

API de GOURMETIA

Dossier técnico para integradores

Conecta el software de contabilidad, conciliación o análisis con el TPV y los pedidos online de un restaurante: API REST de solo lectura y webhooks firmados.

Versión de la API: **v1 · 2026-10-30**

Documentación siempre actualizada: **developers.gourmetia.app**

Base URL: **<https://api.gourmetia.app/v1>**

Soporte técnico: **clientes@gourmetia.app**

Conecta tu software con el TPV y los pedidos online de un restaurante

API REST de **solo lectura** y **webhooks firmados** para programas de contabilidad, conciliación bancaria y análisis. Tickets y facturas, cobros, cierres de caja, pedidos online y un resumen diario listo para el asiento contable.

BASE URL

**api.gourmetia.app/
v1**

FORMATO

JSON · UTF-8

IMPORTE

**Céntimos (enteros)
· EUR**

ACCESO

**Solo lectura · clave
por local**

Qué ofrece

API REST (consultar)

Tu sistema pide los datos cuando los necesita: «dame los tickets de ayer», «dame el cierre de caja del día 29». Ideal para sincronizaciones periódicas y para recuperar histórico.

Webhooks (recibir avisos)

GOURMETIA avisa a tu sistema en el momento en que ocurre algo: un ticket emitido, un cierre de caja, un pedido online nuevo. Cada aviso va firmado para que compruebes que viene de nosotros.

Lo que no hace: la API no crea ni modifica nada en GOURMETIA, no expone datos personales de los clientes del restaurante y no emite facturas. La emisión de tickets y facturas (y su registro VERI*FACTU) es responsabilidad del TPV de GOURMETIA; tu sistema las recibe para contabilizarlas.

Empezar en 5 minutos

1 Pide una clave de pruebas al restaurante. Un administrador del local la crea en `restaurantes.gourmetia.app` → Cuenta → **Integraciones · API** → Crear clave → **Pruebas**. La clave se muestra una sola vez.

2 Haz tu primera llamada. Comprueba a qué local corresponde la clave:

```
curl https://api.gourmetia.app/v1/location \
-H "Authorization: Bearer gm_test_..."
```

3 Lee los tickets de hoy (fechas en hora de Madrid):

```
curl "https://api.gourmetia.app/v1/invoices?from=2026-10-30" \
-H "Authorization: Bearer gm_test_..."
```

4 Monta la integración con datos de pruebas. La clave de pruebas devuelve un restaurante de demostración con ventas nuevas cada hora, cierres de caja diarios y pedidos online.

5 Pasa a real. Cuando esté validado, el restaurante crea una clave **Real**. Es la misma API: solo cambia la clave.

Autenticación

Cada petición lleva la clave de API en la cabecera `Authorization` :

```
Authorization: Bearer gm_live_3f9c...
```

PREFIJO	ENTORNO	DATOS
gm_test_	Pruebas	Restaurante de demostración «Casa Lumbre · DEMO» (datos ficticios)
gm_live_	Real	El local del restaurante que creó la clave

- Una clave da acceso a **un único local** y solo en lectura.
- Guárdala en tu servidor (variable de entorno o gestor de secretos). **Nunca** en una web, una app móvil o una URL.
- La API es **solo de servidor a servidor**: no envía cabeceras CORS, así que un navegador no puede llamarla (esto protege la clave).
- GOURMETIA solo guarda una huella (SHA-256) de la clave: si se pierde, se crea otra.
- El restaurante puede **revocarla** en cualquier momento; a partir de ese instante la API responde `401 invalid_api_key`.
- Usar una clave implica aceptar las Condiciones de uso de la API (el restaurante las acepta al crearla). Entre el restaurante y el integrador debe firmarse un contrato de encargo del tratamiento (modelo disponible).

Entornos: pruebas y real

Pruebas **sandbox**

- Restaurante ficticio con NIF `B000000000` y serie `DEM`.
- Ventas nuevas cada hora en horario de servicio (13–17 h y 20–24 h, Madrid) y un cierre de caja diario hacia las 4:00.
- Pedidos online de ejemplo a lo largo del día.
- Los webhooks de pruebas reciben esos eventos (`livemode: false`).

Real **producción**

- Datos reales del local que creó la clave.
- Mismos endpoints, mismos objetos.
- Los webhooks reales reciben los eventos del local (`livemode: true`).
- Respuestas con cabecera `GOURMETIA-Mode: live`.

