Sphinx

MANUAL PARA INTEGRADORES

API de Sphinx versión 1

Cómo un sistema externo —una tienda en línea, un marketplace, un sistema de despacho o un tablero de gestión— lee el catálogo de Sphinx y le registra ventas. Reemplaza a la interfazWEB.

Catálogo · Ventas · Consultas · Clientes · Claves con alcance

REST · JSON · HTTPS · RFC 9457

Septiembre de 2026

Contenido

1 Introducción

1.1 Qué es y para quién

La **API de Sphinx** es la puerta por la que un sistema externo conversa con el ERP Sphinx de una empresa: lee su catálogo —productos, precios, stock, fotos, marcas y familias—, crea pedidos que reservan stock, cobra con la pasarela de pago de Sphinx y consulta boletas y despachos.

Es la misma API que usa la tienda en línea de Sphinx: todo lo que aquí se describe está probado a diario en tiendas en producción.

Este manual está escrito para quien programa la integración. Cada servicio se documenta con su ruta, sus parámetros, los campos de su respuesta y un ejemplo.

> **Una API, un ERP.** Cada empresa tiene su propio Sphinx, en su propio servidor. La dirección del servidor y el nombre de la base se los da la empresa; la clave, también (ver 2.2).

1.2 Si viene de la interfazWEB

La interfazWEB (`/interfazWEB/…`) va a quedar **obsoleta**: sigue funcionando mientras las integraciones que la usan se cambian a esta API, y después se apaga. Lo que cambia:

- **Códigos HTTP de verdad.** Un error ya no llega como HTTP 200 con un código adentro: llega con su código HTTP (401, 404, 422…) y un cuerpo estándar (3). El número de error que ya conoce viaja igual, en `codigo`.
- **Sin sobre.** La respuesta es directamente el dato, sin `{codigo, mensaje, resultado}` alrededor.
- **Clave propia por sistema**, con alcance, que la empresa crea y revoca por su cuenta (2.2).
- **El producto viaja por partes.** Ficha, precios, stock, disponibilidad y fotos se piden por separado, porque cambian a ritmos distintos: un cambio de precio ya no obliga a bajar de nuevo la descripción y las fotos.
- **Fotos sólo por URL**, nunca en base64.
- **Caché con ETag:** si nada cambió, la respuesta es un 304 sin cuerpo (2.3).
- **Los productos se identifican por `idProducto`**; el código (SKU) viaja en la ficha.

La tabla de equivalencias servicio por servicio está en 10.

2 Cómo conectarse

2.1 La dirección

```
https://<servidor>/api/v1/<base>/<recurso>[/…][?parámetro=valor]
```

Parte	Qué es
`<servidor>`	El nombre del Sphinx de la empresa, el mismo con que sus usuarios entran al sistema. Siempre por **HTTPS**.
`<base>`	La base de datos de la empresa en ese servidor. Es obligatoria: la clave es de esa base y no abre otra.
`<recurso>`	En minúsculas: `productos`, `fichas`, `pedidos` …

Las respuestas son JSON en UTF-8. Los cuerpos que se envían, también, con `Content-Type: application/json`.

2.2 La clave y su alcance

Cada sistema que se conecta usa su propia clave. La crea la empresa en su Sphinx, en **Integraciones > API**, y se la entrega a usted: tiene la forma `spx_` seguida de 43 caracteres. Se envía en cada llamada:

```
Authorization: Bearer spx_Xy7…
```

Al crearla, la empresa le da un **alcance**: lo que ese sistema puede hacer.

Alcance	Abre
Catálogo	`productos`, `fichas`, `precios`, `stock`, `disponibilidad`, `fotos`, `fusiones`, `maestros`, `contenido`, `arma-tu-pc`
Ventas	`pedidos`, `pagos`
Consultas	`boletas`, `ordenes-servicio`, `despacho`
Clientes	`cuenta`, `contactos`, `suscripciones`
Reportes	`reportes` (estadísticas, cuadratura, diario), `usuarios`

`estado` lo abre cualquier clave, para probarla. Una ruta fuera del alcance contesta **403**; en ese caso pídale a la empresa una clave con el alcance que falta.

La clave es de la empresa. Debe vivir en el servidor de su integración y nunca en el navegador ni en una aplicación que se instale en el teléfono del cliente. La empresa ve en Sphinx cuándo y desde qué IP se usó por última vez, y la puede revocar en cualquier momento: desde ahí, cada llamada con ella contesta 401.

2.3 Caché con ETag

Toda respuesta de un GET trae un encabezado `ETag`. Si la próxima vez lo envía en `If-None-Match` y nada cambió, la respuesta es **304**, sin cuerpo. Es lo que permite consultar el stock o los precios de todo el catálogo cada pocos minutos sin mover nada cuando no hay cambios.

```
GET /api/v1/miempresa/stock
ETag: "08cc9e72ab218e5112146e57080434a52"

GET /api/v1/miempresa/stock
If-None-Match: "08cc9e72ab218e5112146e57080434a52"
→ 304 Not Modified
```

2.4 Límites

Límite	Valor
Peticiones por IP	30 por segundo, con ráfagas de hasta 60. Lo que pase de ahí recibe **HTTP 429**: espere unos segundos y reintente.
Tiempo de una petición	300 segundos.
Fichas por página	100 por omisión, hasta 200 con `limite`.

2.5 Compatibilidad

La versión va en la ruta (`/v1`). Dentro de la v1 **sólo se agrega**: rutas nuevas, campos nuevos en una respuesta, parámetros opcionales. Nunca se quita ni se renombra un campo, ni cambia su tipo o su significado. Su programa debe **ignorar los campos que no conoce**: así la actualización de Sphinx no lo rompe. Un cambio incompatible sería una ruta nueva en `/v2`, y la v1 seguiría funcionando.

3 Errores

Un error llega con su código HTTP y un cuerpo `application/problem+json` (RFC 9457):

```
HTTP/1.1 422
{ "type": "about:blank", "title": "El ERP no pudo completar el pedido", "status": 422,
  "detail": "Falta Stock de los productos : NB-001", "instance": "/api/v1/miempresa/pedidos",
  "codigo": 100 }
```

Campo	Descripción
`status`	El mismo código HTTP.
`title`	El tipo de error.
`detail`	Qué pasó, en palabras que se le pueden mostrar al usuario.
`instance`	La ruta que falló.
`codigo`	Sólo en un 422: el número de error, el mismo que contestaba la interfazWEB.
`resultado`	Sólo en un 422 que trae un dato: el comprobante de un pago rechazado (5.4).

