E-commerce Operacional LATAM

Lecciones de producción de un ERP multi-tenant real: DTE/SII, marketplaces, concurrencia y multi-moneda

Rafael Farías

2026-06-03

Tabla de contenidos

Prefacio

Este libro documenta lo que aprendí construyendo y operando un ERP multi-tenant real para e-commerce en Chile: el sistema que sincroniza marketplaces, emite documentos tributarios electrónicos (DTE) ante el SII, coordina despachos, concilia pagos y lleva la contabilidad de un negocio que vende todos los días.

No es un libro de teoría ni un tutorial de "hola mundo". Es el registro honesto de las decisiones que no se pueden deshacer, los bugs que costaron dinero real, y los patrones que separan un sistema que *funciona en la demo* de uno que *funciona bajo carga, con plata de por medio*.

Para quién es este libro

Para quien construye u opera el software que mueve un e-commerce en LATAM: el desarrollador full-stack que integra MercadoLibre o Falabella, el fundador técnico que monta su propio ERP en vez de pagar licencias, el equipo de 2-5 personas que mantiene facturación electrónica, stock y despachos sin un departamento de IT.

Asumo que sabes programar (TypeScript/SQL) y que ya peleaste con al menos una API de un tercero. No asumo que conozcas las particularidades del SII, de MercadoLibre Full o de los redondeos del IVA chileno: de eso trata buena parte del libro.

Qué vas a poder hacer al terminar

- Decidir la arquitectura de un ERP multi-tenant sin pintarte en una esquina.
- Manejar concurrencia e idempotencia para que dos webhooks simultáneos no dupliquen stock ni cobros.
- Emitir DTE (boletas y facturas) sin que el SII te rechace por decimales o campos faltantes.
- Integrar marketplaces y despachos sobreviviendo a sus *gotchas* no documentadas.
- Llevar el dinero (IVA, redondeo, multi-moneda) sin descuadres de "un peso".
- Decidir si tu ERP interno puede convertirse en un producto SaaS vendible.

Cómo está escrito este libro

Cada capítulo sigue el patrón: **problema concreto → qué aprendimos → el patrón → cómo verificarlo**. Las lecciones vienen de producción, con su fecha y su costo.

Nota de transparencia

Este apartado es parte de la propuesta de valor del libro, no un trámite.

- **Fuentes técnicas.** Las lecciones de ingeniería provienen de un sistema real en producción. Presento **patrones**, no la base de código propietaria ni datos de clientes. Donde describo el comportamiento de una API de un tercero (MercadoLibre, SII vía OpenFactura, Shipit, etc.), es lo que observamos en una fecha concreta: **verifica siempre contra la documentación oficial vigente**, porque estas plataformas cambian sin avisar.
- **Fuentes citadas.** Las ideas de negocio y producto están respaldadas por fuentes citadas y verificables (ver Referencias), gestionadas en Zotero. Ninguna cita se escribió de memoria.
- **Erratas.** Las correcciones se publican en la página de erratas del libro: https://libro.cavara.cl. Si encuentras un error —técnico o de cita—, repórtalo ahí y lo corregimos en la siguiente revisión.

Créditos

E-commerce Operacional LATAM
Lecciones de producción de un ERP multi-tenant real: DTE/SII, marketplaces, concurrencia y multi-moneda

Por Rafael Farías.

Primera edición — junio de 2026.

Aviso de responsabilidad

Este libro tiene fines educativos y se entrega **"tal cual", sin garantías de ningún tipo**. Las decisiones de arquitectura, seguridad, tributarias y de integración que aquí se describen reflejan la experiencia del autor en un contexto concreto y **pueden no aplicar a tu caso**: evalúalas con criterio antes de llevarlas a producción.

El comportamiento de las APIs y plataformas de terceros (MercadoLibre, el SII vía OpenFactura, Shipit y otras) cambia sin previo aviso. Los detalles técnicos reflejan lo observado en una fecha concreta; **verifica siempre contra la documentación oficial vigente**. El autor no se responsabiliza por pérdidas o daños derivados del uso de este material.