Convenciones

TEMA	REGLA
Importes	Enteros en céntimos (<code>1250</code> = 12,50 €). Moneda <code>EUR</code> . Nunca decimales en coma flotante.
Impuestos	Los precios de hostelería incluyen IVA. En cada documento: <code>total_cents = subtotal_cents + tax_cents</code> (base + cuota). Tipos en porcentaje (<code>10</code> = 10 %), desglosados en <code>tax_breakdown</code> .
Fechas	ISO 8601 en UTC (<code>2026-10-30T21:41:18Z</code>). El día contable es el de Madrid (<code>Europe/Madrid</code>).
Identificadores	UUID. Estables: úsalos para no duplicar registros (idempotencia).
Campos nuevos	Podemos añadir campos sin aviso; tu código debe ignorar los que no conozca. Nunca quitamos ni cambiamos el significado de un campo dentro de <code>v1</code> .
Nulos	Un campo sin valor viene como <code>null</code> , no se omite.

Paginación y filtros

Los listados devuelven los objetos en **orden cronológico ascendente**, en páginas de hasta 500. Así puedes sincronizar de forma incremental: guarda el último `next_cursor` y continúa desde ahí.

```
{
  "object": "list",
  "data": [ ... ],
  "has_more": true,
  "next_cursor": "MjAyNi0xMC0zMCAyMT00MT0xOC4zNTUy..."
}
```

PARÁMETRO	DESCRIPCIÓN
<code>from</code>	Desde (incluido). <code>2026-10-30</code> = 00:00 de ese día en Madrid, o un instante ISO <code>2026-10-30T08:00:00Z</code> .
<code>to</code>	Hasta (no incluido), mismo formato. <code>from=2026-10-30&to=2026-10-31</code> = todo el día 30.
<code>limit</code>	Tamaño de página, 1-500 (100 por defecto).
<code>cursor</code>	El <code>next_cursor</code> de la página anterior.
<code>status</code>	Solo en cierres de caja y pedidos online (ver cada recurso).

```
# Recorrer todos los tickets de octubre
cursor=""
while ;; do
  r=$(curl -s "https://api.gourmetia.app/v1/invoices?from=2026-10-01&to=2026-11-01&limit=500&cursor=$cursor" \
    -H "Authorization: Bearer $GOURMETIA_KEY")
  echo "$r" | jq -c '.data[]'
  [ "$(echo "$r" | jq -r .has_more)" = "true" ] || break
  cursor=$(echo "$r" | jq -r .next_cursor)
done
```

Errores y límites

Los errores usan códigos HTTP estándar y un cuerpo con un código estable para programar contra él:

```
{
  "error": {
    "code": "invalid_parameter",
    "message": "«limit» debe ser un número entre 1 y 500.",
    "docs": "https://developers.gourmetia.app/#errores"
  }
}
```

HTTP	CODE	CUÁNDO
400	<code>invalid_parameter</code>	Un parámetro no es válido (fecha, límite, cursor, estado).
401	<code>invalid_api_key</code>	Falta la clave, no es válida o ha sido revocada.
404	<code>not_found</code>	La ruta no existe o el objeto no pertenece al local de la clave.
405	<code>method_not_allowed</code>	La API es de solo lectura: solo <code>GET</code> .
429	<code>rate_limited</code>	Más de 120 peticiones por minuto con la misma clave, o demasiados intentos con claves no válidas desde la misma IP. Espera los segundos de <code>Retry-After</code> .
500	<code>internal_error</code>	Error nuestro. Reintenta con espera creciente.

Cada respuesta incluye `X-RateLimit-Limit`, `X-RateLimit-Remaining`, `GOURMETIA-Mode` (`test` / `live`) y `GOURMETIA-Version`.

Local

GET `/v1/location`

Datos fiscales del local al que pertenece la clave. Útil para comprobar que la clave es la correcta.

```
{
  "id": "9ff0eb2b-22a6-40fb-94c5-4ed135b78130",
  "object": "location",
  "name": "Casa Lumbre · DEMO",
  "slug": "gourmetia-demo",
  "legal_name": "CASA LUMBRE RESTAURACION SL (DEMO)",
  "tax_id": "B000000000",
  "address": "Calle Ficticia 1, 28001 Madrid",
  "invoice_series": "DEM",
  "timezone": "Europe/Madrid",
  "currency": "EUR",
  "livemode": false
}
```

Tickets y facturas

GET /v1/invoices

GET /v1/invoices/{id}

Cada cobro en el TPV genera un registro de facturación **inalterable**, numerado de forma correlativa por serie y encadenado con huella (VERI*FACTU). Filtra por fecha de expedición (`issued_at`).