HTTP	Cuándo	Qué hacer
200	Todo bien.	
304	Nada cambió desde el ETag enviado.	Usar lo que ya tiene.
400	Un parámetro o un cuerpo que no se puede leer.	Corregir la llamada.
401	Falta la clave, no es de esa base, o fue revocada.	Revisar la clave con la empresa.
403	La clave no tiene el alcance de esa ruta.	Pedir una clave con ese alcance.
404	La base no existe, o lo pedido no existe.	Revisar la ruta.
409	La empresa todavía no tiene publicado su sitio de fotos, así que las imágenes no tendrían dirección.	Avisar a la empresa.
422	Una regla de negocio: sin stock, un dato que falta, un monto que no cuadra.	Mostrar `detail`. Reintentar igual no cambia nada.
429	Demasiadas peticiones.	Esperar unos segundos.
500	Error del ERP; quedó registrado.	Reintentar más tarde.
502, 503, 504	El servidor se está actualizando o la base no se pudo abrir.	Reintentar en un minuto.

4 Catálogo

Todos son `GET` , llevan ETag y piden el alcance **Catálogo**. Un producto aparece en el catálogo cuando está marcado para la web y su familia también.

4.1 estado

Con qué ERP está hablando: sirve para probar la dirección, la base y la clave sin tocar datos. Lo abre cualquier clave.

Ruta	`GET` `/api/v1/<base>/estado`

Campo	Tipo	Descripción
base	String	La base que atendió.
version	String	La versión del ERP.
hora	String	La hora del servidor, en ISO-8601.

```
{ "base": "miempresa", "version": "202609.166", "hora": "2026-09-27T22:38:47.21-03:00" }
```

4.2 productos

Todos los productos publicados, con la fecha de modificación de su ficha. Es lo que dice qué borrar (lo que usted tiene y ya no viene), qué le falta y qué cambió.

Ruta	`GET` `/api/v1/<base>/productos`

Campo	Tipo	Descripción
productos	Array	Un elemento por producto publicado.
idProducto	Long	El producto.
fechaModificacion	String	La fecha de su ficha (`AAAA-MM-DD HH:MM:SS`). Si es **distinta** —más nueva o más vieja— de la que guardó, vuelva a pedir la ficha.
ids	Array	Los mismos ids, sin fecha. Existe por compatibilidad: use `productos` .

```
{ "productos": [ { "idProducto": 1, "fechaModificacion": "2026-09-27 21:48:43" },
                 { "idProducto": 2, "fechaModificacion": "2026-09-27 21:48:47" }, … ],
  "ids": [1, 2, …] }
```

4.3 fichas

La ficha de los productos, **sin precio, sin stock y sin fotos**: esos tres se piden aparte. De dos formas:

Por ids	`GET` `/api/v1/<base>/fichas?ids=17,20,31`
Por cambios	`GET` `/api/v1/<base>/fichas[?desde=<cursor>][&limite=100]` — las fichas en orden de modificación, de a una página. Sin `desde` parte desde el principio; cada respuesta trae el cursor para pedir la siguiente.

Campo	Tipo	Descripción
`fichas`	Array	Las fichas de esta página.
`siguiente`	String	El cursor para la página que sigue (`desde=`). Guárdelo: con él pide después sólo lo que cambió.
`hayMas`	Boolean	Si puede haber más: siga pidiendo hasta que llegue `false` .

La ficha

Campo	Tipo	Descripción
`idProducto`	Long	El id del producto.
`codigo`	String	El código (SKU).
`nombre`	String	El nombre.
`descripcion`	String	La descripción en HTML; sus imágenes apuntan al sitio de fotos de la empresa. `null` si no tiene.
`url`	String	Ruta amigable sugerida: `/producto/notebook-14-i5-8gb-p17` .
`texto`	String	Texto normalizado para un buscador: código, nombre, marca, part number, códigos alternativos y valores de la ficha técnica.
`metaTitle`, **`metaDescription`**, **`metaKeywords`**	String	Datos para buscadores (SEO).
`lista`	Integer	**1 si se muestra y se vende**. También viaja aparte, en disponibilidad, que es la que manda día a día.
`nuevo`, **`liq`**, **`out`**, **`preventa`**	Integer	Marcas (1/0): nuevo, en liquidación, dado de baja del catálogo, en preventa.
`api`, **`enCamino`**	Integer	1 si se vende del stock de un proveedor mayorista; 1 si viene en una orden de compra.
`idDisponibilidad`	Integer	`0` stock, `1` kit, `2` servicio, `3` kit fijo, `99` a pedido.
`idMarca`	Integer	La marca (maestros/marcas).
`idFamilia`, **`idFamilia2`**, **`idFamilia3`**	Integer	La familia principal y hasta dos más en que también aparece.
`idFamiliaTabulacion`	Integer	La familia cuya ficha técnica usa (maestros/tabulacion).
`partno`, **`upc`**	String	Número de parte del fabricante y código de barras.
`garantia`	Integer	Meses de garantía.

dimPeso, **dimAlto**, **dimAncho**, **dimLargo**	Double	Peso en kilos y medidas en centímetros, para el despacho.
idGrupo, **grupo**	Integer, Object	El grupo de variantes (una polera en tallas y colores): `idGrupo`, `nombre` y `etiqueta1` … `etiqueta3`, los nombres de los ejes.
opcion1…**opcion3**	String	El valor de este producto en cada eje (`M`, `Rojo`).
relacionados	Array	Ids de productos sugeridos junto a éste.
tabulacion, **tabulacionOpc**	Array	La ficha técnica: `idFamilia`, `idDetalle`, `detalle` (el valor para mostrar) y `detalleValor` (el mismo como número, para filtrar). La segunda son las filas de selección múltiple.
fechaCreacion, **fechaModificacion**	String	Fecha y hora.


```
GET /api/v1/miempresa/fichas?ids=17
{ "fichas": [ { "idProducto": 17, "codigo": "NB-001", "nombre": "Notebook 14 i5 8GB",
    "descripcion": "<p>…</p>", "lista": 1, "nuevo": 1, "liq": 0, "out": 0, "preventa": 0,
    "api": 0, "enCamino": 0, "partno": null, "upc": null, "idGrupo": null,
    "opcion1": null, "opcion2": null, "opcion3": null, "idFamilia": 3, "idFamilia2": null,
    "idFamilia3": null, "idFamiliaTabulacion": null, "idMarca": 1, "idDisponibilidad": 0,
    "dimPeso": 1.6, "dimAlto": 3.0, "dimAncho": 33.0, "dimLargo": 23.0, "garantia": 12,
    "texto": "nb 001 notebooks andina notebook 14 i5 8gb", "url": "/producto/notebook-14-i5-8gb-
p17",
    "fechaCreacion": "2026-09-27 21:45:33", "fechaModificacion": "2026-09-27 21:48:47",
    "metaTitle": null, "metaDescription": null, "metaKeywords": null, "grupo": null,
    "relacionados": [18], "tabulacion": [], "tabulacionOpc": [] } ],
  "siguiente": null, "hayMas": false }
```


4.4 precios

Los precios de todos los productos publicados en una lista de precios, con la oferta vigente y los precios por medio de pago ya calculados.

