string('phone'); // Teléfono

$table->string('email')->nullable(); // Email

$table->text('address')->nullable(); // Dirección

$table->timestamps();

});

Schema::create('purchases', function (Blueprint $table) {

$table->id(); // ID compra

$table->foreignId('branch\_id')->constrained(); // Sucursal que compra

$table->foreignId('supplier\_id')->nullable()->constrained(); // Proveedor

$table->foreignId('user\_id')->constrained(); // Usuario que registra

$table->string('invoice\_number')->nullable(); // Número de factura

$table->date('purchase\_date'); // Fecha de compra

$table->decimal('total', 12, 2); // Total de la compra

$table->text('notes')->nullable(); // Observaciones

$table->enum('status', [

'pending', // Pendiente de recepción

'received', // Recibido completo

'cancelled' // Cancelado

])->default('pending');

$table->timestamps();

});

Schema::create('inventory\_movements', function (Blueprint $table) {

$table->id(); // ID movimiento

$table->foreignId('branch\_id')->constrained(); // Sucursal

$table->foreignId('product\_id')->constrained(); // Producto

$table->foreignId('variant\_id')->nullable()->constrained(); // Variante (si aplica)

$table->enum('movement\_type', [

'purchase', // Compra a proveedor

'sale', // Venta a cliente

'adjustment', // Ajuste manual

'waste', // Pérdida/daño

'transfer\_in', // Transferencia desde otra sucursal

'transfer\_out' // Transferencia a otra sucursal

]);

$table->integer('quantity'); // Cantidad (positivo/negativo según tipo)

$table->decimal('unit\_cost', 12, 2)->nullable(); // Costo unitario

$table->foreignId('user\_id')->constrained(); // Usuario que registra

$table->morphs('sourceable'); // Relación polimórfica (compra, venta, etc)

$table->text('notes')->nullable(); // Observaciones

$table->timestamps();

});

7. Ventas y Pedidos

Schema::create('orders', function (Blueprint $table) {

$table->id(); // ID pedido

$table->foreignId('branch\_id')->constrained()->onDelete('cascade'); // Sucursal

$table->foreignId('user\_id')->constrained()->onDelete('cascade'); // Usuario que registra

$table->foreignId('cash\_register\_id')->nullable()->constrained()->onDelete('set null'); // Caja

$table->foreignId('customer\_id')->nullable()->constrained()->onDelete('set null'); // Cliente

$table->string('order\_number')->unique(); // Número único (ej: "ORD-00001")

$table->enum('status', [

'pending', // Pendiente

'preparing', // En preparación

'ready', // Listo para servir

'delivered', // Entregado

'completed', // Completado (pagado)

'cancelled' // Cancelado

])->default('pending');

$table->enum('type', [

'dine\_in', // Consumo en local

'takeaway', // Para llevar

'delivery' // Domicilio

])->default('dine\_in');

$table->string('table\_number')->nullable(); // Número de mesa (para dine-in)

$table->decimal('subtotal', 12, 2); // Subtotal antes de impuestos/descuentos

$table->decimal('tax\_amount', 12, 2)->default(0); // Monto de impuestos

$table->decimal('discount\_amount', 12, 2)->default(0); // Monto de descuentos

$table->decimal('total', 12, 2); // Total a pagar

$table->text('notes')->nullable(); // Instrucciones especiales

$table->timestamp('completed\_at')->nullable(); // Fecha de finalización

$table->timestamps();

$table->softDeletes(); // Para cancelaciones

});

Schema::create('order\_items', function (Blueprint $table) {

$table->id(); // ID ítem

$table->foreignId('order\_id')->constrained()->onDelete('cascade'); // Pedido

$table->foreignId('product\_id')->constrained()->onDelete('cascade'); // Producto

$table->string('product\_name'); // Nombre capturado al momento (para histórico)

$table->decimal('price', 12, 2); // Precio unitario

$table->integer('quantity'); // Cantidad

$table->decimal('subtotal', 12, 2); // Precio x cantidad

$table->text('notes')->nullable(); // Especificaciones (ej: "sin cebolla")

$table->timestamps();

});

Schema::create('payments', function (Blueprint $table) {

$table->id(); // ID pago

$table->foreignId('order\_id')->constrained()->onDelete('cascade'); // Pedido

$table->enum('method', [

'cash', // Efectivo

'credit\_card', // Tarjeta crédito

'debit\_card', // Tarjeta débito

'transfer', // Transferencia

'mobile\_payment', // Pago móvil

'other' // Otro método

]);

$table->decimal('amount', 12, 2); // Monto pagado

$table->string('reference')->nullable(); // Referencia/transacción

$table->string('card\_last\_four')->nullable(); // Últimos 4 dígitos tarjeta

$table->string('card\_brand')->nullable(); // Visa, Mastercard, etc.

$table->string('payment\_provider')->nullable(); // Procesador de pago

$table->enum('status', [

'pending', // Pendiente

'completed', // Completado

'failed', // Fallido

'refunded' // Reembolsado

])->default('completed');

$table->text('notes')->nullable(); // Observaciones

$table->timestamps();

});

8. Tablas Adicionales

Schema::create('audit\_logs', function (Blueprint $table) {

$table->id(); // ID registro

$table->string('action'); // Acción (create, update, delete)

$table->string('model\_type'); // Modelo afectado (ej: "App\Models\Order")

$table->unsignedBigInteger('model\_id')->nullable(); // ID del modelo

$table->json('old\_values')->nullable(); // Valores anteriores (para updates)

$table->json('new\_values')->nullable(); // Valores nuevos

$table->text('description')->nullable(); // Descripción legible

$table->foreignId('user\_id')->nullable()->constrained()->onDelete('set null'); // Usuario

$table->string('ip\_address', 45)->nullable(); // IP desde donde se realizó

$table->string('user\_agent')->nullable(); // Navegador/dispositivo

$table->timestamps();

});

Este diseño comentado proporciona una comprensión clara de cada tabla y campo en el sistema. ¿Necesitas que profundice en alguna relación específica o en cómo implementar alguna funcionalidad en particular?