CAMPO	DESCRIPCIÓN
<code>number</code>	Número completo de serie (<code>DEM26-000004</code>): serie + año + correlativo.
<code>series</code> · <code>sequence</code>	Serie y número correlativo dentro de ella. Sin huecos: útil para control de numeración.
<code>type</code>	<code>F2</code> factura simplificada (ticket) · <code>F1</code> factura completa · <code>R1</code> – <code>R5</code> rectificativas.
<code>issued_at</code>	Fecha y hora de expedición (UTC).
<code>issuer</code>	<code>tax_id</code> (NIF) y <code>legal_name</code> (razón social) del emisor.
<code>subtotal_cents</code> · <code>tax_cents</code> · <code>total_cents</code>	Base imponible, cuota de IVA y total.
<code>tax_breakdown[]</code>	Por tipo de IVA: <code>rate</code> , <code>base_cents</code> , <code>tax_cents</code> , <code>total_cents</code> .
<code>lines[]</code>	<code>description</code> , <code>variant</code> , <code>quantity</code> , <code>unit_price_cents</code> , <code>tax_rate</code> , <code>total_cents</code> (IVA incluido).
<code>payments[]</code>	Cobros del ticket (ver Cobros). Un ticket puede pagarse con varios métodos.
<code>tips_cents</code>	Propinas del ticket (no forman parte del total facturado).
<code>table</code> · <code>diners</code> · <code>staff</code>	Mesa, comensales y camarero.
<code>cash_closing_id</code>	Cierre de caja al que pertenece el cobro.
<code>verifactu</code>	<code>hash</code> , <code>previous_hash</code> , <code>generated_at</code> y <code>aeat_status</code> (ver VERI*FACTU).

```
{
  "id": "9f23f3aa-4b24-4c72-9b39-98d175170274",
  "object": "invoice",
  "number": "DEM26-000004",
  "series": "DEM",
  "sequence": 4,
  "type": "F2",
  "issued_at": "2026-10-30T21:41:18Z",
  "issuer": { "tax_id": "B00000000", "legal_name": "CASA LUMBRE RESTAURACION SL (DEMO)" },
  "currency": "EUR",
  "subtotal_cents": 9636,
  "tax_cents": 964,
  "total_cents": 10600,
  "tax_breakdown": [
    { "rate": 10, "base_cents": 9636, "tax_cents": 964, "total_cents": 10600 }
  ],
  "lines": [
    { "description": "Pulpo a la brasa", "variant": null, "quantity": 2,
      "unit_price_cents": 1850, "tax_rate": 10, "total_cents": 3700 },
    { "description": "Secreto ibérico", "variant": null, "quantity": 2,
      "unit_price_cents": 1900, "tax_rate": 10, "total_cents": 3800 },
    ...
  ],
  "payments": [
    { "id": "14a2da81-d566-424b-9776-41dd93efa502", "method": "card",
      "amount_cents": 10600, "tip_cents": 400, "charged_cents": 11000 }
  ],
  "tips_cents": 400,
  "table": "S4",
  "diners": 3,
  "staff": "Inés",
  "cash_closing_id": "72e7734f-c2cb-433b-aaa6-4378c82f7cb5",
  "verifactu": {
    "hash": "26473DC7DB4287CF6AA13D30EDA77F74095BAFBF0088EEF9C4AC32D4931D12B1",
    "previous_hash": "4A82DABF2A24168D1D6D6523426C1F0B596AA5B1A5AECBAEA8EBA12B765DC580",
    "generated_at": "2026-10-30T22:41:18+01:00",
    "aeat_status": "pending"
  }
}
```

Cobros

GET /v1/payments

GET /v1/payments/{id}

Cada cobro individual del TPV. Es el recurso para **conciliar** con el datáfono, el banco y el efectivo.