Ruta	**GET** `/api/v1/<base>/precios[?lista=<id_lista>]`
lista	La lista de precios (maestros/listas). Sin ella, la lista web de la empresa.

Una fila por producto con precio en esa lista. Montos en pesos, con IVA salvo los `…Neto`; los que la empresa no usa vienen en `null`.

Campo	Tipo	Descripción
idLista	Integer	La lista de estos precios.
filas	Array	
idProducto	Long	El producto.
precio	Double	Precio normal.
precioCash	Double	Pagando al contado (transferencia, efectivo).
precioDebito, **precioTarjeta**	Double	Con tarjeta de débito y de crédito.

precioOferta, **precioDebitoOferta**, **precioTarjetaOferta**	Double	La oferta vigente, y con débito y crédito.
desde, **hasta**	String	Vigencia de la oferta; puede empezar en el futuro.
precioNeto… **precioTarjetaOfertaNeto**	Double	Cada uno de los anteriores sin IVA.
precioDolar	Double	El precio neto en dólares.

Qué precio cobrar según la forma de pago lo dice el campo `precio` de cada forma de pago en maestros/formas (`precio`, `precio_cash` o `precio_tarjeta`).


```
{ "idLista": 1, "filas": [ { "idProducto": 1, "idLista": 1, "precio": 165410.0, "precioCash":
159990.0,
    "precioDebito": null, "precioTarjeta": null, "precioOferta": null, "precioDebitoOferta":
null,
    "precioTarjetaOferta": null, "precioNeto": 139000.0, "precioCashNeto": 134445.0,
"precioDebitoNeto": null,
    "precioTarjetaNeto": null, "precioOfertaNeto": null, "precioDebitoOfertaNeto": null,
    "precioTarjetaOfertaNeto": null, "precioDolar": 151.65, "desde": null, "hasta": null }, … ] }
```


4.5 stock

El stock disponible de todos los productos publicados, por sucursal.

Ruta	`GET` `/api/v1/<base>/stock`

Campo	Tipo	Descripción
filas	Array	Una fila por producto y sucursal **con stock mayor que cero**: lo que no viene no tiene stock.
idProducto	Long	El producto.
idSucursal	Integer	La sucursal (maestros/sucursales).
stock	Integer	Unidades. Si el producto se vende del stock de un mayorista, es el del mayorista y se repite en cada sucursal: no se suma.


```
{ "filas": [ { "idProducto": 1, "idSucursal": 1, "stock": 130 }, { "idProducto": 2, "idSucursal":
1, "stock": 145 }, … ] }
```


4.6 disponibilidad

Si cada producto se muestra hoy y de dónde sale su venta. Se pide aparte de la ficha porque cambia sin que la ficha cambie: por el stock mínimo para vender en la web, por una orden de compra que llega o por la oferta del mayorista.

Ruta	`GET` `/api/v1/<base>/disponibilidad`

Campo	Tipo	Descripción
filas	Array	Una fila por producto publicado.
idProducto	Long	El producto.
lista	Integer	**1 si se muestra y se vende; 0 si se oculta** (se conserva, porque puede volver).
enCamino	Integer	1 si viene en una orden de compra.
api	Integer	1 si la venta sale del stock de un mayorista.

```
{ "filas": [ { "idProducto": 1, "lista": 1, "enCamino": 0, "api": 0 }, … ] }
```

4.7 fotos

Las fotos del catálogo de todos los productos publicados, como direcciones públicas: no hay que bajar ningún archivo.

Ruta	GET /api/v1/<base>/fotos

Campo	Tipo	Descripción
filas	Array	Una fila por foto.
idProducto	Long	El producto.
id	Integer	El orden en la galería; la 1 es la principal.
url	String	La foto en su tamaño original.
url200	String	La miniatura de 200 px; `null` si no existe.

La dirección de una foto cambia cuando la foto cambia, así que se puede guardar en caché sin límite.

```
{ "filas": [ { "idProducto": 17, "id": 1, "url":
"https://static.miempresa.cl/fotos/17/1/1759012127000.webp",
               "url200": "https://static.miempresa.cl/fotos/17/1/200/1759012127000.webp" }, … ] }
```

4.8 fusiones

Los productos que desaparecieron al fusionarse con otro en Sphinx (el mismo producto creado dos veces). Sirve para redirigir la página del que ya no existe a la del que quedó.

Ruta	GET /api/v1/<base>/fusiones

Campo	Tipo	Descripción
filas	Array	
idProductoFusionado	Long	El producto que ya no existe.

idProducto	Long	El que quedó.

4.9 maestros

Las tablas chicas que se copian enteras. Cada una llega como `{"tablas": {"<tabla>": [filas]}}`, con los nombres de columna con guion bajo. Lo que no viene ya no existe.

Ruta	**GET** `/api/v1/<base>/maestros/<nombre>`

Maestro	Tablas	Qué trae
`sucursales`	`empresa_sucursal`, `empresa_sucursal_bodega`	Las sucursales publicadas (dirección, comuna, teléfono, coordenadas, horario, mapa, foto) y la bodega web de cada una.
`listas`	`producto_lista`	`id_lista`, `lista`.
`formas`	`forma_pago`, `forma_entrega`, `transporte`	Formas de pago (con el `precio` que les corresponde), de entrega (con `despacho` 1 si es a domicilio) y transportes. Sus ids son los que se envían al crear un pedido.
`marcas`	`producto_marca`	`id_marca`, `marca` y su logo.
`familias`	`producto_familia`, `producto_familia_dep`	Las familias publicadas (con `orden`, `url` y su familia de ficha técnica) y el árbol: qué familia cuelga de cuál.
`tabulacion`	`producto_familia_detalle`, `…_opc`, `…_rango`	Las filas de la ficha técnica de cada familia: sus nombres y tipos, las opciones y los rangos para filtrar.
`division-politica`	`region`, `comuna`	Las 16 regiones y 346 comunas con los códigos que Sphinx usa en las direcciones.
`despacho-promocion`	`despacho_promocion` y sus productos, familias, marcas y comunas	Las promociones de despacho (gratis o con descuento) y a qué se aplican.
`ventas`	`producto_ventas`	Lo más vendido: `id_producto` y su `posicion`.

```
GET /api/v1/miempresa/maestros/formas
{ "tablas": {
    "forma_pago":    [ { "id_formapago": 9, "formapago": "Transferencia Bancaria", "precio":
"precio" },
                       { "id_formapago": 6, "formapago": "WebPay", "precio": "precio" } ],
    "forma_entrega": [ { "id_formaentrega": 3, "formaentrega": "Despacho a Domicilio",
"despacho": 1 },
                       { "id_formaentrega": 4, "formaentrega": "Retiro en Tienda", "despacho": 0
} ],
    "transporte":    [ { "id_transporte": 20, "transporte": "ChileExpress" } ] } }
```

