Documentación técnica del dominio de ventas,
motor de promociones y ledger distribuido

**Versión:** v202604070300

**Base:** core.md (202604070108) · promociones.md (202604070300)

**Modelos JSON:** jsonmodel_optionB_consolidado_promociones_v4

**Implementación objetivo:** Java 8 · POJOs · sin Lombok

# Tabla de contenidos

# 1. Introducción y principios del modelo

Este documento es la referencia técnica completa del dominio POS / Ticket / Ledger. Consolida **core.md** y **promociones.md** en un único documento integrado, incluyendo diagramas, modelos de datos, ejemplos JSON y convenciones de implementación Java 8.

## 1.1 Jerarquía de documentos

| Prioridad | Documento | Descripción |
|---|---|---|
| 1 | core.md (202604070108) | Referencia central del modelo de dominio POS/Ticket/Ledger |
| 2 | promociones.md (202604070300) | Complemento específico del motor de promociones |
| 3 | jsonmodel_optionB_v4 | Ejemplos estructurales JSON del modelo corregido |
| 4 | Documentos históricos | Referencias previas (menor prioridad ante contradicción) |

## 1.2 Principios de diseño

- **Detallado:** Cada monto relevante debe poder explicarse línea por línea.
- **Trazable:** Toda transformación económica debe poder rastrearse en movimientos[].
- **Reconciliable:** La suma algebraica del ledger debe cerrar en 0 al finalizar el ticket.
- **Auditable:** El modelo debe soportar auditoría fiscal y contable completa.
- **Extensible:** Nuevos tipos de promoción, pago o impuesto se incorporan sin romper el núcleo.
- **Conservador:** Ante dudas de modelado, se elige la solución más explicable y segura.

✓ Regla fundamental: no introducir reglas no confirmadas. Cuando exista duda de modelado, preferir la solución más conservadora y explicable en términos de cálculo y auditoría.

# 2. Agregado raíz: Ticket

El agregado raíz es **Ticket**. Representa el estado completo de una operación POS y es la unidad principal para cálculo, validación, testing, auditoría y reconciliación.

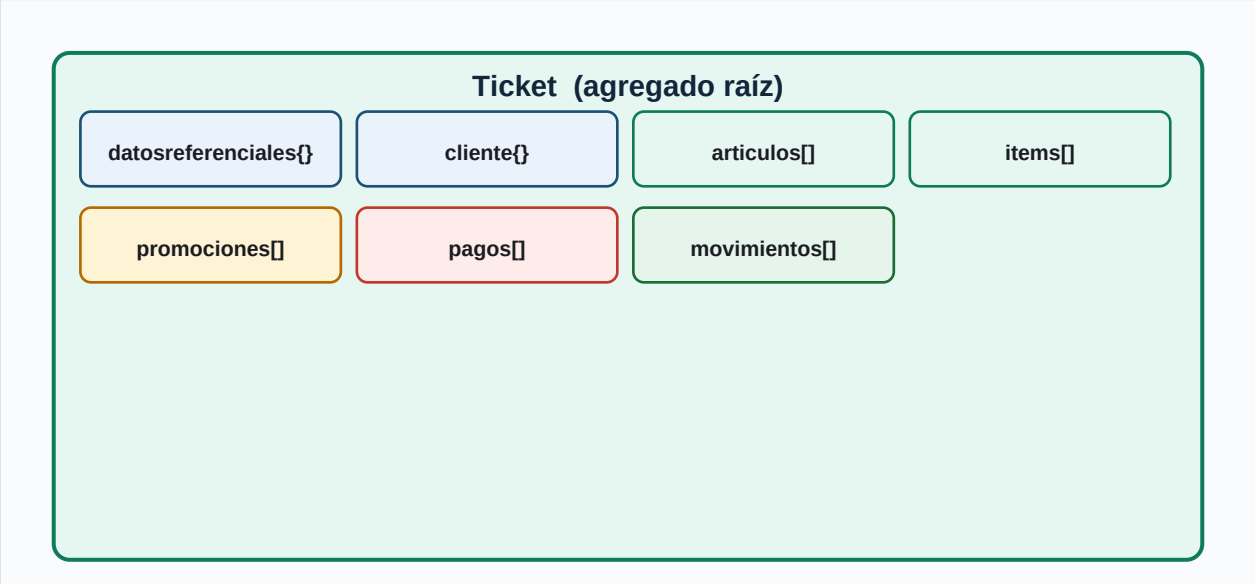


*Figura 1 — Estructura del agregado raíz Ticket*

## 2.1 Arrays del ticket

| Array | Tipo | Descripción |
|---|---|---|
| datosreferenciales | Objeto | Metadatos generales: nroTicket, comercio, sucursal, fechaHora, totales |
| cliente{} | Objeto | Identificación, tipo de cliente, información tributaria |
| articulos[] | Catálogo | Artículos reutilizables. Evita duplicar datos de producto |
| items[] | Registro | Captura comercial original. Un registro por ingreso operativo |
| promociones[] | Registro | Registros maestros de promociones aplicadas en el ticket |
| pagos[] | Registro | Registros maestros de pagos ingresados en el ticket |
| movimientos[] | Ledger | Detalle económico distribuido. Fuente de verdad fiscal/contable |

## 2.2 Estructura JSON base

**JSON — TICKET (ESTRUCTURA BASE)**


```
{
  "ticket": {
    "datosreferenciales": { "nroTicket": 12213, "total": 2620.0, "saldo": 0.0, "vuelto": 380.0 },
    "cliente": { "id": 1, "tipodecliente": { "id": "CONSUMIDOR_FINAL" } },
    "articulos": [],
    "items": [],
    "promociones": [],
    "pagos": [],
    "movimientos": []
  }
}
```

# 3. Granos del modelo

El modelo trabaja con tres granos funcionales claramente separados. Esta separación es obligatoria y no debe perderse bajo ninguna circunstancia.