CAMPO	DESCRIPCIÓN
<code>method</code>	<code>card</code> tarjeta · <code>cash</code> efectivo · <code>other</code> otros (vales, transferencia...).
<code>amount_cents</code>	Importe aplicado al ticket (sin propina).
<code>tip_cents</code>	Propina cobrada junto al pago.
<code>charged_cents</code>	<code>amount_cents + tip_cents</code> : lo que se cobró de verdad (lo que verás en el datáfono).
<code>tendered_cents</code> · <code>change_cents</code>	Solo efectivo: entregado por el cliente y cambio devuelto.
<code>invoice_id</code> · <code>invoice_number</code>	Ticket al que pertenece.
<code>cash_closing_id</code>	Cierre de caja en el que entra.

```
{
  "id": "14a2da81-d566-424b-9776-41dd93efa502",
  "object": "payment",
  "created_at": "2026-10-30T21:41:18Z",
  "method": "card",
  "amount_cents": 10600,
  "tip_cents": 400,
  "charged_cents": 11000,
  "tendered_cents": null,
  "change_cents": null,
  "invoice_id": "9f23f3aa-4b24-4c72-9b39-98d175170274",
  "invoice_number": "DEM26-000004",
  "cash_closing_id": "72e7734f-c2cb-433b-aaa6-4378c82f7cb5"
}
```

Cierres de caja

GET `/v1/cash-closings ?status=open|closed`

GET `/v1/cash-closings/{id}`

El cierre Z de cada turno o jornada: totales por método de pago, desglose de IVA, rango de tickets y el arqueo de efectivo. Una caja `open` devuelve los totales acumulados hasta ese momento; al cerrarse quedan fijos. Se filtra por fecha de apertura (`opened_at`).

CAMPO	DESCRIPCIÓN
<code>status</code>	<code>open</code> o <code>closed</code> .
<code>opened_at</code> · <code>closed_at</code> · <code>opened_by</code> · <code>closed_by</code>	Apertura y cierre, y quién los hizo.
<code>tickets</code> · <code>diners</code> · <code>sales_cents</code>	Número de tickets, comensales y ventas totales (IVA incluido).
<code>payments_by_method</code>	<code>{ cash, card, other }</code> en céntimos (sin propinas).
<code>tips_cents</code>	Propinas del cierre.
<code>tax_breakdown[]</code>	Base y cuota por tipo de IVA.
<code>first_invoice</code> · <code>last_invoice</code>	Primer y último ticket del cierre.
<code>opening_cash_cents</code>	Fondo de caja inicial.
<code>expected_cash_cents</code> · <code>counted_cash_cents</code> · <code>difference_cents</code>	Efectivo esperado, contado y descuadre (contado – esperado).

```
{
  "id": "72e7734f-c2cb-433b-aaa6-4378c82f7cb5",
  "object": "cash_closing",
  "status": "closed",
  "opened_at": "2026-10-30T10:16:45Z",
  "closed_at": "2026-10-31T03:00:12Z",
  "opened_by": "Laura",
  "closed_by": "Laura",
  "currency": "EUR",
  "tickets": 8,
  "diners": 25,
  "sales_cents": 51350,
  "payments_by_method": { "cash": 9050, "card": 42300, "other": 0 },
  "tips_cents": 550,
  "tax_breakdown": [ { "rate": 10, "base_cents": 46682, "tax_cents": 4668, "total_cents": 51350
} ],
  "first_invoice": "DEM26-000001",
  "last_invoice": "DEM26-000008",
  "opening_cash_cents": 15000,
  "expected_cash_cents": 24050,
  "counted_cash_cents": 24050,
  "difference_cents": 0,
  "notes": null
}
```

Pedidos online

GET /v1/online-orders ?status=...

GET /v1/online-orders/{id}

Pedidos de la tienda online del restaurante (recogida y a domicilio). Sin datos personales del cliente final. Se filtra por fecha de creación.

CAMPO	DESCRIPCIÓN
code	Código visible del pedido.
status	pending (sin aceptar) → accepted → ready → out (en reparto) → delivered / completed ; o cancelled .
fulfillment	pickup recogida · delivery a domicilio.
payment.mode · payment.status	online (pagado con tarjeta en la web: paid / unpaid) u on_arrival (se paga al recibir).
items[]	Líneas con precio unitario, cantidad, tipo de IVA y total (IVA incluido).
subtotal_cents · discount_cents · coupon_code	Suma de productos, descuento aplicado y cupón.
delivery_fee_cents	Gastos de envío cobrados al cliente.
total_cents	Total cobrado al cliente.
platform_fee_cents	Comisión de la plataforma en pagos online.
courier	Reparto externo: provider , fee_cents (coste del rider), status . null si reparte el propio local.