Las imágenes (logos, fotos de sucursal) llegan como dirección pública en `path_imagen`, y en `url_imagen` donde la tabla la tiene.

4.10 contenido

El contenido del sitio que la empresa administra en Sphinx. Sólo sirve a una tienda que tome ese contenido de Sphinx.

Ruta	`GET` `/api/v1/<base>/contenido/<nombre>`

Contenido	Qué trae
`datos`	Banners, imbatibles, avisos, destacados, logos del encabezado y páginas de aterrizaje vigentes, más la imagen de «no disponible».
`destacados-familia`	El producto destacado de cada familia.
`paginas`	Los parámetros del sitio, sus páginas y los sinónimos del buscador.

`arma-tu-pc` (`?plataforma=&sucursal=&productos=`) entrega el configurador de PC por compatibilidad, para las empresas que lo usan.

5 Ventas

Piden el alcance **Ventas**. Una venta en línea se registra así: se crea el pedido al confirmar el carro (reserva el stock) y el cliente paga, de una de dos formas: con la pasarela de Sphinx, que cobra y registra el pago sola, o con una pasarela suya, informándole después el pago a Sphinx. El recorrido completo está en 9.2.

5.1 Crear un pedido

Ruta	`POST` `/api/v1/<base>/pedidos`
Cuerpo	El pedido en JSON, con los mismos campos del documento de la interfazWEB.

```
{
  "tipo": "PEDIDO",
  "nroOrden": 51259,
  "idSucursal": 1,
  "idFormapago": 6,
  "idFormaentrega": 3,
  "rut": 21614514,
  "urlRetorno": "https://www.mitienda.cl/pago/retorno",
  "urlPedido": "https://www.mitienda.cl/pedido",
  "neto": 338639, "descuento": 0, "afecto": 338639, "iva": 64341, "total": 402980,
  "despacho": 5990,
  "cliente": { "nombre": "María Pérez Soto", "email": "maria@correo.cl", "fono": "+56 9 8765
4321",
               "direccion": "Los Aromos", "numero": "123", "adicional": "Depto 101",
               "idComuna": 322, "ciudad": "Santiago" },
  "despachoDomicilio": { "nombre": "María Pérez Soto", "fono": "+56 9 8765 4321",
               "direccion": "Los Aromos", "numero": "123", "adicional": "Depto 101",
               "idComuna": 322, "ciudad": "Santiago", "idTransporte": 20, "nota": "Dejar en
conserjería" },
  "detalle": [ { "idProducto": 17, "cantidad": 1, "precio": 396990, "descuento": 0,
                 "descripcion": "Notebook 14 i5 8GB" } ]
}
```

Encabezado

Campo	Tipo	Oblig.	Descripción
`tipo`	String	Sí	`PEDIDO`, `BOLETA`, `FACTURA` o `COTIZACION`. Casi todas las empresas trabajan con **Pedido Web**: con ellas todo lo que no es cotización queda como Pedido Web, y la boleta o factura la emite la empresa al despacharlo.
`nroOrden`	Long	Sí	El número de la orden en su sistema. **Evita las ventas duplicadas**: reenviar la misma orden con el mismo total devuelve el pedido que ya existe, sin crear otro.
`idSucursal`	Integer	Sí*	Sucursal que vende. *Con `PEDIDO` puede omitirse y se usa la sucursal web.

idBodega	Integer	No	Bodega de la que sale el stock; por omisión, la web de la sucursal.
idFormapago	Integer	Sí	Forma de pago (maestros/formas). Si es un medio en línea de Sphinx (WebPay, MercadoPago), la respuesta trae `id_pago` para cobrar (5.3).
idFormaentrega	Integer	No	Forma de entrega. Si es despacho, envíe `despachoDomicilio`.
idLista	Integer	No	Lista contra la que se validan precios y descuentos; por omisión, la web.
rut	Integer	Ver nota	RUT del comprador **sin dígito verificador**. Una boleta sin RUT va a «Cliente sin nombre»; una factura lo exige.
neto, **descuento**, **afecto**, **iva**, **total**	Double	Sí	En pesos enteros. Se validan, no se recalculan (ver abajo).
despacho	Double	No	Lo que se cobra por el despacho, **con IVA**.
urlRetorno	String	No	Con un medio de pago de Sphinx: adónde avisa Sphinx que el pago cambió (5.3). Sphinx le agrega `/<id_web>` al final.
urlPedido	String	No	Adónde vuelve el cliente después de pagar; sin ella, a `urlRetorno`. También con `/<id_web>` al final.
idWeb	String	No	Un identificador propio de la orden, que también evita duplicados. Sin él, Sphinx genera uno.
observaciones, **ordenCompra**, **ipAddress**	String	No	Nota del pedido, orden de compra del cliente e IP del comprador.

Cliente (`cliente`): `nombre` y `email` obligatorios si viene `rut`; `fono`, `direccion`, `numero`, `adicional`, `idComuna` (maestros/division-politica), `ciudad`; y para factura además `giro`. Con `rut`, Sphinx crea el cliente o actualiza el que ya tiene.

Despacho (`despachoDomicilio`): `nombre`, `fono`, `direccion`, `numero`, `adicional`, `idComuna`, `ciudad`, `idTransporte` y `nota`.

Productos (`detalle`): `idProducto` (el que descuenta stock; una línea sin producto, como un cargo, va sin él), `cantidad`, `precio` unitario **con IVA**, `descuento` unitario con IVA, y `descripcion`, el texto que se imprime (obligatorio).