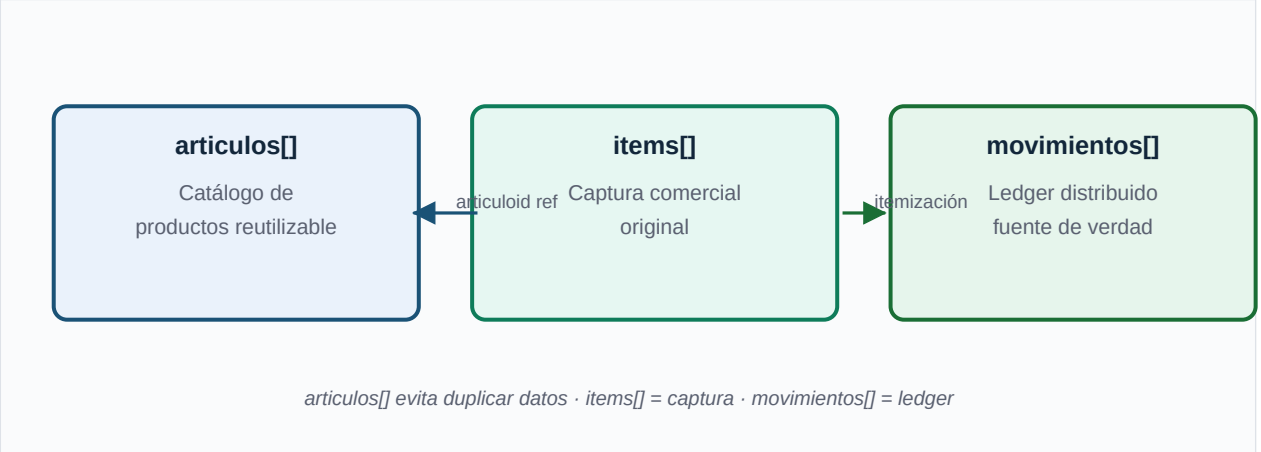


*Figura 2 — Los tres granos: articulos[], items[] y movimientos[]*

## 3.1 articulos[]

Catálogo de artículos reutilizables dentro del ticket. Su objetivo es evitar duplicar datos de producto cuando varios **items[]** referencian el mismo artículo. Si al ingresar un EAN el artículo ya existe, se reutiliza su **id**.

## 3.2 items[]

Representa la captura comercial original. Cada ingreso de ítem tiene un **id** incremental, referencia un **articuloid** e informa **unidades**. Conserva el dato tal como fue ingresado operativamente.

> ✓ Regla de itemización: si un item tiene unidades enteras, deben generarse tantos movimientos VENTA_ITEM como unidades registradas en movimientos[]. items[] es captura comercial — movimientos[] es impacto económico distribuido.

## 3.3 movimientos[]

Representa el ledger distribuido y detallado. Es el core de explicación económica del ticket. Contiene ventas itemizadas, promociones distribuidas, pagos distribuidos, excedentes y vueltos. **No reemplaza a items[], pagos[] ni promociones[] — los complementa.**

| Campo | Tipo | Descripción |
|---|---|---|
| id | integer | Identificador incremental del movimiento |
| concepto | enum | VENTA_ITEM · PROMOCION · PAGO |
| origenid | integer | Referencia al registro maestro origen (item, promo o pago) |
| movimientoid | integer/null | Referencia al VENTA_ITEM sobre el que aplica. null para excedentes y vuelto |

| | | |
|---|---|---|
| nucleoimpositivo[] | array | Estructura de importes e impuestos del movimiento |

# 4. Objetos del dominio

## 4.1 NucleoImpositivo

Es la estructura central de importes e impuestos. Se usa en artículos, movimientos y datosreferenciales. No asumir que todo impuesto es porcentual.

**JSON — NUCLEOIMPOSITIVO (EJEMPLO IVA 21%)**

```
{
  "nucleoimpositivo": [
    { "impuesto": { "id": "NETO_IVA_21" },          "monto": 1000.0 },
    { "impuesto": { "id": "IVA_21" },                "monto":  210.0 },
    { "impuesto": { "id": "IMPUESTOINTERNO_IVA_21" },"monto":  100.0 }
  ]
}
```

## 4.2 Articulo

| Campo | Tipo | Descripción |
|---|---|---|
| ean | string | Código de barras EAN |
| plu | string | Código PLU |
| descripcion | string | Nombre del artículo |
| pesable | boolean | Indica si se vende por peso |
| preciolista | number | Precio de lista base para cálculo de promociones |
| rubro / depto / marca | string | Atributos comerciales para filtros de promoción |
| codigoclasificacion | integer | Código de clasificación genérico |
| proveedor | string | Proveedor del artículo |
| nucleoimpositivo[] | array | Composición impositiva base del artículo |

## 4.3 Item

| Campo | Tipo | Descripción |
|---|---|---|
| id | integer | Identificador incremental del ingreso |
| articuloid | integer | Referencia al artículo en articulos[] |
| unidades | number | Cantidad ingresada (admite decimales para pesables) |

## 4.4 TipoDePago — Maestro de medios de pago

Catálogo de configuración externo al ticket. Define el comportamiento funcional de cada medio de pago. La API de Promociones lo consulta internamente para resolver la lógica de vuelto.

| Campo | Tipo | Descripción |
|---|---|---|
| id | integer | ID único del medio de pago (engloba sub-medio y cuotas) |
| descripcion | string | Nombre descriptivo |
| davuelto | boolean | true: vuelto en el mismo medio · false: ver vueltomediodepago |
| vueltomediodepago | integer/null | ID del medio alternativo para el vuelto. null = operación denegada |

| Campo | EFECTIVO | CHEQUE | TARJETA_DEBITO |
|---|---|---|---|
| id | 1 | 2 | 3 |
| davuelto | true | false | false |
| vueltomediodepago | null | 1 (EFECTIVO) | null |
| Caso vuelto | A — mismo medio | C — medio alternativo | B — denegado |

## 4.5 Pago

Registro maestro de pago dentro del ticket. El monto puede ser **positivo o negativo**. Un pago negativo es válido cuando el medio de pago participa como vuelto.