Importante: hoy un pedido online **no** genera por sí mismo un ticket del TPV. Si el restaurante lo cobra en caja, aparecerá además como ticket en /v1/invoices ; tenlo en cuenta para no contabilizarlo dos veces.

```
{
  "id": "c882bbf7-fda7-4cc8-bd37-6eec7e108160",
  "object": "online_order",
  "code": "CL-1041",
  "status": "delivered",
  "fulfillment": "pickup",
  "created_at": "2026-10-30T18:17:00Z",
  "scheduled_for": null,
  "status_changed_at": "2026-10-30T18:52:20Z",
  "store_location": { "id": "ed85ae42-ac9e-4df1-8a46-9308d4126d19", "name": "Casa Lumbre" },
  "currency": "EUR",
  "payment": { "mode": "on_arrival", "status": "on_arrival" },
  "items": [
    { "description": "Pulpo a la brasa", "quantity": 1, "unit_price_cents": 1850, "tax_rate": 10, "total_cents": 1850 },
    { "description": "Tarta de queso cremosa", "quantity": 1, "unit_price_cents": 700, "tax_rate": 10, "total_cents": 700 },
    { "description": "Agua mineral", "quantity": 1, "unit_price_cents": 250, "tax_rate": 10, "total_cents": 250 }
  ],
  "subtotal_cents": 2800,
  "discount_cents": 0,
  "coupon_code": null,
  "delivery_fee_cents": 0,
  "total_cents": 2800,
  "platform_fee_cents": 0,
  "courier": null
}
```

Resumen diario

GET /v1/daily-summary?date=2026-10-30

Todo lo de un día contable (00:00–24:00 de Madrid) en una sola llamada: lo que normalmente se lleva al **asiento diario**. Sin `date`, devuelve el día de hoy (todavía abierto).

```
{
  "object": "daily_summary",
  "date": "2026-10-30",
  "timezone": "Europe/Madrid",
  "currency": "EUR",
  "pos": {
    "tickets": 16,
    "subtotal_cents": 111000,
    "tax_cents": 11100,
    "total_cents": 122100,
    "tax_breakdown": [ { "rate": 10, "base_cents": 111000, "tax_cents": 11100, "total_cents":
122100 } ],
    "payments_by_method": { "cash": 25350, "card": 96750, "other": 0 },
    "tips_cents": 1200,
    "card_charged_cents": 97950,
    "first_invoice": "DEM26-000001",
    "last_invoice": "DEM26-000016"
  },
  "cash_closings": [ /* cierres cerrados ese día (objeto completo) */ ],
  "online_orders": {
    "orders": 5,
    "subtotal_cents": 17700,
    "discount_cents": 0,
    "delivery_fee_cents": 598,
    "total_cents": 18298,
    "by_payment": { "online": 0, "on_arrival": 18298 },
    "by_status": { "accepted": 1, "ready": 1, "out": 1, "delivered": 2 }
  }
}
```

`online_orders` suma los pedidos aceptados (excluye `pending` y `cancelled`); `by_status` cuenta todos. `card_charged_cents` incluye propinas: es la cifra a comparar con la liquidación del datáfono.

Webhooks

El restaurante da de alta la URL de tu sistema en **Integraciones · API → Añadir webhook** (con datos de pruebas o reales) y elige los eventos. GOURMETIA hace un `POST` con JSON a esa URL cada vez que ocurre uno.

Requisitos de la URL: `https://`, un **dominio público** (no se admiten IPs, `localhost` ni redes internas) y puerto 443 u 8443. No se siguen redirecciones.

EVENTO	CUÁNDO	DATA
<code>invoice.created</code>	Se emite un ticket o factura en el TPV.	Objeto invoice
<code>cash_closing.closed</code>	Se cierra una caja (cierre Z).	Objeto cash_closing
<code>online_order.created</code>	Entra un pedido online.	Objeto online_order
<code>online_order.status_changed</code>	Un pedido online cambia de estado.	<code>online_order</code> + <code>previous_status</code>
<code>ping</code>	El restaurante pulsa «Enviar prueba».	<code>{ message }</code>