Las marcas comerciales y nombres de productos mencionados pertenecen a sus respectivos titulares y se citan únicamente con fines informativos, sin ánimo de infracción.

Erratas y correcciones: https://libro.cavara.cl

Parte I.

Parte I — Por qué construir

1. El problema: operar un e-commerce en LATAM

1.1. El caos multicanal

Un e-commerce que crece en LATAM termina, casi sin darse cuenta, operando en cinco o seis sistemas que no se hablan entre sí: una tienda propia (WooCommerce o Shopify), uno o más marketplaces (MercadoLibre, Falabella, Walmart), un proveedor de despachos (Shipit), una bodega (propia o 3PL), un emisor de DTE para el SII, y una pasarela de pagos. Cada uno con su propio catálogo, su propio stock y su propia verdad.

El síntoma que delata el problema es siempre el mismo: el inventario. Tienes 8 unidades de un SKU en bodega, pero MercadoLibre cree que hay 12 porque ayer no alcanzaste a descontar dos ventas, WooCommerce muestra 5 porque alguien lo ajustó a mano la semana pasada, y Falabella todavía publica 10. No hay un número de stock; hay cuatro números distintos que envejecen a velocidades distintas. Mientras nadie vende, la discrepancia es invisible. El problema aparece el viernes de un *cyber*, cuando tres canales venden la última unidad de un producto al mismo tiempo. Eso es overselling: vendiste algo que no tienes, y ahora alguien tiene que cancelar la venta, pedir disculpas y comerse la mala calificación en el marketplace.

A esto se suma la conciliación. El dinero entra por varios caminos —Mercado Pago, Transbank, depósitos de los marketplaces que pagan a 15 o 30 días con comisiones y retenciones descontadas— y cada uno reporta a su manera. El dueño termina el mes con una planilla de Excel donde pega, fila por fila, las ventas de cada canal contra los abonos del banco, tratando de entender por qué el depósito de MercadoLibre no calza con lo que vendió. Esa planilla *es* el ERP que todavía no construyó.

1.1.1. El costo invisible

Lo caro de operar así no aparece en ninguna factura, y por eso casi nadie lo mide. Son tres fugas simultáneas:

- **Horas-persona.** Copiar pedidos de un panel a otro, descontar stock a mano, generar etiquetas de despacho una por una, cuadrar pagos. Trabajo repetitivo que no escala: el día que duplicas las ventas, duplicas las horas de tipeo.
- **Errores.** Cada copiar-y-pegar es una oportunidad de equivocar un RUT, una dirección o una cantidad. Un dígito mal en un DTE no es un typo: es un documento tributario que el SII puede rechazar, o peor, aceptar mal.

- **Ventas perdidas y reputación.** El overselling cancela ventas reales y baja la reputación en marketplaces que rankean por tasa de cancelación. En MercadoLibre, perder la categoría de reputación puede sacarte de los primeros resultados de búsqueda. Es un costo de oportunidad que nunca verás como línea en un reporte.

La trampa es que cada fuga, por separado, parece tolerable. "Son solo dos horas al día." "Fue solo un pedido cancelado." Sumadas y proyectadas sobre el crecimiento, son la diferencia entre un negocio que escala y uno que se ahoga en su propia operación.

> ⚠ El costo crece con el cuadrado de los canales
>
> Con un solo canal no hay nada que sincronizar. Con dos, hay un par de números que cuadrar. Con cinco, hay diez pares de relaciones que pueden desincronizarse. El trabajo manual de conciliación no crece lineal con los canales: crece con el número de *pares* de canales. Por eso el caos llega de golpe, no de a poco.