> ! Un Pago negativo representa egreso o compensación (vuelto). No agregar campos adicionales como montoingresado o montoaplicado — la versión base del modelo usa solo monto.

# 5. Modelo de promociones

## 5.1 Separación en tres capas

El modelo de promociones distingue tres capas obligatorias que no deben mezclarse.

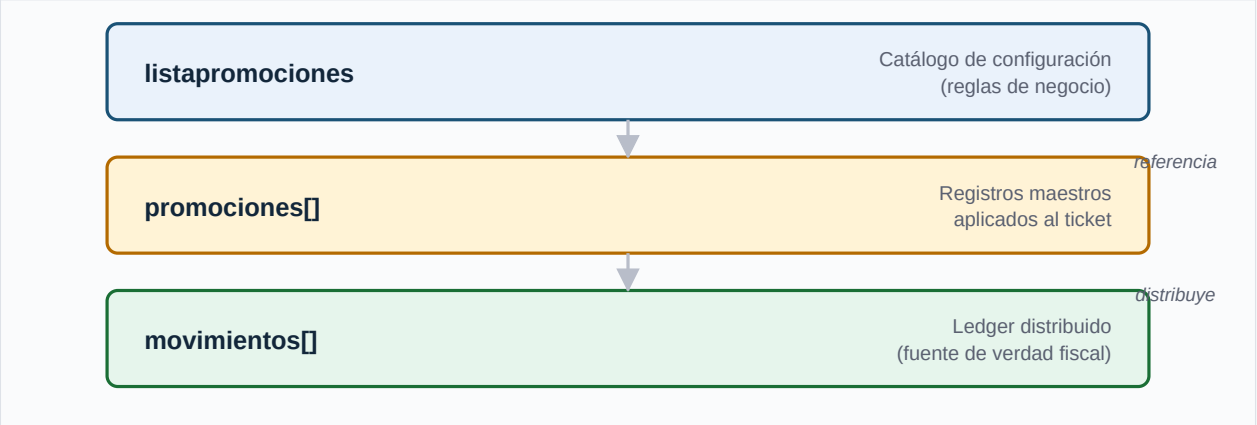


*Figura 3 — Las tres capas del modelo de promociones*

| Capa | Estructura | Tipo | Descripción |
|---|---|---|---|
| Definición | listapromociones | Catálogo / configuración | Regla configurable de negocio. No es un registro operativo del ticket |
| Aplicada | promociones[] | Registro operativo del ticket | Resultado maestro de la aplicación. Referencia a la definición. |
| Ledger | movimientos[] | Ledger distribuido | Impacto económico distribuido. Fuente de verdad fiscal. |

## 5.2 Catálogos auxiliares

| Catálogo | Valores | Descripción |
|---|---|---|
| promocionalcance | ITEM · PAGO | Etapa funcional en que aplica la promoción |
| promocionbeneficio | PORCENTAJE · MONTO · NUEVOPRECIOITEM | Cómo interpretar el campo valor |
| promocionmetodo | CANTIDAD · COMBO | Lógica base de resolución |
| promociondecision | NOACUMULATIVA · ACUMULATIVA | Base de cálculo del precio (lista o acumulado) |
| promocionestado | POSIBLE · APLICADA · NOAPLICA · ANULADA | Estado de una promoción aplicada en el ticket |
| promocionlistatype | INCLUSION · EXCLUSION | Tipo de entrada en la lista de la promoción |
| promocionlistanumber | LISTA1 · LISTA2 | Agrupa listas lógicas distintas dentro de una promo |

## 5.3 promociontipoelemento

| Valor | Descripción |
|---|---|
| EAN | Código de barras EAN del producto |
| PLU | Código PLU del producto |

| | |
|---|---|
| DEPTO | Departamento |
| RUBRO | Rubro |
| CODIGOCLASIFICACION | Código de clasificación genérico |
| PROVEEDOR | Proveedor |
| MARCA | Marca |
| MEDIODEPAGO | ID de medio de pago (engloba sub-medio y cuotas) |
| SUCURSAL | Sucursal |
| TICKET | La condición aplica al conjunto completo de ítems del ticket |
| CANTIDAD_MAX_PROMOS | Restricción operativa: cantidad máxima de aplicaciones de la promoción en el ticket. Siempre con valo |

## 5.4 Definición de promoción — listapromociones

Cada entrada en **listapromociones** define la regla completa de una promoción. Es un catálogo de configuración externo al ticket.

| Campo | Tipo | Obligatorio | Descripción |
|---|---|---|---|
| id | integer | Sí | Identificador único |
| descripcion | string | Sí | Nombre descriptivo |
| promocionalcance | objeto {id} | Sí | ITEM o PAGO |
| promocionbeneficio | objeto {id} | Sí | Tipo de beneficio |
| valor | number | Sí | Valor numérico del beneficio. Su semántica depende de promocionben |
| promocionmetodo | objeto {id} | Sí | Método de cómputo |
| promociondecision | objeto {id} | Sí | ACUMULATIVA o NOACUMULATIVA — por promoción |
| vigencia | objeto | Sí | fechadesde, fechahasta, diassemana[], horadesde, horahasta |
| lista[] | array | Sí | Inclusiones, exclusiones y restricciones |