Los totales

Sphinx calcula los totales y los compara con los enviados; si no calzan, rechaza el pedido. Con IVA de 19 %:

```
neto   = redondeo( ( despacho + Σ cantidad × (precio − descuento) ) / 1,19 )   ± 10 pesos
afecto = neto − descuento                                                      exacto
iva    = redondeo( afecto × 0,19 )                                             ± 5 pesos
total  = afecto + iva                                                          exacto
```

En el ejemplo: (5.990 + 396.990) / 1,19 = 338.638,66 → `neto` 338.639; `iva` 64.341; `total` 402.980.

La respuesta

Campo	Tipo	Descripción
`id_doc`	Long	El documento en Sphinx.
`nro_orden`	Long	El número de orden enviado.
`id_web`	String	El identificador del pedido. **Guárdelo**: con él se consulta (5.2).
`id_pago`	String	Sólo con un medio de pago de Sphinx: el cobro a abrir (5.3).
`documento`	String	El documento creado: `Pedido WEB 51259`.
`pdf`	String	Sólo si en esta llamada se emitió una boleta o factura: su PDF en base64.
`correo`	String	El correo del cliente.

```
{ "id_doc": 5397, "id_pago": "9f86d081884c7d659a2feaa0c55ad015", "nro_orden": 51259,
  "id_web": "1b0c6f0e-4f7e-4c50-9a3d-0f2f1d7c2b11", "documento": "Pedido WEB 51259", "correo":
"maria@correo.cl" }
```

Un rechazo es un **422** con el motivo en `detail`, listo para mostrar: `Falta Stock de los productos : NB-001`, `Falta email del Cliente`, `Total […] no coincide con Afecto + IVA…`, `Producto [17] no existe`, `… del Pedido #51259 ya existe en el Sistema` (mismo número de orden con otro total). Un cliente bloqueado para comprar llega con `codigo` 42.

5.2 Consultar un pedido

Ruta	**GET** `/api/v1/<base>/pedidos/<id_web>`

El documento completo, con su cliente, sus líneas, sus totales y sus pagos, con los mismos nombres de campo con que se creó. Si no existe, 422 con `codigo` 51.

5.3 Cobrar con la pasarela de Sphinx

Si la empresa tiene configurado en Sphinx un medio de pago en línea —WebPay, MercadoPago o Linkify—, Sphinx cobra por usted: no necesita integrar la pasarela.

1. Cree el pedido con una forma de pago en línea y con `urlRetorno` (y `urlPedido` si la página de vuelta es otra). La respuesta trae `id_pago`.
2. Mande al cliente a pagar, en su navegador, a:

   ```
   https://<servidor>/interfazPAGO/<base>/<id_pago>
   ```

   Sphinx abre el medio de pago con el monto del pedido.

3. Cuando el pago cambia, Sphinx hace dos cosas: le envía un `POST` servidor a servidor a `urlRetorno/<id_web>` con `uuid=<id_pago>` e `interno=1`, y devuelve al cliente a `urlPedido/<id_web>` (o a `urlRetorno`) con el mismo `uuid`.
4. Al recibir cualquiera de las dos, confirme con `GET /pagos/<uuid>`: es la única fuente de verdad. **No dé una venta por pagada sólo porque llegó el aviso.**

Si el pago se aprueba, Sphinx lo registra en la caja y el pedido queda pagado; no hay que informar nada más. Si el cliente no paga, el pedido vence solo —unos 20 minutos— y libera el stock.

> **¿Cobra con su propia pasarela?** Cree el pedido con la forma de pago que corresponda (sin `urlRetorno`) y, cuando su pasarela apruebe, infórmele el pago a Sphinx (5.5).

5.4 Consultar un pago

Ruta	`GET` `/api/v1/<base>/pagos/<uuid>`

Si el pago está aprobado contesta 200 con el comprobante. Pendiente o sin pago: 422 con `codigo` 100. **Rechazado: 422 con `codigo` 21** y el comprobante en `resultado`, con el motivo en `respuesta`.

Campo	Tipo	Descripción
`medio`	String	El medio de pago.
`valido`	Integer	1 si fue aprobado.
`codigo_respuesta`, **`cod_autorizacion`**	Int, String	La respuesta del medio y el código de autorización.
`fecha_transaccion`	String	Cuándo se pagó.
`tipo_pago`, **`num_cuotas`**, **`ult_digitos`**	String, Int, String	Débito o crédito, cuotas y los últimos dígitos de la tarjeta.
`monto`	Double	El monto pagado.
`respuesta`	String	El texto de la respuesta; en un rechazo, el motivo.

Si el pago fue rechazado, anule el pedido (5.6) para liberar el stock.

5.5 Informar un pago de su pasarela

Si el cliente pagó con una pasarela suya —Flow, Khipu, MercadoPago con su propia cuenta, una transferencia que usted verificó—, infórmele el pago a Sphinx. Sphinx lo registra y, si con eso la boleta o la factura queda pagada, la emite.

Ruta	`POST` `/api/v1/<base>/pedidos/<id_web>/pagos[?correo=true]`
correo	Si se emitió la boleta o factura, se le envía al cliente por correo.

```
{
  "entidad": "FLOW",
  "valido": 1,
  "codAutorizacion": "A1B2C3",
  "monto": 402980,
  "nroTransaccion": "T-77",
  "fechaTransaccion": "2026-09-27",
  "horaTransaccion": "12:34:55"
}
```

Campo	Tipo	Oblig.	Descripción
`entidad`	String	Sí	Quién cobró, en hasta 15 caracteres (ver abajo).
`valido`	Integer	Sí	Debe ser `1`: sólo se informan pagos aprobados.
`codAutorizacion`	String	Sí	Código de autorización del pago (hasta 10 caracteres).
`monto`	Double	Sí	El monto pagado: el `total` del pedido, con 5 pesos de tolerancia.
`nroOrden`	String	No	El número de orden del pedido; si no viene, se toma del pedido.
`tipoPago`	String	No	El tipo de pago de Transbank: `VD` débito, `VN` crédito sin cuotas, `VC` / `NC` / `S1` / `S2` en cuotas, `VP` prepago.
`tipoTarjeta`, **`abrevTarjeta`**	String	No	Tipo (`DB`, `CR`) y marca de la tarjeta (`VI`, `MC`, `AX`, `DC`, `DB` …).
`idBanco`	Integer	No	El banco, en un pago que no es con tarjeta.
`fechaTransaccion`, **`horaTransaccion`**	String	No	`AAAA-MM-DD` y `HH:MM:SS`.
otros	String	No	`nroTransaccion`, `codigoRespuesta`, `codigoComercio`, `terminalId`, `numCuotas`, `ultDigitos`, `numOperacion`, `rutCliente`: se guardan con el pago tal como los entregó su pasarela.

