GUÍA COMPLETA: MÓDULO DE INTEGRACIONES EXTERNAS (DELIVERY)

Versión: 1.0
Fecha: 2026-01-19
Tipo: Documentación Técnica + Guía de Usuario

ÍNDICE

RESUMEN EJECUTIVO

¿Qué es este módulo?

El módulo de **Integraciones Externas** permite a tu restaurante recibir pedidos automáticamente desde plataformas de delivery como:

- 🟢 **Rappi**
- 🟡 **Glovo**
- 🔴 **PedidosYa**
- ⚫ **UberEats** (estructura preparada)

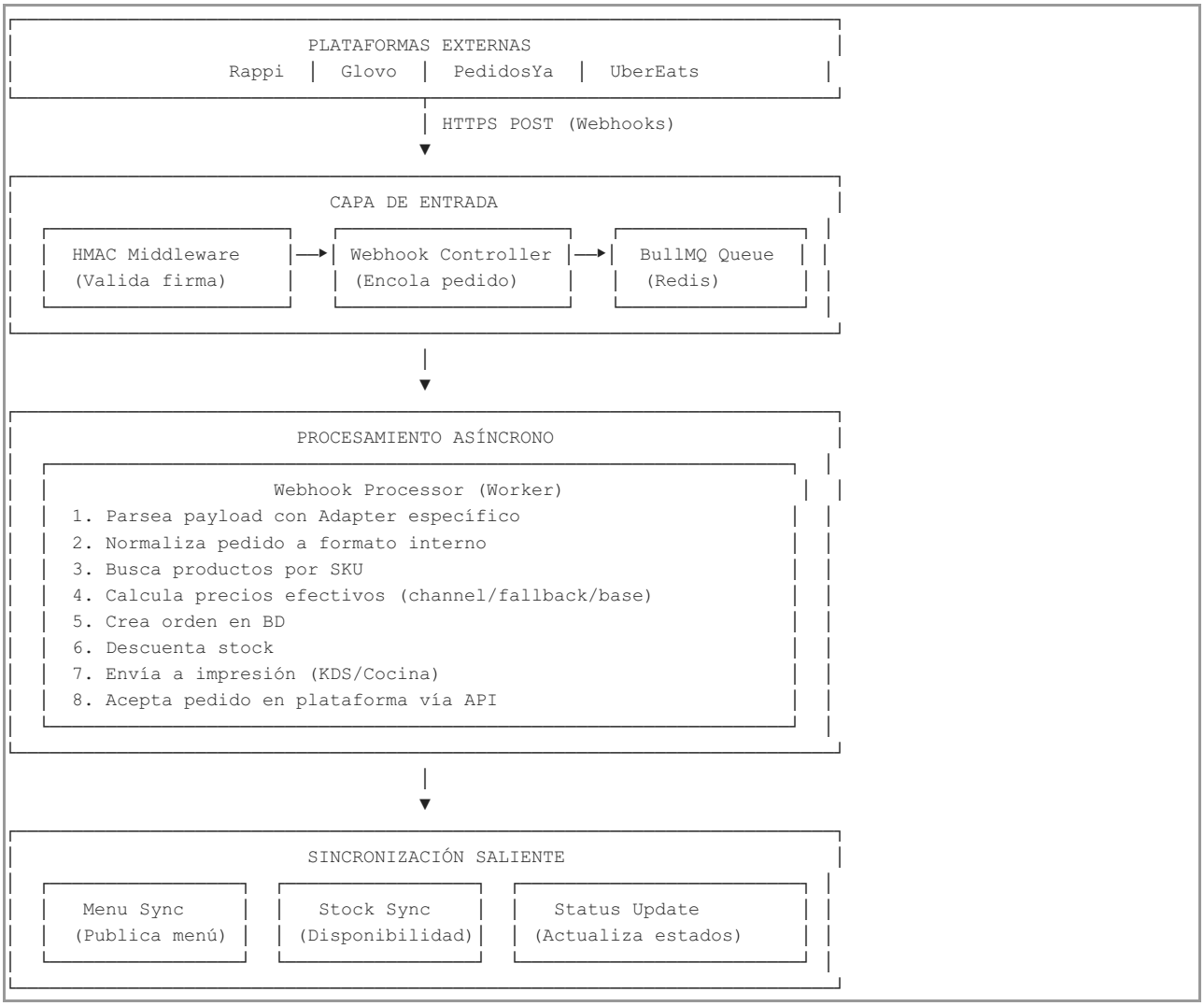
¿Qué problema resuelve?