**JSON — LISTAPROMOCIONES (PROMO_2X1_ARROZ)**


```
{
  "listapromociones": [
    {
      "id": 1,
      "descripcion": "PROMO_2X1_ARROZ",
      "promocionalcance": { "id": "ITEM" },
      "promocionbeneficio": { "id": "PORCENTAJE" },
      "valor": 50.0,
      "promocionmetodo": { "id": "CANTIDAD" },
      "promociondecision": { "id": "NOACUMULATIVA" },
      "vigencia": {
        "fechadesde": "2026-03-01", "fechahasta": "2026-03-31",
        "diassemana": ["MIERCOLES"], "horadesde": "10:00", "horahasta": "11:00"
      },
      "lista": [
        { "listaindex": 1, "promocionlistatype": { "id": "INCLUSION" },
          "promocionlistanumber": { "id": "LISTA1" },
          "promociontipoelemento": { "id": "EAN" },
          "valordeelemento": "7791234567890", "cantidad": 2.0 },
        { "listaindex": 2, "promocionlistatype": { "id": "INCLUSION" },
          "promocionlistanumber": { "id": "LISTA1" },
          "promociontipoelemento": { "id": "CANTIDAD_MAX_PROMOS" },
          "valordeelemento": "1" }
      ]
    }
  ]
}
```


## 5.5 Regla CANTIDAD_MAX_PROMOS

Limita cuántas veces puede resolverse una promoción dentro del ticket, independientemente de cuántas unidades válidas existan.

→ Ejemplo: PROMO_2X1_ARROZ con CANTIDAD_MAX_PROMOS=1 y 3 unidades en el ticket. La promoción se resuelve una sola vez → 2 unidades alcanzan el beneficio. La 3ª unidad queda fuera. Agregar más unidades no habilita nuevas aplicaciones.

## 5.6 Promoción aplicada — promociones[]

| Campo | Tipo | Req. | Descripción |
|---|---|---|---|
| id | integer | Sí | Identificador del registro de promoción aplicada |
| promocionid | integer | Sí | Referencia a la definición en listapromociones |
| descripcion | string | Sí | Nombre heredado de la definición |
| tipoPromo | string | Sí | ITEM o PAGO (naming vigente en JSONs del modelo) |
| monto | number | Sí | Monto total del beneficio (negativo para descuentos) |
| promocionestado | objeto {id} | No | Estado de la promoción aplicada |

| | | | |
|---|---|---|---|
| elementos[] | array | No | Trazabilidad: movimientoid, articuloid, unidadesimpactadas, monto |

✓ Regla: la suma de elementos[].monto debe coincidir con promociones[].monto. El detalle fiscal definitivo sigue en movimientos[] — no usar promociones[].monto como sustituto del ledger.

# 6. Motor de promociones — Modos de cómputo

El motor admite dos modos de cómputo. El orden general sigue el flujo de **core.md sección 9**: promociones de ítem (pasos 5-6) y promociones de pago (pasos 9-11).

## 6.1 MODO ITEM

Se ejecuta **una sola vez** por ticket. Cada vez que se invoca, resetea a cero todas las promociones previamente calculadas. Evalúa únicamente promociones con **promocionalcance=ITEM**. No calcula ni devuelve promociones de medio de pago.