Qué recibes

```
POST /webhooks/gourmetia HTTP/1.1
Content-Type: application/json
User-Agent: GOURMETIA-Webhooks/1.0
GOURMETIA-Event: invoice.created
GOURMETIA-Event-Id: evt_5b0c7e2f4a7d4f0f9a3c2a8d1e6b9f10
GOURMETIA-Delivery: 1842
GOURMETIA-Attempt: 1
GOURMETIA-Signature: t=1793396700,v1=6f1c0a...e93b

{
  "id": "evt_5b0c7e2f4a7d4f0f9a3c2a8d1e6b9f10",
  "object": "event",
  "type": "invoice.created",
  "created_at": "2026-10-30T21:45:00Z",
  "livemode": true,
  "location": { "id": "...", "name": "Nombre del local" },
  "data": { /* objeto invoice completo, igual que en la API */ }
}
```

Cómo responder

- Devuelve un **2xx en menos de 10 segundos**. Guarda el evento y procésalo después (cola), no dentro de la petición.
- **Idempotencia**: un mismo evento puede llegar más de una vez (reintentos). Usa `id` del evento o el `id` del objeto para no duplicar.
- **Orden**: no está garantizado. Si necesitas el estado más reciente, consulta la API con el `id`.
- Los webhooks complementan a la API, no la sustituyen: haz una sincronización periódica (por ejemplo, el resumen diario) para cubrir cualquier aviso perdido.

Verificar la firma

La cabecera `GOURMETIA-Signature` contiene `t` (marca de tiempo Unix) y `v1` = **HMAC-SHA256** en hexadecimal de `t + "." + cuerpo`, con el **secreto del webhook** (`whsec_...`, visible para el restaurante en la ficha del webhook). Usa el cuerpo **tal cual llega**, sin volver a serializarlo, y rechaza marcas de tiempo de más de 5 minutos.

Node.js

```
import crypto from "node:crypto";

// rawBody: el cuerpo exacto recibido (string o Buffer), antes de JSON.parse
export function verifyGourmetia(rawBody, header, secret, toleranceSec = 300) {
  const p = Object.fromEntries(String(header || "").split(",").map(kv => kv.split("=")));
  const t = Number(p.t);
  if (!t || Math.abs(Date.now() / 1000 - t) > toleranceSec) return false;
  const expected = crypto.createHmac("sha256", secret).update(`${t}.${rawBody}`).digest("hex");
  const a = Buffer.from(expected), b = Buffer.from(p.v1 || "");
  return a.length === b.length && crypto.timingSafeEqual(a, b);
}
```

Python

```
import hmac, hashlib, time

def verify_gourmetia(raw_body: bytes, header: str, secret: str, tolerance: int = 300) → bool:
    parts = dict(kv.split("=", 1) for kv in (header or "").split(",") if "=" in kv)
    t = int(parts.get("t", "0"))
    if not t or abs(time.time() - t) > tolerance:
        return False
    expected = hmac.new(secret.encode(), f"{t}.".encode() + raw_body, hashlib.sha256).hexdigest()
    return hmac.compare_digest(expected, parts.get("v1", ""))
```

PHP

```
function gourmetia_verify(string $raw, string $header, string $secret, int $tol = 300): bool {
    $p = [];
    foreach (explode(',', $header) as $kv) {
        [$k, $v] = array_pad(explode('=', $kv, 2), 2, '');
        $p[$k] = $v;
    }
    $t = (int)($p['t'] ?? 0);
    if (!$t || abs(time() - $t) > $tol) return false;
    $expected = hash_hmac('sha256', $t . '.' . $raw, $secret);
    return hash_equals($expected, $p['v1'] ?? '');
}

// $raw = file_get_contents('php://input');
```

Reintentos

Si tu URL no responde 2xx (o tarda más de 10 s), reintentamos con espera creciente: **1 min, 5 min, 15 min, 1 h, 3 h, 6 h y 12 h**. Son 8 intentos en unas 22 horas; después el envío queda como fallido. El restaurante ve el estado de cada envío en la ficha del webhook y puede enviar una prueba cuando quiera.

Guía: contabilidad y conciliación

Recetas habituales para programas de contabilidad de restaurantes. Las cuentas son orientativas (Plan General Contable); el criterio final es del asesor del restaurante.