Sin el Módulo	Con el Módulo
Un empleado debe mirar cada app constantemente	Los pedidos llegan automáticamente al POS
Errores al transcribir pedidos manualmente	Pedidos creados automáticamente sin errores
No hay control de stock integrado	El stock se actualiza automáticamente
Precios diferentes en cada plataforma = caos	Precios configurables por plataforma
Sin visibilidad de comisiones	Comisiones calculadas automáticamente

---

ARQUITECTURA TÉCNICA

Componentes del Sistema



Estructura de Archivos

```
backend/src/integrations/delivery/
├── adapters/                       # Adaptadores por plataforma
│   ├── AbstractDeliveryAdapter.ts  # Clase base abstracta
│   ├── AdapterFactory.ts           # Factory para crear adapters
│   └── RappiAdapter.ts             # Implementación Rappi
│
├── webhooks/                       # Recepción de webhooks
│   ├── hmac.middleware.ts          # Validación de firma HMAC
│   ├── webhook.controller.ts       # Controller HTTP
│   └── webhook.routes.ts           # Rutas Express
│
├── jobs/                           # Procesamiento asíncrono
│   └── webhookProcessor.ts         # Worker BullMQ
│
├── sync/                           # Sincronización saliente
│   ├── menuSync.service.ts         # Sincronizar menú
│   ├── stockSync.service.ts        # Sincronizar stock
│   └── statusUpdate.service.ts     # Actualizar estados
│
└── types/
    └── normalized.types.ts         # Tipos normalizados
```