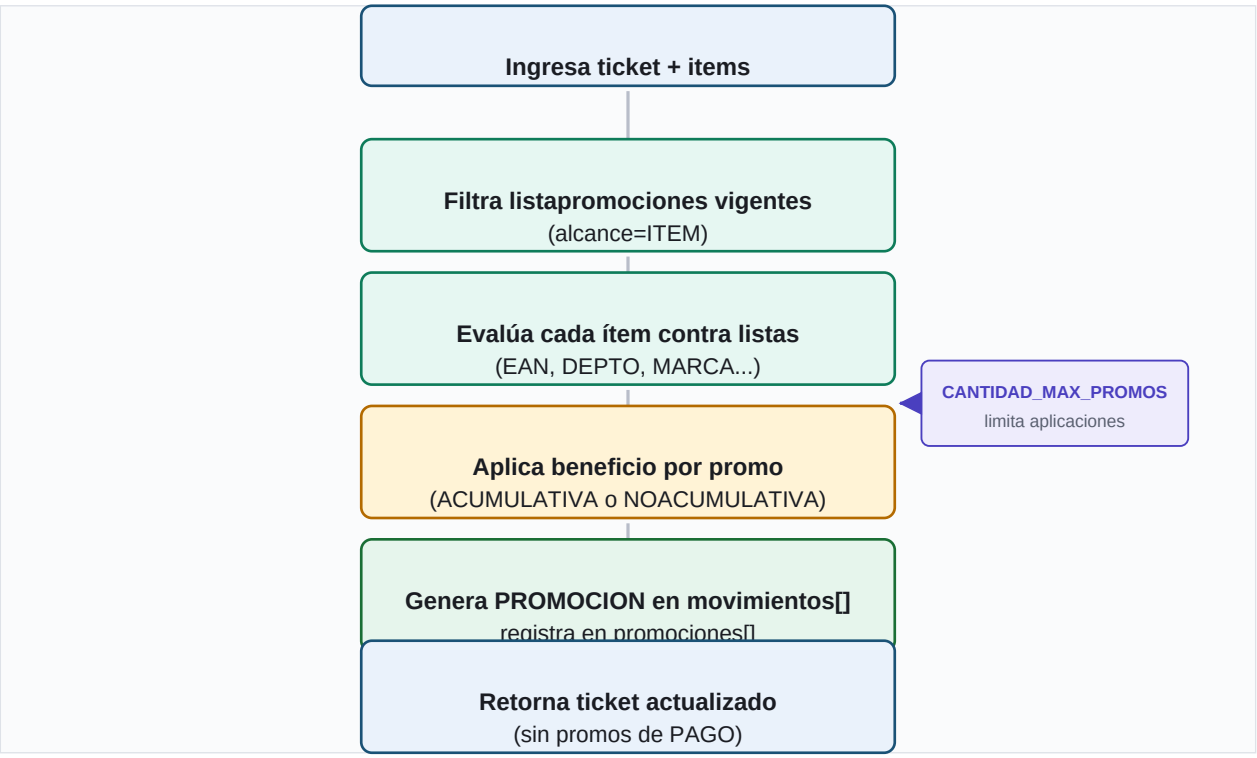


*Figura 4 — Flujo MODO ITEM*

## 6.2 MODO PAGO

Puede invocarse **múltiples veces** por ticket. Evalúa promociones con **promocionalcance=PAGO**. Los pagos se evalúan contra el saldo real del ticket luego de promociones e impuestos. Opera en dos sub-etapas.

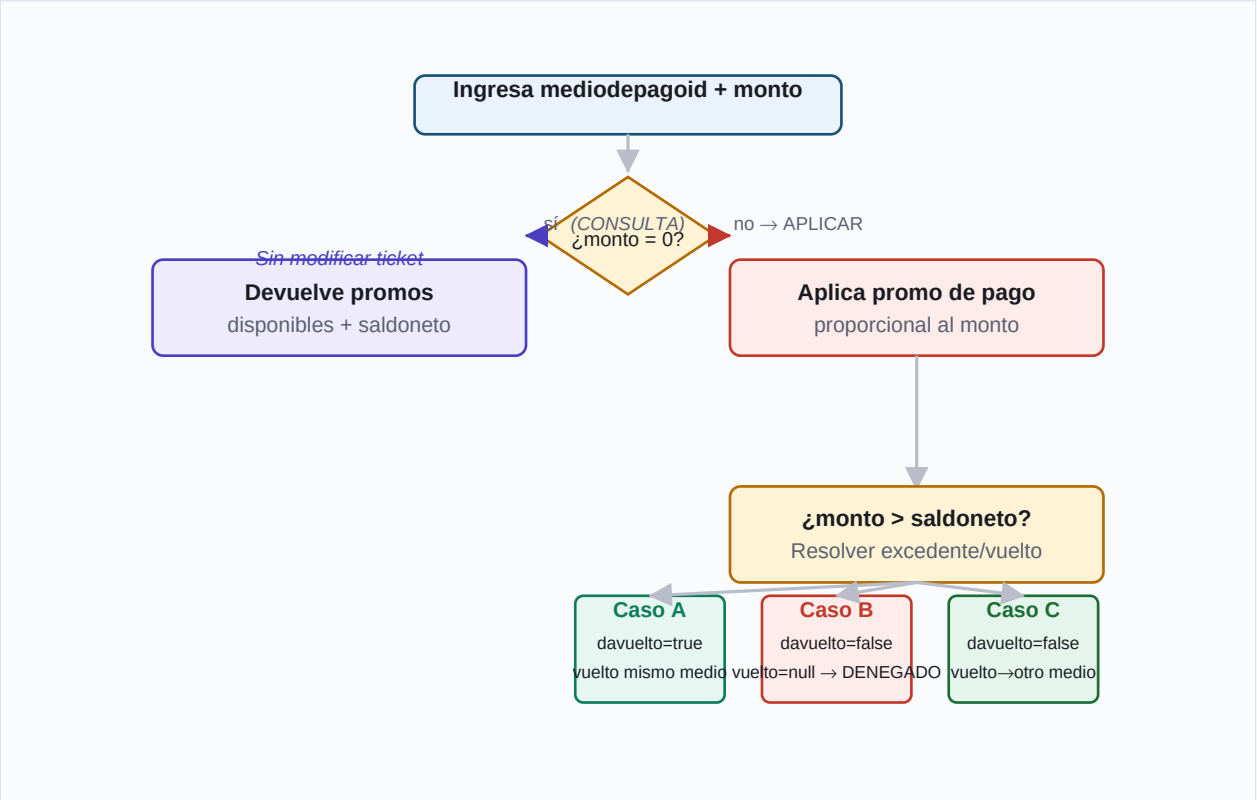


*Figura 5 — Flujo MODO PAGO con sub-etapas CONSULTA y APLICAR*

## 6.3 Sub-etapa CONSULTA

El POS informa el **mediodepagoid**. La API devuelve las promociones disponibles para ese medio y el saldo neto resultante. **No modifica el ticket.**

## 6.4 Sub-etapa APLICAR — escenarios de monto

| Escenario | Condición | Comportamiento |
|---|---|---|
| Pago parcial | monto < saldo neto | Promoción aplicada proporcionalmente al monto. Queda saldo pendiente. |
| Pago exacto | monto = saldo neto | Promoción aplicada completa. Ticket saldado. |
| Pago con excedente | monto > saldo neto | Se consume hasta el saldo neto. Excedente resuelto como vuelto según tiposdepago. |

## 6.5 Lógica de vuelto

| Caso | davuelto | vueltomediodepago | Resultado |
|---|---|---|---|
| A | true | — | Vuelto en el mismo medio de pago. Operación aceptada. |
| B | false | null | No se permite vuelto. Operación DENEGADA. |
| C | false | id_medio | Vuelto en el medio alternativo indicado (habitualmente EFECTIVO). Aceptada. |


*Figura 6 — Flujo de excedente y vuelto en el ledger*

# 7. Ledger — movimientos[]

## 7.1 Convenciones de signos

Las convenciones de signos en **movimientos[]** son fundamentales para la reconciliación algebraica del ticket.