1. Asiento diario de ventas

Con `GET /v1/daily-summary?date=...` del día anterior:

CUENTA (ORIENTATIVA)	DEBE	HABER	DATO DE LA API
570 Caja (efectivo)	✓		<code>pos.payments_by_method.cash</code>
572 Bancos / cuenta puente de tarjeta	✓		<code>pos.payments_by_method.card</code>
700 Ventas (por tipo de IVA)		✓	<code>pos.tax_breakdown[].base_cents</code>
477 HP IVA repercutido (por tipo)		✓	<code>pos.tax_breakdown[].tax_cents</code>

Las propinas (`tips_cents`) no son ventas del local: llévalas a la cuenta que indique el asesor.

2. Conciliación de tarjeta

Compara `pos.card_charged_cents` (importe + propinas con tarjeta) con la liquidación diaria del datáfono o el abono del banco. La diferencia suele ser la comisión bancaria. Para localizar un cobro concreto, usa `/v1/payments` con `from / to` alrededor de su hora y filtra por `method` y `charged_cents`.

3. Arqueo de efectivo

Cada `cash_closing` trae el efectivo esperado, el contado y el descuadre (`difference_cents`). Un descuadre distinto de cero es una incidencia a revisar.

4. Control de numeración

`sequence` es correlativo por `series`. Un hueco en la secuencia indica que falta un registro en tu sistema: vuelve a pedir ese rango a `/v1/invoices`.

5. Pedidos online

Distingue `payment.mode` : `online` (cobrado por pasarela; la comisión va en `platform_fee_cents`) frente a `on_arrival` (se cobra al entregar). Los gastos de envío vienen aparte en `delivery_fee_cents` .

VERI*FACTU

El TPV de GOURMETIA es el **sistema informático de facturación** del restaurante: expide los tickets y facturas, los numera, los registra de forma inalterable y los encadena con una huella (SHA-256) según el Reglamento de sistemas de facturación (VERI*FACTU).

- Tu programa **recibe y contabiliza** esos registros; **no debe volver a emitirlos** como facturas propias.
- Los registros no se modifican nunca. Una corrección llegará como una nueva factura rectificativa (`type` R1-R5) que referencia a la original.
- `verifactu.hash` y `verifactu.previous_hash` permiten comprobar la cadena: el `previous_hash` de un registro es el `hash` del anterior del mismo emisor.
- `verifactu.aeat_status` indica el estado de remisión a la Agencia Tributaria (`pending` mientras no se haya remitido).

Seguridad y privacidad

- **Solo lectura.** Ninguna llamada puede crear, cambiar o borrar datos.
- **Una clave por local**, creada y revocada por el propio restaurante. Guardamos solo su huella SHA-256.
- **Solo HTTPS**, tanto la API como los webhooks (las URLs de webhook deben ser `https://`).
- **Sin datos personales** de los clientes del restaurante (nombre, teléfono, email o dirección no se exponen).
- **Webhooks firmados** con un secreto propio de cada URL, que el restaurante puede regenerar en cualquier momento.
- **Registro de accesos:** cada petición queda anotada (hora, IP y datos pedidos) durante 6 meses, y el restaurante lo ve en la ficha de cada clave.
- **Freno a ataques:** límite de peticiones por clave y bloqueo temporal de las IP que prueban claves no válidas.
- El restaurante es el responsable de los datos y decide a quién da acceso. Si tu sistema trata datos de su negocio, formalizad entre vosotros el contrato de encargo del tratamiento que corresponda (RGPD).
- Sin acceso al código ni a la base de datos de GOURMETIA: la integración es exclusivamente a través de esta API y de los webhooks.
- La API, su documentación y la marca GOURMETIA® son propiedad de OPIO AGENCIA. Su uso se rige por las Condiciones de uso de la API, que prohíben ceder la clave, usar los datos para otros fines y aplicar ingeniería inversa o crear productos que repliquen GOURMETIA.

Versiones y soporte

La versión va en la ruta (`/v1`). Dentro de `v1` solo hacemos cambios compatibles (añadir campos, eventos o endpoints). Un cambio incompatible sería `/v2` , con convivencia y aviso previo.

FECHA	CAMBIO
2026-10-30	Primera versión pública: local, tickets y facturas, cobros, cierres de caja, pedidos online, resumen diario y webhooks.

Soporte técnico: `clientes@gourmetia.app`. Indica el local con el que integras y, si es un error, la hora, el endpoint y el código de respuesta (nunca envíes la clave completa: basta con su prefijo, p. ej. `gm_live_3f9c...`).