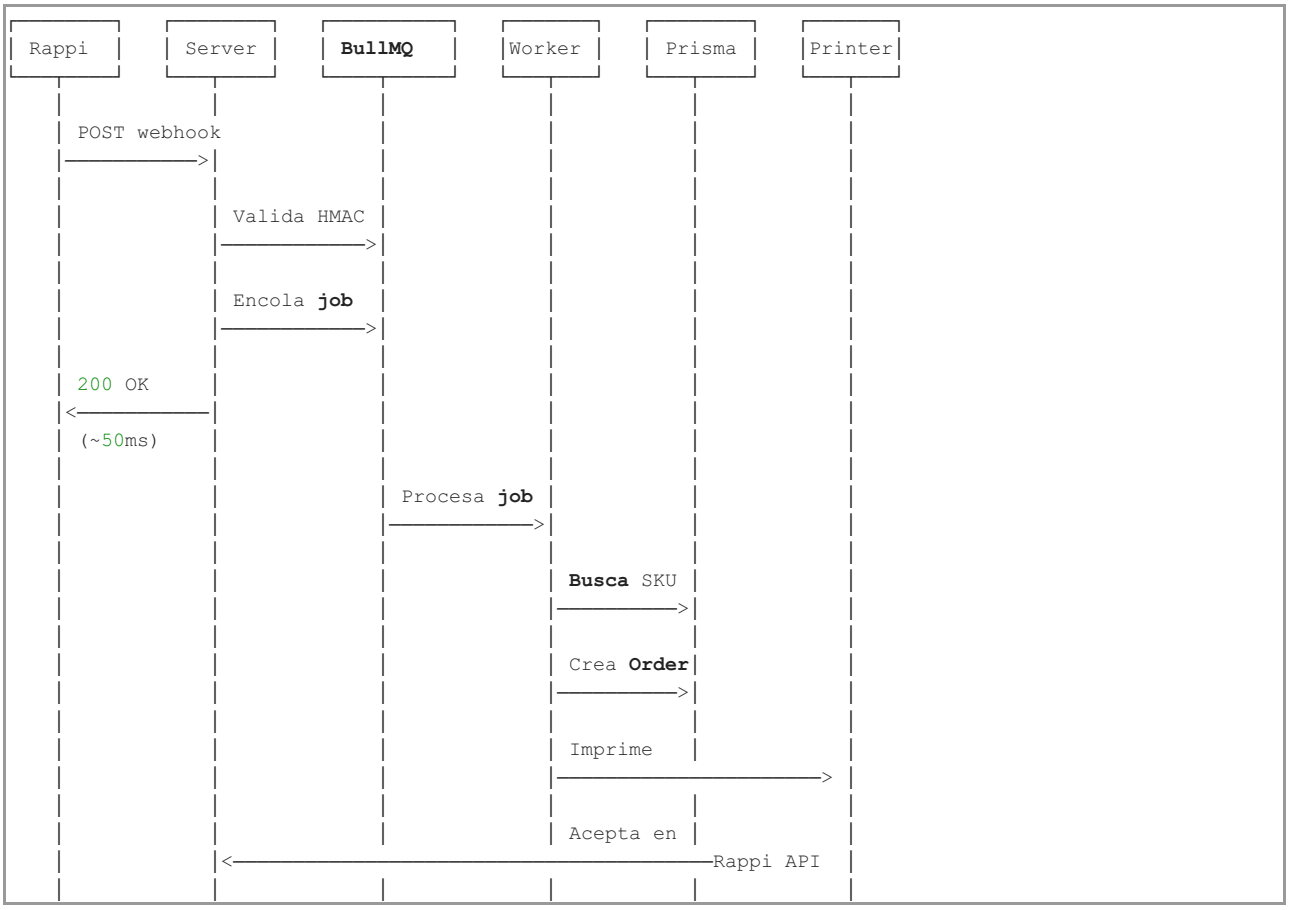
Patrón de Diseño: Adapter Pattern

Cada plataforma tiene su propio formato de datos. El sistema usa el **Adapter Pattern** para normalizar todos a un formato común:

```
// Formato Rappi (diferente)
{
  order_id: "RAP-123",
  customer: { first_name: "Juan", last_name: "Pérez" },
  items: [{ sku: "PIZZA-001", unit_price: 1500 }]
}

// Formato Glovo (diferente)
{
  id: "GLV-456",
  client: { name: "María García" },
  products: [{ reference: "PIZZA-001", price: 1650 }]
}

// ⬇️ AMBOS se transforman a ⬇️

// Formato Normalizado (único)
{
  externalId: "RAP-123",
  platform: "RAPPI",
  customer: { name: "Juan Pérez" },
  items: [{ externalSku: "PIZZA-001", unitPrice: 1500 }]
}
```

FLUJO COMPLETO DE UN PEDIDO

Diagrama de Secuencia



Paso a Paso Detallado

1️⃣ Cliente hace pedido en Rappi

El cliente usa la app de Rappi, selecciona productos del menú de tu restaurante y confirma el pedido.

2️⃣ Rappi envía Webhook

Rappi envía un `POST` a tu servidor con los datos del pedido:

```
POST https://tu-servidor.com/api/v1/webhooks/rappi
Headers:
  X-Rappi-Signature: sha256=abc123...
Body:
  { order_id: "RAP-99887", items: [...], totals: {...} }
```

3 Validación de Seguridad (HMAC)

El middleware verifica que el webhook viene realmente de Rappi usando firma HMAC-SHA256:

```
// El servidor calcula: HMAC-SHA256(body, webhookSecret)
// Si coincide con X-Rappi-Signature → VÁLIDO
// Si no coincide → 401 Unauthorized
```

4 Encolamiento Inmediato

Para responder rápido (< 100ms), el webhook se encola en Redis:

```
await queueService.enqueue("delivery:webhooks", {
  platform: "RAPPI",
  eventType: "ORDER_NEW",
  payload: webhookData,
  requestId: "wh_1737329800_abc123",
});
return res.status(200).json({ success: true });
```

5 Procesamiento Asíncrono

El worker toma el job de la cola y:

1. **Parsea** el payload con `RappiAdapter.parseWebhookPayload()`
2. **Mapea** productos externos a productos internos por SKU
3. **Calcula** precios efectivos (ver sección de Precios)
4. **Crea** la orden en la base de datos
5. **Descuenta** stock automáticamente
6. **Imprime** comanda en cocina (si configurado)
7. **Acepta** el pedido en Rappi vía API

6 El pedido aparece en el POS

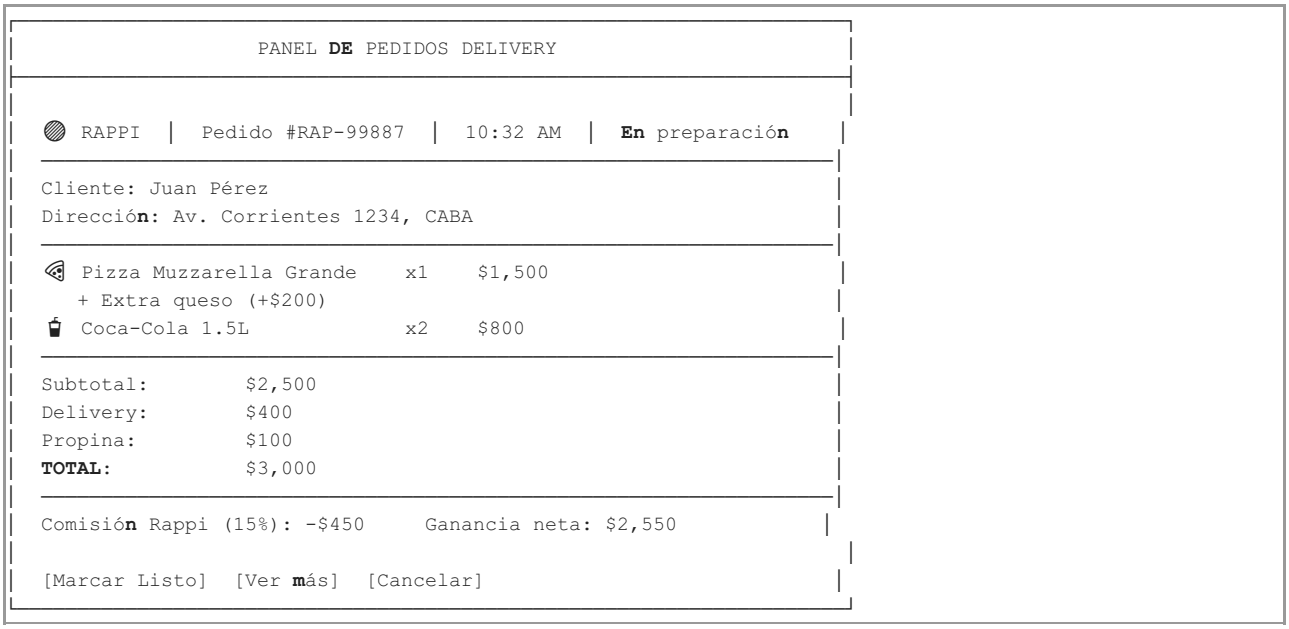
El pedido aparece automáticamente en:

- Pantalla de cocina (KDS)
- Lista de pedidos delivery
- Dashboard en tiempo real

GUÍA PARA EL RESTAURANTE

¿Cómo funciona para mi negocio?

Lo que VES en el sistema:



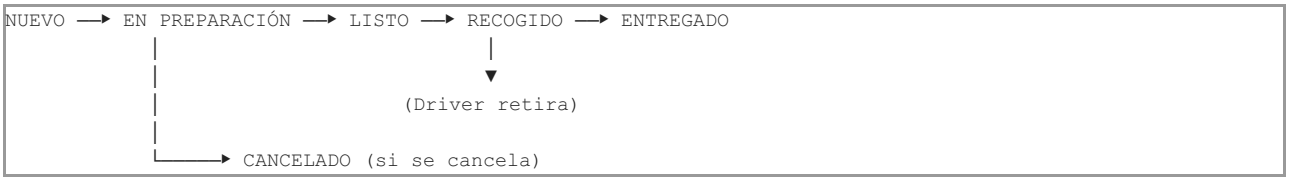
Lo que HACE el sistema automáticamente:

Acción	Automático	Manual
Recibir el pedido	☑	-
Aceptar en la app de delivery	☑	-
Descontar stock de ingredientes	☑	-
Imprimir comanda en cocina	☑	-
Calcular comisión de plataforma	☑	-
Marcar como "Listo para recoger"	-	☑ Botón
Asignar repartidor propio	-	☑ Si usas flota propia

Tipos de Delivery

Tipo	Descripción	Quién entrega
PLATFORM_DELIVERY	Rappi/Glovo envía su propio repartidor	La plataforma
SELF_DELIVERY	Usas tu propia flota de repartidores	Tu restaurante
TAKEAWAY	Cliente retira en local	El cliente

Estados del Pedido



CONFIGURACIÓN DE PLATAFORMAS

Desde el Panel de Administración

Datos necesarios por plataforma:

Campo	Descripción	Dónde obtenerlo
Código	Identificador (RAPPI, GLOVO, etc.)	Automático
API Key	Clave para conectar con la plataforma	Panel de Partners de la plataforma
Webhook Secret	Clave para validar webhooks	Panel de Partners de la plataforma
Store ID	ID de tu local en la plataforma	Panel de Partners de la plataforma
Comisión (%)	Porcentaje que cobra la plataforma	Tu contrato con la plataforma

Activar una Plataforma

1. **Ingresar credenciales** (API Key, Webhook Secret, Store ID)
2. **Configurar URL de Webhook** en el panel de la plataforma:

   ```
   https://tu-servidor.com/api/v1/webhooks/rappi
   ```

3. **Activar** la plataforma con el switch

⚠ Protocolo de Seguridad de Margen

¿Qué es?

Es un **Safety Lock** que protege tu negocio de perder dinero.

El Problema:

- Rappi cobra 20-30% de comisión
- Si vendes una pizza a $1,000 y Rappi cobra 25%...
- Te quedan $750, pero la pizza te cuesta $600
- **Ganancia: solo $150** (vs $400 si vendes en local)

La Solución:

El sistema te obliga a revisar los precios antes de activar delivery:

```
┌─────────────────────────────────────────────────────────┐
│                  ⚠ ADVERTENCIA DE MARGEN                │
├─────────────────────────────────────────────────────────┤
│                                                         │
│  Estás por activar el pricing automático para RAPPI.    │
│                                                         │
│  Si no configuraste precios específicos para delivery,  │
│  usará los precios base de tu menú.                     │
│                                                         │
│  Con una comisión del 25%, esto puede resultar en PÉRDIDAS.  │
│                                                         │
│  Opciones:                                              │
│  1. Configurar precios específicos para Rappi (recomendado)  │
│  2. Activar markup automático del 30%                   │
│  3. Aceptar el riesgo y usar precios base               │
│                                                         │
│  ☑ Entiendo el riesgo y acepto usar precios base        │
│                                                         │
│  [Cancelar]                    [Aceptar y Continuar]    │
└─────────────────────────────────────────────────────────┘
```

SISTEMA DE PRECIOS MULTI-CANAL

¿Por qué precios diferentes?

Canal	Comisión	Precio sugerido (Pizza $1000)
Local	0%	$1,000
Rappi	25%	$1,300 (+30%)
Glovo	20%	$1,200 (+20%)
PedidosYa	18%	$1,180 (+18%)

Estrategia de Precios (Smart Fallback)

El sistema busca el precio en este orden:

```
1. ¿Tiene precio específico para esta plataforma?
   └─ SÍ → Usar ese precio
   └─ NO ↓

2. ¿Tiene fallback pricing activado con markup?
   └─ SÍ → Precio base × (1 + markup%)
   └─ NO ↓

3. Usar precio base (⚠ posible pérdida)
```