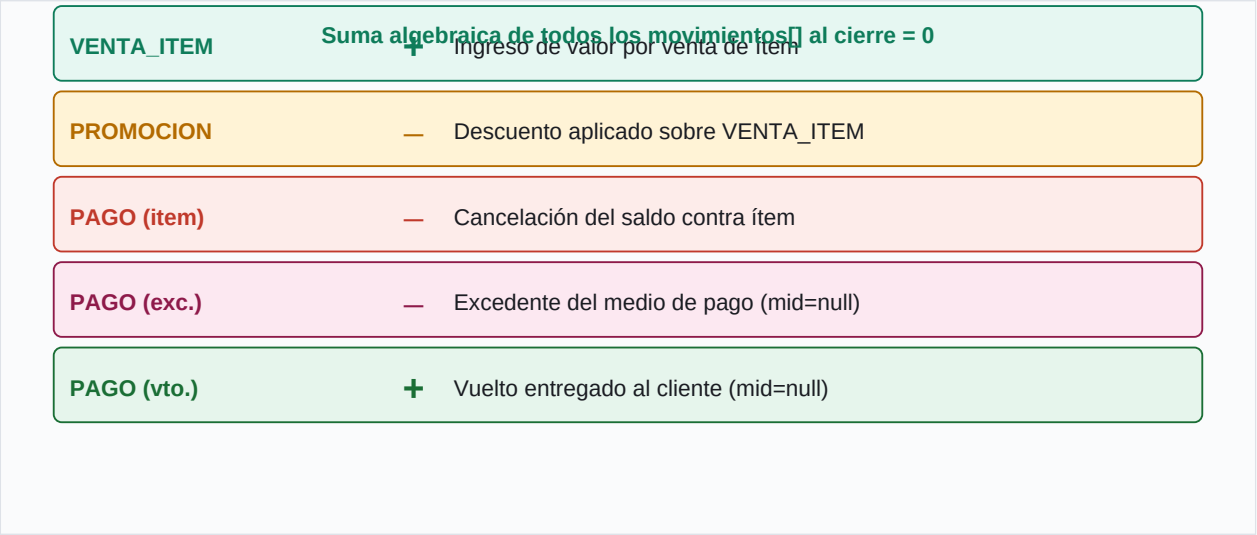


*Figura 7 — Convenciones de signos del ledger*

## 7.2 Semántica de referencias

| Concepto | origenid referencia | movimientoid referencia |
|---|---|---|
| VENTA_ITEM | items[].id | null siempre |
| PROMOCION | promociones[].id | id del VENTA_ITEM afectado |
| PAGO (contra ítem) | pagos[].id | id del VENTA_ITEM cancelado |
| PAGO excedente | pagos[].id | null — no refiere a ítem concreto |
| PAGO vuelto | pagos[].id (vuelto) | null — contrapartida del excedente |

## 7.3 Regla de reconciliación

✓ Regla fuerte del ledger: la suma de todos los nucleoimpositivo.monto en movimientos[] debe dar exactamente 0 al cierre final del ticket. Ventas positivas + promociones negativas + pagos negativos + excedente negativo + vuelto positivo = 0.

| Concepto | Detalle | Subtotal |
|---|---|---|
| VENTA_ITEM ×3 | +1310 +1310 +1310 | +3930.00 |
| PROMOCION | −655 −655 | −1310.00 |
| PAGO (items) | −655 −655 −1310 | −2620.00 |
| PAGO excedente | −380 | −380.00 |
| PAGO vuelto | +380 | +380.00 |
| ✓ Suma algebraica ledger | | = 0.00 |

*Figura 8 — Reconciliación algebraica del ledger (ejemplo PROMO_2X1_ARROZ + CHEQUE + vuelto EFECTIVO)*

# 8. Ejemplo completo integrado

Escenario: 3 unidades de ARROZ ($1310 c/u), PROMO_2X1_ARROZ (50% en 2u, máx 1 aplicación), pago con CHEQUE $3000, vuelto $380 en EFECTIVO.

## 8.1 Artículo y ticket inicial

**JSON — ARTICULOS[] + ITEMS[]**

```
{
  "articulos": [
    {
      "id": 1,
      "articulo": {
        "ean": "7791234567890", "plu": "112233", "descripcion": "ARROZ",
        "pesable": false, "preciolista": 1310.0,
        "rubro": "ALIMENTOS", "depto": "ALMACEN", "marca": "GALLO",
        "nucleoimpositivo": [
          { "impuesto": { "id": "NETO_IVA_21" },           "monto": 1000.0 },
          { "impuesto": { "id": "IVA_21" },                "monto":  210.0 },
          { "impuesto": { "id": "IMPUESTOINTERNO_IVA_21" }, "monto":  100.0 }
        ]
      }
    }
  ],
  "items": [
    { "id": 1, "articuloid": 1, "unidades": 2.0 },
    { "id": 2, "articuloid": 1, "unidades": 1.0 }
  ]
}
```