La entidad

Sirve para **cualquier pasarela**: lo que cambia de una a otra es la `entidad`. `WEBPAY`, `ONEPAY` o `TRANSBANK` quedan en Sphinx como Webpay; `MERCADOPAGO` como MercadoPago; cualquier otro nombre (`FLOW`, `KHIPU`, `GETNET`, `TRANSFERENCIA` …) como un medio de pago con ese nombre. Use siempre el mismo nombre para la misma pasarela: con él la empresa cuadra y concilia esos pagos. Para que el dinero quede en la cuenta correcta, la empresa crea en Sphinx una cuenta del tipo *Medios Pago* con ese mismo nombre.

La respuesta

Campo	Tipo	Descripción
`documento`	String	El documento pagado: `Pedido WEB 51259`, `Boleta Electronica 245678`.

pdf	String	Sólo si se emitió una boleta o factura: su PDF en base64.
correo	String	El correo del cliente.

```
{ "documento": "Pedido WEB 51259", "correo": "maria@correo.cl" }
```

Un **Pedido Web** queda pagado y la empresa emite la boleta o factura al despacharlo. **Reenviar el mismo pago no lo cobra dos veces**, así que si la llamada no contesta, repítala.

Errores (422): `codigo` 100 `Monto del Voucher $X no coincide con el Documento $Y` o `Voucher Pago no es valido` (falta `valido` = 1 o la autorización); 12 el pedido está anulado —venció antes del pago: hay que devolverle el dinero al cliente o crear la venta de nuevo—; 25 el documento no se puede pagar; 51 el pedido no existe; 20 el pago no se pudo registrar.

5.6 Anular un pedido

Anula un pedido que no se pagó —el cliente abandonó el pago o la pasarela lo rechazó— y libera el stock que reservaba.

Ruta	POST `/api/v1/<base>/pedidos/<id_web>/anulacion`
Cuerpo	`{ "motivo": "Pago rechazado por la pasarela" }` — el motivo es obligatorio y queda en el pedido.

```
{ "mensaje": "Pedido anulado" }
```

Anular dos veces no es un error: la segunda contesta `El pedido ya estaba anulado`. Sólo se anula lo que sigue pendiente; si no se puede, contesta 422 con `codigo` 100 y el motivo:

detail	Por qué
`Pago asociado`	El pedido ya tiene un pago registrado. Si hay que devolver el dinero, lo resuelve la empresa en Sphinx.
`Documento asociado`	La empresa ya lo convirtió en boleta, factura u otro documento.
`… no esta pendiente, no lo anulo`	Es una boleta o factura ya emitida: se corrige con una nota de crédito.
`El pedido tiene un pago en linea en curso…`	El cliente está pagando con la pasarela de Sphinx. Se anula sólo si ese pago se rechaza o vence: espere el resultado.

Si no existe, `codigo` 51. Si nadie lo anula, un pedido sin pagar vence solo y libera el stock igual, pero mientras tanto ese stock queda bloqueado.

6 Consultas

Para que un sitio le muestre algo a su cliente. Piden el alcance **Consultas**.

6.1 boletas

El PDF de una boleta electrónica, buscada por los tres datos que el cliente tiene impresos.

Ruta	`GET` `/api/v1/<base>/boletas/<folio>?fecha=AAAA-MM-DD&monto=<total>`

La respuesta es el PDF mismo (`application/pdf`), en formato térmico. Si algún dato no calza: 422 con `codigo` 69 (no existe), o 60, 61, 62 si falta el folio, la fecha o el monto.

6.2 ordenes-servicio

Las órdenes de servicio técnico de un cliente de los últimos tres meses.

Ruta	`GET` `/api/v1/<base>/ordenes-servicio?rut=12608204-5`

```
{ "ordenes": [ { "fecha": "2026-09-10 00:00:00", "folio": 2, "estado": "Pendiente" } ] }
```

Errores (422): `codigo` 45 RUT inválido, 46 no es cliente, 42 cliente bloqueado.

6.3 despacho/seguimiento

Dónde va un pedido: el documento, su estado de pago y el seguimiento del despacho. Requiere que la empresa despache con Shipit.

Ruta	`GET` `/api/v1/<base>/despacho/seguimiento?nroOrden=51259&email=maria@correo.cl`

Campo	Tipo	Descripción
`tipo`, **`folio`**, **`rut`**	String, Long, String	El documento.
`despacho`	Boolean	Si es con despacho a domicilio.
`operador`, **`nroSeguimiento`**, **`urlSeguimiento`**	String	El transporte y cómo seguirlo en su sitio.
`fechaEstimada`, **`origen`**, **`destino`**	String	Fecha estimada de entrega y ciudades.
`pasos`	Array	El historial: `fecha` y `estado` de cada paso.
`transferencia`, **`urlTransferencia`**	Boolean, String	Si se paga por transferencia y dónde informarla.

Errores (422): `codigo` 200 sin Shipit; 300 si el número de orden y el correo no calzan.

6.4 despacho/tarifa

Cotiza el despacho de un carro: los transportes disponibles con su precio y su plazo. Requiere Shipit.

Ruta	`POST` `/api/v1/<base>/despacho/tarifa`

```
{ "id_comuna": 322, "id_lista": 1, "total": 402980,
  "producto": [ { "id_producto": 17, "cantidad": 1 } ] }
```

La respuesta trae `transporte`: una fila por opción con `id_transporte`, `transporte`, `precio`, `dias`, `fecha_estimada` e `id_promocion` si aplica una promoción de despacho.

6.5 despacho de un documento

El despacho de una factura, boleta o guía, para que el cliente lo siga con su RUT y el folio impreso. A diferencia de despacho/seguimiento, no necesita Shipit.

Ruta	`GET` `/api/v1/<base>/despacho/<tipo>/<folio>?rut=12608204-5`
tipo	`factura`, `boleta` o `guia`.

Campo	Tipo	Descripción
`tipo`, **`folio`**	String, Long	El documento (`Factura Electronica`).
`rut`, **`rutDv`**	Integer, String	RUT del cliente y su dígito verificador.
`fecha`	String	Fecha del despacho.
`transporte`	String	Empresa de transporte.
`orden`	String	Número de seguimiento del transporte.
`bultos`, **`peso`**, **`destino`**	Int, String, String	Bultos, peso y ciudad de destino.

```
{ "tipo": "Factura Electronica", "folio": 1783, "rut": 12608204, "rutDv": "5", "fecha": "2026-09-22",
  "transporte": "ChileExpress", "orden": "99812345", "bultos": 1, "peso": "2", "destino": "SANTIAGO" }
```

Errores (422): `codigo` 45 RUT inválido, 46 no es cliente, 42 cliente bloqueado, 80 tipo inválido, 81 folio inválido, 82 el documento no tiene despacho.