Ejemplo Práctico

```
Producto: Pizza Muzzarella
Precio base: $1,000

┌──────────────┬──────────────────────┬──────────────────┐
│ Plataforma   │ Configuración        │ Precio final     │
├──────────────┼──────────────────────┼──────────────────┤
│ Local        │ -                    │ $1,000           │
│ Rappi        │ Precio: $1,350       │ $1,350           │
│ Glovo        │ Markup: 25%          │ $1,250           │
│ PedidosYa    │ Sin config (base)    │ $1,000 ⚠         │
└──────────────┴──────────────────────┴──────────────────┘
```

Configurar Precios por Plataforma

```
Menú ▸ Productos ▸ [Producto] ▸ Precios por Canal
```

GESTIÓN DE REPARTIDORES

Para Delivery Propio (SELF_DELIVERY)

Si usas tu propia flota de repartidores:

Agregar Repartidor

```
Menú ▸ Delivery ▸ Repartidores ▸ [Nuevo]

Nombre: Carlos López
Teléfono: +54 11 5555-1234
Vehículo: Moto
Patente: AB123CD
```

Estados del Repartidor

Estado	Significado
🟢 **Disponible**	Puede recibir pedidos
🟡 **En entrega**	Tiene un pedido asignado
🔴 **No disponible**	Fuera de turno o inactivo

Asignar Pedido a Repartidor

1. Abrir pedido con estado "Listo"
2. Click en "Asignar Repartidor"
3. Seleccionar repartidor disponible
4. El sistema:
    - Marca al repartidor como ocupado
    - Cambia estado del pedido a "En camino"
    - Registra hora de salida

REFERENCIA DE API

Endpoints de Webhooks (Entrada)

```
POST /api/v1/webhooks/:platform
```

Recibe webhooks de plataformas externas.

Plataforma	URL
Rappi	`/api/v1/webhooks/rappi`
Glovo	`/api/v1/webhooks/glovo`
PedidosYa	`/api/v1/webhooks/pedidosya`

Endpoints de Plataformas

Método	Endpoint	Descripción
GET	`/api/v1/delivery/platforms`	Listar plataformas
POST	`/api/v1/delivery/platforms`	Crear plataforma
PATCH	`/api/v1/delivery/platforms/:id`	Actualizar
PATCH	`/api/v1/delivery/platforms/:id/toggle`	Activar/desactivar
DELETE	`/api/v1/delivery/platforms/:id`	Eliminar

Endpoints de Repartidores

Método	Endpoint	Descripción
GET	`/api/v1/delivery/drivers`	Listar repartidores
GET	`/api/v1/delivery/drivers/available`	Solo disponibles
POST	`/api/v1/delivery/drivers`	Crear repartidor
PATCH	`/api/v1/delivery/drivers/:id`	Actualizar
PATCH	`/api/v1/delivery/drivers/:id/availability`	Toggle disponibilidad
POST	`/api/v1/delivery/drivers/:id/assign`	Asignar pedido
POST	`/api/v1/delivery/drivers/:id/release`	Liberar repartidor

Endpoints de Órdenes Delivery

Método	Endpoint	Descripción
GET	`/api/v1/delivery/orders`	Listar pedidos delivery
GET	`/api/v1/delivery/orders?status=OPEN`	Filtrar por estado

RESUMEN DE COMPORTAMIENTO

¿Qué pasa cuando llega un pedido de Rappi?

1. ⏱ **0-100ms**: Webhook recibido, validado y encolado
2. ⏱ **100-500ms**: Worker procesa y crea orden
3. ⏱ **500-1000ms**: Stock descontado, comanda impresa
4. ⏱ **1-2s**: Pedido aceptado en Rappi automáticamente
5. 🔔 **Notificación** en pantalla de cocina/POS

¿Qué errores pueden ocurrir?

Error	Causa	Solución
Firma inválida	Webhook Secret incorrecto	Verificar secret en config
Producto no encontrado	SKU no existe	Mapear SKUs en precios por canal
Stock insuficiente	No hay ingredientes	Igual se crea, pero con advertencia
Timeout de API	Rappi no responde	Se reintenta automáticamente (10 veces)

Documento generado: 2026-01-19 - Control Gastronómico V2