1.1.2. Por qué las soluciones "todo en uno" no calzan

La reacción obvia es comprar una suite genérica internacional y delegar el problema. El choque viene cuando esa suite, diseñada para Estados Unidos o Europa, se encuentra con la realidad regulatoria chilena:

- **DTE y el SII.** En Chile no emites "una factura": emites un Documento Tributario Electrónico que debe firmarse, foliarse contra un CAF autorizado y reportarse al SII en tiempos definidos. Boletas, facturas, notas de crédito y guías de despacho tienen reglas propias. Una suite genérica no tiene este modelo; lo "resuelve" con un PDF que no es válido tributariamente.
- **El RUT.** El identificador del cliente y de la empresa no es un campo de texto cualquiera: tiene dígito verificador, formato propio y es la llave con la que se asocia todo documento tributario. Un sistema que lo trata como string libre genera duplicados y documentos imposibles de conciliar.
- **Sucursales y modalidades de despacho.** El SII pide declarar sucursales; los marketplaces tienen sus propias modalidades (retiro, despacho propio, fulfillment del marketplace). Mapear esto contra un modelo de "warehouse" genérico se rompe en el primer caso real.

No es que las suites internacionales sean malas. Es que el *trabajo* que necesitas hacer está atravesado por reglas locales que ellas no modelan. Y ahí es donde conviene cambiar de pregunta: en vez de "¿qué software compro?", preguntar "¿qué trabajo necesito que se haga?".

1.2. El trabajo que el ERP viene a hacer

Antes de escribir una línea de código conviene preguntarse qué "trabajo" contrata el operador cuando decide construir o comprar un ERP. La Teoría de los Jobs to Be Done lo plantea así:

> "A job is defined as the progress that a customer desires to make in a particular circumstance." (Christensen et al., 2016)

El operador no quiere "un ERP"; quiere **dejar de adivinar** y dejar de operar a mano. Vale la pena mirar dónde la gente ya inventa parches:

> "If you observe people employing a workaround or 'compensating behavior' to get a job done, pay close attention. It's usually a clue that you have stumbled on to a high-potential innovation opportunity." (Christensen et al., 2016)

La planilla de Excel con la que el dueño concilia pagos *es* ese workaround. Ahí está el trabajo a resolver. Y no está solo: la pizarra con los pedidos del día, el grupo de WhatsApp donde se avisa "ojo que ese producto se agotó", la libreta donde se anota qué falta despachar. Cada parche es una pista de un trabajo que el sistema actual no hace, y que alguien suple con esfuerzo manual.

1.2.1. Las tres dimensiones del trabajo

El *job* nunca es solo funcional. Christensen insiste en que tiene tres capas, y un ERP que solo resuelve la primera se siente incompleto:

Dimensión	Lo que el operador busca	Síntoma de que falta
Funcional	Que el stock cuadre, que el DTE se emita, que el pedido llegue	Overselling, documentos rechazados
Emocional	Dormir tranquilo; no temer al *cyber* ni al cierre de mes	Ansiedad, revisar el celular a medianoche
Social	Verse profesional ante el cliente y ante el SII	Vergüenza por una boleta mal hecha o un despacho atrasado

La dimensión funcional es la que todos ven y la que más fácil se especifica. Pero la que decide si el operador *adopta* o *abandona* el sistema suele ser la emocional. El dueño que dejó de despertarse a revisar si hubo overselling no te lo va a poner en un ticket de feature, pero es la razón real por la que sigue usando el sistema. Y la social es la que te hace ganar o perder al cliente: una boleta bien emitida y un despacho que llega cuando dijiste construyen una confianza que ninguna campaña de marketing compra.

El valor, entonces, no es una propiedad del software: es el progreso que el operador logra hacer. Como lo resume Klement, **el progreso define el valor; el contraste lo revela** (Klement, 2018). Y el contraste relevante aquí es revelador: tu ERP no compite contra otro ERP. Compite contra **seguir haciéndolo a mano**. El operador no se pregunta "¿este ERP es mejor que aquel?", sino "¿esto es suficientemente mejor que mi planilla como para justificar cambiar?". Si la respuesta no es un sí evidente, se queda con la planilla, porque la planilla, aunque dolorosa, ya la conoce.