7 Clientes del sitio

Lo que necesita una tienda en línea que tiene cuentas de cliente, formulario de contacto y suscripción a novedades. Piden el alcance **Clientes**. Las rutas de la cuenta identifican al cliente con el token que entrega el inicio de sesión, en el encabezado `X-Cliente-Token`.

Ruta	Qué hace
`POST` `cuenta/sesion`	`{email, password}`: entra y devuelve el cliente con su token y sus direcciones.
`POST` `cuenta`	Registra una cuenta (envía el código de activación por correo) o la activa con el código.
`GET` / `PUT` `cuenta`	Los datos del cliente, o los modifica (incluida la clave).
`POST` `cuenta/recuperacion`	Envía un código para recuperar la clave, y lo canjea por una nueva.
`GET` `cuenta/documentos`	Sus compras, de a 15 (`?page=`); el PDF de cada una en `cuenta/documentos/<idDoc>/pdf`. `DELETE` anula una pendiente.
`GET` / `POST` / `DELETE` `cuenta/favoritos`	Sus productos favoritos.
`POST` `contactos`	Crea un contacto (consulta, reclamo) y devuelve su folio. Se sigue con `GET contactos/<uuid>`, se le agregan notas y archivos con `POST contactos/<uuid>/notas` y se cierra con `DELETE`.
`POST` `suscripciones`	`{email, ip, referer}`: suscribe un correo a las novedades.

Los errores de la cuenta llegan como 422 con su `codigo`: 40 correo o clave inválidos (o cuenta sin activar), 41 bloqueada por intentos, 50 token inválido.

8 Reportes

Para tableros de gestión y cuadraturas contables. Todos son `GET` y piden el alcance **Reportes**. Los tres que leen mucho no corren dos veces a la vez para la misma empresa: la segunda llamada contesta 422 con `codigo` 8 y se reintenta cuando termine la primera. Una vez por hora es más que suficiente para un tablero.

8.1 reportes/estadisticas

Las ventas de un período agrupadas por hasta cuatro criterios, como en la pantalla Estadísticas de Sphinx.

Ruta	`GET` `/api/v1/<base>/reportes/estadisticas?desde=AAAA-MM-DD&hasta=AAAA-MM-DD[&agrupar=DOC,VEND][&pagina=1]`
Tope	Hasta 120 días entre las dos fechas; 100 filas por página.

`agrupar`: hasta cuatro, en orden, separados por coma: `DOC` documento, `DTD` tipo de documento, `DOC_NUM` documento con su número, `DBOD` bodega, `PROVI` origen de la venta, `VEND` vendedor, `D` día, `H` hora.

Campo	Tipo	Descripción
filas	Array	Una fila por combinación de las agrupaciones.
grupos	Array	El valor de cada agrupación, en el orden pedido: `codigo` y `descripcion`.
cantidad	Double	Documentos.
unidades	Double	Unidades vendidas.
monto	Double	Venta neta; las notas de crédito restan.
pagina, **paginas**, **registros**	Long	La página actual, cuántas hay y el total de filas.

```
{ "filas": [ { "grupos": [ { "codigo": "BVE", "descripcion": "Boleta Electronica" },
                           { "codigo": "DH", "descripcion": "Daniel Hasan" } ],
               "cantidad": 35.0, "unidades": 84.0, "monto": 39206380.0 } ],
  "pagina": 1, "paginas": 1, "registros": 3 }
```

Un rango de más de 120 días o una agrupación desconocida: 422 con `codigo` 100. Una fecha mal escrita: 400.

8.2 usuarios

Los usuarios activos de Sphinx, para cruzarlos con el vendedor de la estadística (`username`) o el cajero del reporte diario.

Ruta	`GET` `/api/v1/<base>/usuarios`

```
[ { "idUser": 1, "username": "DH", "rut": "12345678-9", "nombre": "Daniel Hasan", "idSucursal":
1, "sucursales": [1, 2] } ]
```

8.3 reportes/cuadratura

Lo recaudado en un período, sumado por sucursal y por medio de pago, como en la pantalla Cuadratura de Caja.

Ruta	`GET` `/api/v1/<base>/reportes/cuadratura[?desde=&hasta=][&sucursal=][&usuario=][&cliente=false]`

Sin fechas, el día de hoy. `sucursal` y `usuario` (el `idUser` del cajero) filtran; `cliente=false` da lo pagado a proveedores en vez de lo cobrado a clientes.

```
[ { "sucursal": "Matriz", "tipo": "CAJA", "cantidad": 11.0, "monto": 2436287.0 },
  { "sucursal": "Matriz", "tipo": "TARJETA MASTERCARD", "cantidad": 5.0, "monto": 8852493.0 } ]
```

8.4 reportes/diario

El reporte diario de caja de una sucursal: cada documento pagado ese día con el detalle de cómo se pagó, como en la pantalla Reporte Diario.

Ruta	`GET` `/api/v1/<base>/reportes/diario?fecha=AAAA-MM-DD&sucursal=1[&usuario=][&cliente=false]`

Trae cinco listas con las mismas columnas: `pagosDia` (documentos del día pagados el mismo día), `pagosOtroDia` (pagos del día de documentos de otros días), `pagosCambio`, `pagosEgreso` y `pagosGiftCard`. Cada fila lleva el documento (`documento`, `td`, `folio`, `fecha`), el cliente (`rut`, `nombre`), `monto`, `saldo`, `sucursal`, `cajero`, y sus pagos por medio: `efectivo`, `deposito`, `tarjeta`, `webpay`, `chequeDia`, `chequeFecha`, `otro` y `abonos`, cada uno con `tipo`, `folio`, `fecha` y `monto`.

