

Registro de venta con APIView & Transacciones ACID

¿QUÉ CONSTRUIMOS?

Registramos una venta comercial en una **única transacción atómica (ACID)** sobre la base de datos. Se validan cliente y productos en estado activo, se bloquean pesimistamente las filas con `select_for_update()`, se congela el `unit_price` del catálogo, se calculan subtotal, descuentos y total, y se descuenta el stock en el mismo bloque.

ARQUITECTURA DE CAPAS

- **SaleQuerySet.register**: Encapsula la invariante de negocio y las reglas de dominio.
- **transaction.atomic()**: Delimita el bloque de transacción de BD.
- **select_for_update()**: Bloquea pesimistamente cliente y productos ordenados por ID para evitar interbloqueos (deadlocks).
- **SaleRegisterSerializer**: Mapea la entrada JSON camelCase e invoca el manager.
- **SaleRegisterAPIView**: Endpoint HTTP dedicado en acceso OPEN (Fase I).

FLUJO TRANSACCIONAL (ACID)

1. **HTTP POST /api/sales/** → Entrada de payload JSON.
2. **DRF Validation** → Validación de estructura e ítems.
3. **Lock & Verify** → `SELECT FOR UPDATE` de cliente y productos active.
4. **Snapshot Price** → `Copia unit_price = product.price`.
5. **Save Sale & Lines** → Totales calculados, status ACTIVE.
6. **Stock Update** → `Product.stock -= quantity`.
7. **COMMIT or ROLLBACK** → Si todo OK → `COMMIT`; si falla → `ROLLBACK`.

CONCEPTOS CLAVE & DECISIONES DE DISEÑO

- **¿Por qué APIView en lugar de ModelViewSet?** Porque un alta transaccional multi-recurso (Venta + Detalle + Inventario) es una operación específica. ModelViewSet descompondría el alta en llamadas no atómicas.
- **Inmutabilidad de Precio Histórico**: El valor `unit_price` se congela en ProductSale al vender. Cambios posteriores en `Product.price` no alteran el histórico.
- **Prevención de Deadlocks**: Los productos se bloquean con `.order_by('id')` para asegurar un orden estricto de candados en el motor de base de datos.

VERIFICACIÓN, EVIDENCIAS Y GATE

Criterio de Aceptación	Verificación	Evidencia	Resultado
AC-11-01: Cálculo de totales y estado active	POST 2×10.00 + 1×4.50 – 1.50 = 23.00	EVI-11-01	PASS
AC-11-02: Preservación de precio histórico	Cambio de Product.price a 99.00 no afecta la venta	EVI-11-01	PASS
AC-11-03: Rechazo por stock insuficiente	POST con cantidad > stock responde 400	EVI-11-01	PASS
AC-11-04: Rollback ante fallo de guardado stock	Simulación de error en save() revierte Venta y Detalle	EVI-11-01	PASS
AC-11-05: Rechazo de cliente inactivo	POST con client.status=INACTIVE responde 400	EVI-11-01	PASS