> **Diseña contra el workaround, no contra el competidor**
>
> Cuando dudes de una decisión de producto, no la compares con lo que hace otro ERP. Compárala con la planilla, la pizarra o el WhatsApp que el operador usa hoy. Si tu solución no es claramente mejor que *ese* parche concreto, todavía no resuelve el trabajo.

1.3. Operar con datos, no con intuición

La razón de fondo para centralizar la operación es poder decidir con datos:

> "Most stores throw stuff up just to see what sticks; they're literally guessing at business. Data tracking is a crystal ball [...] so you can make informed business decisions." (Larsson, 2016)

Un ERP bien hecho no es solo un sincronizador: es la única fuente de verdad desde la cual se miden margen por producto, costo de adquisición y rentabilidad por cliente. Mientras los datos vivan repartidos en cinco paneles, ninguna pregunta de negocio tiene una respuesta confiable. Cuando viven en un solo modelo, las respuestas aparecen casi solas.

1.3.1. Qué métricas importan desde el día 1

El riesgo al "operar con datos" es enamorarse de las *vanity metrics*: visitas totales, seguidores, ventas brutas. Suben bonito en un gráfico y no informan ninguna decisión. Las métricas que importan desde el primer día son las que cambian lo que haces mañana:

- **Margen por producto**, no solo precio de venta. Un producto que vende mucho con margen negativo te quiebra más rápido que uno que vende poco. Necesitas el costo real (compra + flete + comisión del canal) contra el precio neto.
- **Rentabilidad por cliente y por canal.** No todos los clientes ni todos los canales valen lo mismo. Un canal con alto volumen pero comisiones altas y muchas devoluciones puede ser menos rentable que uno chico y limpio.
- **Stock disponible real**, descontado de reservas en curso. No el stock teórico: el que de verdad puedes prometer sin caer en overselling.
- **Tasa de cancelación y devoluciones por canal.** Es el indicador temprano de problemas de operación y de reputación.

Una sola consulta sobre datos centralizados ilustra la diferencia entre adivinar y saber:

```sql
-- Margen real por producto, neto de comisión de canal.
-- ILUSTRATIVO del patrón, no un esquema de producción.
SELECT
  p.sku,
  p.nombre,
  SUM(li.cantidad)                AS unidades,
  SUM(li.precio_neto)             AS ingreso_neto,
  SUM(li.costo + li.comision_canal) AS costo_total,
  SUM(li.precio_neto)
```

```sql
    - SUM(li.costo + li.comision_canal) AS margen
FROM linea_item li
JOIN producto p ON p.id = li.producto_id
WHERE li.fecha >= date_trunc('month', now())
GROUP BY p.sku, p.nombre
ORDER BY margen ASC;   -- los de margen negativo, primero
```

Esa consulta es imposible mientras el costo viva en una planilla, las comisiones en el panel de cada marketplace y las ventas repartidas en cinco sistemas. Centralizar no es un fin estético: es lo que hace que preguntas como esta tengan respuesta.

1.3.2. El modelo de "fábrica de clientes"

Una forma útil de pensar el negocio completo —no solo el inventario— es verlo como una fábrica por la que fluyen clientes, en la línea de lo que propone Maurya (Maurya, 2016). En esa fábrica importan tres variables: el *throughput* (clientes que realmente llegan a comprar y volver), el inventario de oportunidades atascadas en el proceso (carritos abandonados, cotizaciones sin cerrar, pedidos sin despachar) y los gastos operacionales de hacer todo eso funcionar. El objetivo no es maximizar una métrica suelta, sino aumentar el flujo que sale convertido en dinero mientras bajas el inventario atascado y los gastos.