## 8.2 Resultado MODO ITEM

**JSON — RETORNO MODO ITEM**


```
{
  "promociones": [
    {
      "id": 1, "promocionid": 1, "descripcion": "PROMO_2X1_ARROZ",
      "tipoPromo": "ITEM",
      "promocionestado": { "id": "APLICADA" },
      "monto": -1310.0,
      "elementos": [
        { "movimientoid": 1, "articuloid": 1, "unidadesimpactadas": 1.0, "monto": -655.0 },
        { "movimientoid": 2, "articuloid": 1, "unidadesimpactadas": 1.0, "monto": -655.0 }
      ]
    }
  ],
  "movimientos": [
    { "id": 1, "concepto": "VENTA_ITEM", "origenid": 1, "movimientoid": null,
      "nucleoimpositivo": [{"impuesto":{"id":"NETO_IVA_21"},"monto":1000},
                           {"impuesto":{"id":"IVA_21"},"monto":210},
                           {"impuesto":{"id":"IMPUESTOINTERNO_IVA_21"},"monto":100}] },
    { "id": 2, "concepto": "VENTA_ITEM", "origenid": 1, "movimientoid": null, "nucleoimpositivo": [.
    { "id": 3, "concepto": "VENTA_ITEM", "origenid": 2, "movimientoid": null, "nucleoimpositivo": [.
    { "id": 4, "concepto": "PROMOCION",  "origenid": 1, "movimientoid": 1,
      "nucleoimpositivo": [{"impuesto":{"id":"NETO_IVA_21"},"monto":-500},
                           {"impuesto":{"id":"IVA_21"},"monto":-105},
                           {"impuesto":{"id":"IMPUESTOINTERNO_IVA_21"},"monto":-50}] },
    { "id": 5, "concepto": "PROMOCION",  "origenid": 1, "movimientoid": 2, "nucleoimpositivo": [...]
  ]
}
```


## 8.3 Resultado MODO PAGO — CHEQUE $3000 + vuelto EFECTIVO

**JSON — RETORNO MODO PAGO (CHEQUE + VUELTO EFECTIVO)**


```
{
  "pagos": [
    { "id": 1, "mediodepagoid": 2, "descripcion": "CHEQUE",   "monto":  3000.0 },
    { "id": 2, "mediodepagoid": 1, "descripcion": "EFECTIVO", "monto":  -380.0 }
  ],
  "movimientos": [
    // mov1-5: VENTA_ITEM (×3) + PROMOCION (×2) — igual que MODO ITEM
    { "id": 6, "concepto": "PAGO", "origenid": 1, "movimientoid": 1, "nucleoimpositivo": [NETO:-500,
    { "id": 7, "concepto": "PAGO", "origenid": 1, "movimientoid": 2, "nucleoimpositivo": [NETO:-500,
    { "id": 8, "concepto": "PAGO", "origenid": 1, "movimientoid": 3, "nucleoimpositivo": [NETO:-1000
    { "id": 9, "concepto": "PAGO", "origenid": 1, "movimientoid": null,
      "nucleoimpositivo": [{"id":"NETO_IVA_21","monto":-290.08},
                           {"id":"IVA_21","monto":-60.92},
                           {"id":"IMPUESTOINTERNO_IVA_21","monto":-29.0}] },
    { "id":10, "concepto": "PAGO", "origenid": 2, "movimientoid": null,
      "nucleoimpositivo": [{"id":"NETO_IVA_21","monto": 290.08},
                           {"id":"IVA_21","monto":  60.92},
                           {"id":"IMPUESTOINTERNO_IVA_21","monto": 29.0}] }
  ],
  "resultado": {
    "estado": "ACEPTADO", "saldopendiente": 0.0, "vuelto": 380.0, "vueltomediodepagoid": 1
  }
}
```


## 8.4 Verificación de reconciliación

| Verificación | Cálculo | Resultado | OK |
|---|---|---|---|
| Items bruto | 3 × $1310 | $3930.00 | ✓ |
| Promo maestro | promociones[1].monto | –$1310.00 | ✓ |
| Movs PROMOCION (suma) | mov4+mov5 | –$1310.00 | ✓ |
| Neto ticket | $3930 – $1310 | $2620.00 | ✓ |
| Pagado neto | $3000 – $380 | $2620.00 | ✓ |
| Saldo final | $2620 – $2620 | $0.00 | ✓ |
| Excedente + vuelto | mov9 + mov10 | $0.00 | ✓ |
| Suma ledger total | Σ mov1..10 | $0.00 | ✓ |

# 9. Orden base de cálculo

Orden sugerido por core.md sección 9, salvo regla explícita en contrario:

| Paso | Acción | Observación |
|---|---|---|
| 1 | Identificar cliente | Determina tipo de comprobante y alícuotas aplicables |
| 2 | Ingresar ítems | Deduplicar articulos[], crear items[] con articuloid y unidades |
| 3 | Resolver referencias a artículos | Cada item referencia su articuloid |
| 4 | Determinar bases iniciales | preciolista × unidades por ítem |
| 5-6 | Aplicar promociones ITEM | MODO ITEM: evaluar listapromociones con alcance=ITEM |
| 6 | Recalcular bases afectadas | Precio post-promo por ítem para cálculos siguientes |
| 7 | Calcular impuestos y núcleo | nucleoimpositivo del ticket |
| 8 | Determinar total | datosreferenciales.total |
| 9 | Pre-consulta promos de pago | MODO PAGO CONSULTA: promos disponibles por medio |
| 10 | Registrar pagos + aplicar promos de pago | MODO PAGO APLICAR: registrar en pagos[] y movimientos[] |
| 11 | Calcular excedente y vuelto | Registrar mov PAGO excedente y PAGO vuelto |
| 12 | Distribuir pagos en ledger | Movimientos PAGO contra VENTA_ITEM |
| 13 | Calcular saldo final | datosreferenciales.saldo |
| 14 | Validar reconciliación | Σ movimientos[] = 0 |

# 10. Convenciones de implementación Java 8

La implementación objetivo es **Java 8 clásico**. El diseño debe ser claro, simple y auditable.

## 10.1 Restricciones del modelo Java

- **POJOs simples:** Clases de datos sin dependencias de frameworks de mapping.
- **Campos privados:** Todos los atributos son private.
- **Getters/setters clásicos:** Métodos getX() / setX() explícitos.
- **Constructor por defecto:** Siempre presente para serialización.
- **Métodos explícitos:** Sin azúcar sintáctico de versiones posteriores.
- **Sin Lombok:** No usar @Data, @Builder, @Value ni similares.
- **Sin records:** Introducidos en Java 16 — fuera de scope.
- **Sin builders:** No usar el patrón builder generado automáticamente.
- **Sin features post-Java 8:** No usar var, streams avanzados, sealed classes, etc.