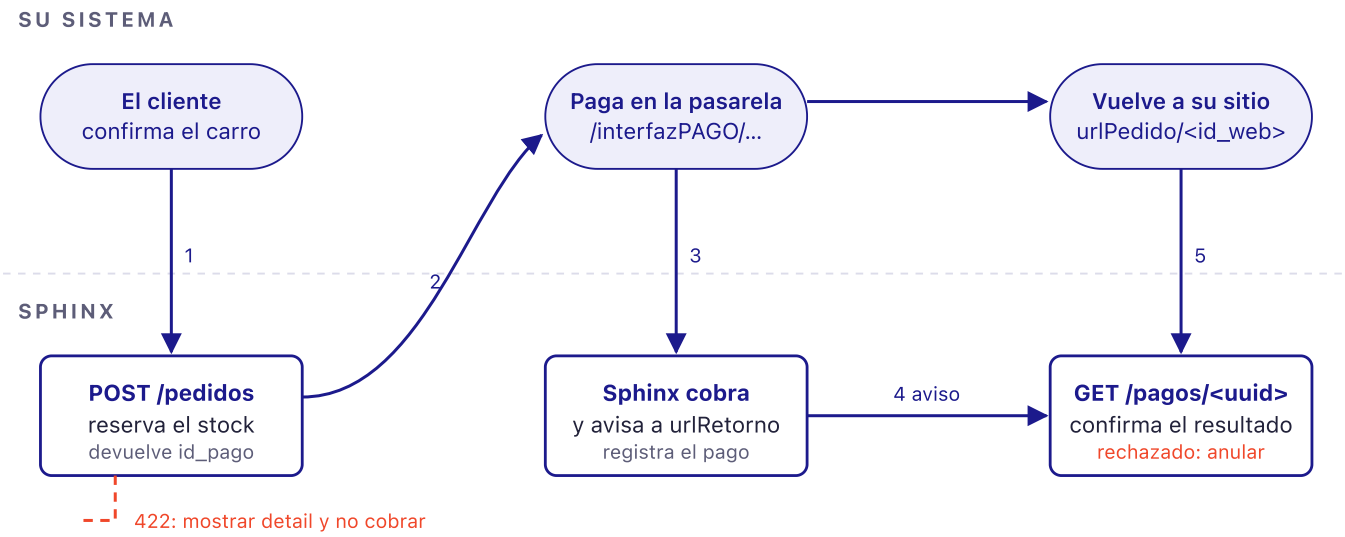
```
{ "pagosDia": [], "pagosCambio": [], "pagosEgreso": [], "pagosGiftCard": [],
  "pagosOtroDia": [ { "idCtacte": 100666, "documento": "FVE 678", "td": "FVE", "folio": 678,
"fecha": "2026-09-22",
                      "rut": "15678901-1", "nombre": "ANA MORALES CASTRO", "monto": 231693.0,
"sucursal": "Matriz",
                      "cajero": "DH", "efectivo": [ { "tipo": "", "folio": null, "fecha": null,
"monto": 231693.0 } ],
                      "deposito": [], "tarjeta": [], "webpay": [], "chequeDia": [],
"chequeFecha": [], "otro": [], "abonos": [] } ] }
```

9 Procesos recomendados

9.1 Sincronizar el catálogo

1. **Maestros, una vez al día:** `maestros/sucursales`, `listas`, `formas`, `marcas`, `familias`, `division-politica`. Cada uno trae la tabla completa.
2. **Qué productos hay, cada 15 minutos:** `productos`. Despublique lo que ya no viene; marque para actualizar lo nuevo y lo que tiene otra `fechaModificacion`.
3. **Las fichas marcadas:** `fichas?ids=…`, de a 100. O, más simple, recorra `fichas?desde=<cursor>` hasta `hayMas` en `false` y guarde el último `siguiente`: la próxima vez parte de ahí y trae sólo lo que cambió.
4. **Precio, stock, disponibilidad y fotos, cada 5 minutos**, con `If-None-Match`: si no cambió nada, cada llamada es un 304 sin cuerpo.
5. **Muestre** sólo lo que tiene `lista` = 1 en `disponibilidad`.

9.2 Registrar una venta



1. Al confirmar el carro, `POST /pedidos`. Si contesta 422, muestre `detail` y no mande a pagar. Si contesta 200, el stock quedó reservado: guarde `id_web` e `id_pago`.
2. Mande al cliente a `/interfazPAGO/<base>/<id_pago>`.
3. Sphinx cobra con el medio de pago de la empresa.
4. Sphinx avisa a `urlRetorno` y devuelve al cliente a `urlPedido`, los dos con el `uuid`.
5. Confirme con `GET /pagos/<uuid>`: aprobado, muestre el éxito; rechazado, anule con `POST /pedidos/<id_web>/anulacion` para liberar el stock.

Con su propia pasarela, los pasos 2 a 5 son: cobre con ella y, cuando apruebe, informe el pago con `POST /pedidos/<id_web>/pagos` (5.5). Si el cliente no paga, el pedido vence solo y libera el stock.

Reintente sin miedo. Si `POST /pedidos` no contesta, repítalo con el mismo `nroOrden` y el mismo total: si el pedido alcanzó a crearse, Sphinx devuelve el mismo en vez de crear otro.

9.3 Antes de salir a producción

- `GET /estado` contesta 200 con su clave en la base de producción.
- Su clave tiene todos los alcances que usa, y ninguno más.
- Guardó los ids de las formas de pago y de entrega de `maestros/formas`.
- Una venta de prueba completa —pedido, pago, confirmación— y otra rechazada y anulada funcionaron. Coordínelas con la empresa, porque quedan registradas en su sistema.
- Su programa ignora los campos que no conoce y usa `If-None-Match`.

Las dudas sobre la integración se atienden en **soporte@sphinx.cl**. Incluya la ruta, la hora de la llamada y el cuerpo enviado.

10 Equivalencias con la interfazWEB

Qué usar en la API v1 en lugar de cada servicio de la interfazWEB.

interfazWEB	API v1
`Check`	`GET /estado`
`Sucursales`, `Listas`, `Formas`, `Marcas`, `DivisionPolitica`, `Familias`	`GET /maestros/sucursales`, `listas`, `formas`, `marcas`, `division-politica`, `familias`
`Productos`	`GET /productos` (por `idProducto`)
`Producto`	`GET /fichas` + `/precios` + `/stock` + `/disponibilidad` + `/fotos`
`WEB_Datos`	`GET /contenido/datos`
`Documento`	`POST /pedidos` (el mismo cuerpo)
`Boleta`	`GET /boletas/<folio>?fecha=&monto=` (el PDF directo, no en base64)
`OrdenServicio`	`GET /ordenes-servicio?rut=`
`Documento_Voucher`	`POST /pedidos/<id_web>/pagos` (el mismo cuerpo)
`Documento_Anular`	`POST /pedidos/<id_web>/anulacion` con `{"motivo": …}`
`Estadisticas`	`GET /reportes/estadisticas?desde=&hasta=&agrupar=`
`Usuarios`	`GET /usuarios`
`Cuadratura`	`GET /reportes/cuadratura?desde=&hasta=`
`ReporteDiario`	`GET /reportes/diario?fecha=&sucursal=`
`Despacho`	`GET /despacho/<tipo>/<folio>?rut=`