Mirado así, el caos multicanal del inicio deja de ser un problema "de TI" y se vuelve lo que realmente es: una restricción de la fábrica. Cada hora gastada conciliando a mano es inventario atascado y gasto operacional puro. El ERP, en este modelo, es la intervención sobre el cuello de botella: libera horas, destraba pedidos y deja ver el flujo real para poder mejorarlo.

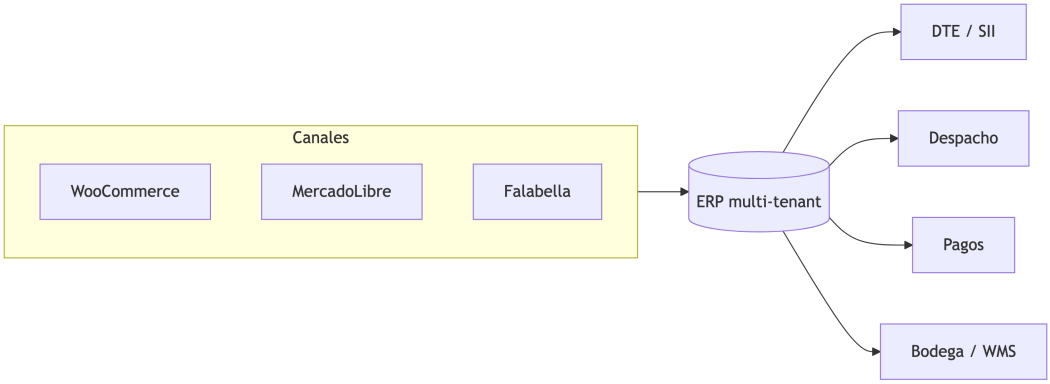
```
        fábrica de clientes

  visitas → carritos → pedidos → despachados → recompra

    throughput  = los que cruzan toda la línea
    inventario  = atascados (carritos, sin
                  despachar, sin cobrar)
    gasto op.   = horas y errores de operar
```

Vanity vs. accionable

Regla simple para distinguirlas: si una métrica sube y no cambia ninguna decisión que tomarás mañana, es vanity. Margen por producto cambia qué compras y a qué precio vendes; "visitas totales" rara vez cambia algo. Mide lo segundo solo si ya tienes lo primero.

1.4. Qué vamos a construir en este libro

Con el problema claro —caos multicanal, costo invisible, reglas locales que las suites genéricas no modelan— podemos dibujar el sistema que lo resuelve. La idea central es simple: los canales dejan de hablar entre sí y pasan a hablar todos con un único centro, el ERP, que es la fuente de verdad. Desde ese centro salen las integraciones hacia el SII, el despacho, los pagos y la bodega.



Ese diagrama es el mapa del resto del libro. Cada flecha es un capítulo:

- **El centro como fuente de verdad** y por qué un ERP de e-commerce serio nace multi-tenant lo ves en arquitectura.
- **El problema del overselling**, que es en el fondo una *race condition*, se resuelve con las técnicas de concurrencia y con un buen sistema de jobs en background para sincronizar sin bloquear.
- **El dinero** —conciliación, comisiones, IVA neto vs. bruto— tiene su propio capítulo en dinero.
- **Las integraciones**: los patrones comunes en integraciones, los marketplaces, el DTE con el SII y los despachos y bodega.
- **Que todo esto sea observable y sobreviva en producción** se trata en deploy y observabilidad, y los tropiezos reales en errores.

A lo largo del camino, el código que verás es **ilustrativo del patrón, no copia exacta de un sistema en producción**. Los detalles de las APIs de terceros —endpoints, campos, códigos de error, límites— se presentan como *lo que observamos operando a una fecha*; antes de implementar, verifica siempre contra la documentación oficial vigente, porque cambian. El valor del libro no está en pegar un endpoint exacto, sino en entender el trabajo que el sistema viene a hacer y las decisiones que lo hacen sostenible.