## 10.2 Estructura de paquetes

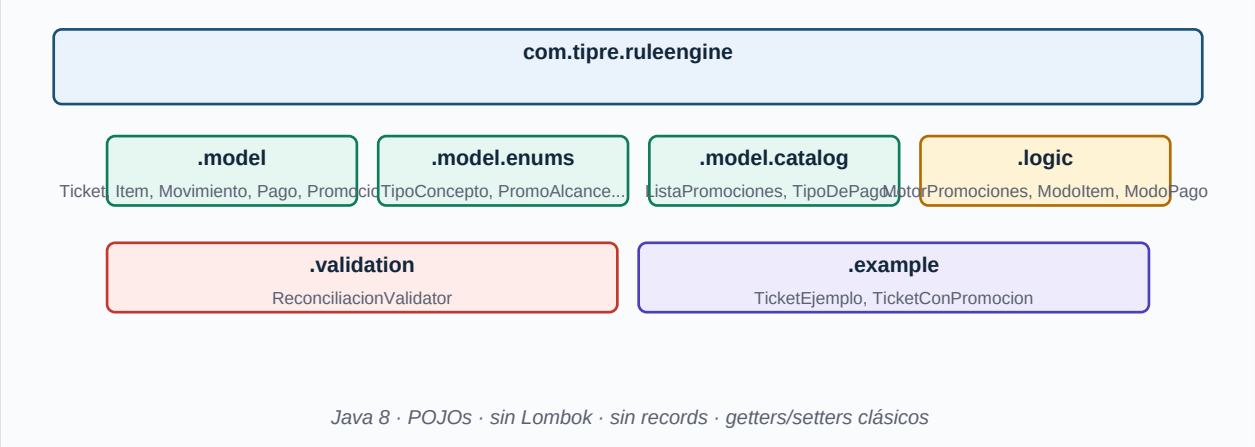


*Figura 9 — Estructura de paquetes sugerida*

| Paquete | Contenido |
|---|---|
| com.tipre.ruleengine.model | Ticket, Item, Articulo, Movimiento, Pago, Promocion, NucleoImpositivo |
| com.tipre.ruleengine.model.enums | TipoConcepto, PromoAlcance, PromoBeneficio, PromoMetodo, PromoDecision, PromoEst |
| com.tipre.ruleengine.model.catalog | ListaPromociones, DefinicionPromocion, ListaPromoEntry, TipoDePago, Vigencia |
| com.tipre.ruleengine.logic | MotorPromociones, ModoItemProcessor, ModoPagoProcessor, BeneficioCalculator |
| com.tipre.ruleengine.validation | ReconciliacionValidator, LedgerConsistencyChecker |
| com.tipre.ruleengine.example | TicketEjemplo, TicketConPromocion2x1 |

## 10.3 Ejemplo de POJO — Item

**JAVA 8 — ITEM.JAVA**

```java
package com.tipre.ruleengine.model;

public class Item {

    private Integer id;
    private Integer articuloid;
    private Double unidades;

    public Item() {}

    public Integer getId() { return id; }
    public void setId(Integer id) { this.id = id; }

    public Integer getArticuloid() { return articuloid; }
    public void setArticuloid(Integer articuloid) { this.articuloid = articuloid; }

    public Double getUnidades() { return unidades; }
    public void setUnidades(Double unidades) { this.unidades = unidades; }
}
```

## 10.4 Ejemplo de POJO — Movimiento

**JAVA 8 — MOVIMIENTO.JAVA**

```java
package com.tipre.ruleengine.model;

import java.util.List;

public class Movimiento {

    private Integer id;
    private String concepto;       // VENTA_ITEM | PROMOCION | PAGO
    private Integer origenid;
    private Integer movimientoid;  // null para excedentes y vuelto
    private List<NucleoImpositivoEntry> nucleoimpositivo;

    public Movimiento() {}

    // getters y setters ...
}
```

## 10.5 Criterios de revisión

| Área | Verificación |
|---|---|
| Consistencia de dominio | Objetos bien separados, responsabilidades claras, sin pérdida de granularidad |
| Consistencia de cálculo | Orden lógico correcto, promociones e impuestos interactúan de forma explícita |
| Consistencia de ledger | Separación maestro/distribuido, movimientos trazables y reconciliables |
| Auditabilidad | Todo monto relevante puede explicarse, toda transformación puede rastrearse |

# 11. Decisiones diferidas y extensiones futuras

Las siguientes áreas están reconocidas en el modelo pero no han sido definidas en esta versión. Su incorporación futura requiere decisión explícita y no debe inferirse del modelo actual.

| Área | Estado | Impacto esperado |
|---|---|---|
| Filtros por tipo de cliente / convenio / fidelización | Diferido | Nuevo atributo en listapromociones, filtro adicional en el motor |
| Orden de evaluación / prioridad entre promociones | Diferido | Campo prioridad en listapromociones, cambio en el algoritmo de evaluación |
| Precio mayorista (NUEVOPRECIOITEM) | Definido en catálogo | Requiere manejo especial en la lógica de acumulación |
| COMBO como método de cómputo | Definido en catálogo | Lógica de combinación de ítems no especificada aún |
| LISTA3, LISTA4... en promocionlistanumber | Extensible | Agregar valores al catálogo sin cambios estructurales |
| IDs externos vs internos en artículos | Pendiente | Diferenciar ids autogenerados locales de ids externos del sistema |

→ Este documento debe mantenerse consistente con core.md y con los JSON del modelo. Ante cualquier contradicción, la jerarquía es: core.md > promociones.md > JSON > históricos.