Checkout para Moodle

Documentación técnica · versión 1.11.0 (2026100702) · `local_tiendacursos` + `block_tiendacursos`

1. Compatibilidad y requisitos

- Moodle 4.5 LTS (2024100700) a 5.1 (`$plugin->supported = [405, 501]`), PHP 8.1 o superior, PostgreSQL, MySQL o MariaDB.
- Cron cada 1–5 minutos. Salida HTTPS hacia `www.flow.cl` , `sandbox.flow.cl` , `api.mercadopago.com` y `licencias.norteduc.cl` (activación única).
- Sin dependencias externas ni CDN: JavaScript mínimo en línea, estilos en `styles.css` con variables del tema.
- El bloque depende de `local_tiendacursos` ≥ 2026100221 (bloque 1.2.0).

2. Arquitectura y por qué no se usa paygw + enrol_fee

Moodle trae un subsistema de pagos (`core_payment` , pasarelas `paygw_*`) y la matrícula por pago (`enrol_fee`). Ese diseño exige que el comprador **ya sea usuario y esté autenticado**: la «cuenta de pago» se asocia a un `userid` y la matrícula se hace sobre ese usuario. La tienda necesita lo contrario:

- compra **sin cuenta**: el pedido existe antes que el usuario, y la cuenta se crea al confirmarse el pago;
- un pedido con **N estudiantes** distintos del comprador (compra grupal);
- **renovación** de la misma matrícula vencida, cupones y descuentos por tramo;
- pasarelas chilenas (Flow) con confirmación servidor a servidor y conciliación propia.

Por eso la tienda es un complemento local con su **propia tabla de pedidos** y **adaptadores de pasarela** (`classes/gateway/*`) con una interfaz común (`start` , `verify` , `order_from_notification` , `test_connection`). La matrícula usa la API estándar de `enrol_manual` (`enrol_user` , `update_user_enrol`), así que el resto de Moodle (informes, finalización, certificados) ve matrículas normales.

3. Flujo de un pedido

```
checkout.php ──► orders::create()            pedido "pending" + filas de estudiantes (transacción)
             └─► orders::start_payment()     $0 → mark_paid · transferencia → instrucciones · en línea → URL pasarela
pasarela ──server──► notify.php              NO_MOODLE_COOKIES, sin exigir user-agent; GET+POST+JSON
             └─► gateway::order_from_notification() → gateway::verify()   (consulta a la API, nunca confía en el aviso)
             └─► orders::apply_verification() estado + monto ≥ pedido + referencia propia
             └─► orders::mark_paid()          lock por pedido; idempotente
                   ├─ primera vez: estado paid, uso de cupón, evento order_paid, comprobante, aviso a ventas
                   └─ fulfil(): por estudiante → ensure_user() → enrol() / renovación → evento student_enrolled → bienvenid
result.php (regreso del navegador)            si sigue pendiente, vuelve a consultar la API; nunca matricula por sí mismo
task\reconcile_orders (cada 15 min)          consulta pedidos en línea pendientes (>5 min); expira: en línea 3 días, transfe
```

4. Seguridad

Riesgo	Control
Matrícula por un regreso falsificado del navegador	Solo `apply_verification()` con datos leídos de la API de la pasarela marca un pedido pagado.
Aviso falso a notify.php	El aviso solo aporta un token o id; el estado, el monto y la referencia se consultan a la API firmada (Flow: HMAC-SHA256; MP: Bearer token). Se compara `commerceOrder` / `external_reference` y la moneda.
Aviso firmado falso (1.5)	Khipu: HMAC-SHA256 del cuerpo sin tocar con el secreto (`hash_equals`); Getnet: SHA-256/SHA-1 de requestId+estado+fecha+secretKey; MACH: token del webhook; PayPal: verify-webhook-signature. Un aviso que no pasa se registra como `notify_badsignature` y se ignora; aun firmado, el estado se lee de la API.
Pago por menos	Si el monto informado es menor que el pedido, se registra `amount_mismatch` y no se matricula.

Avisos repetidos o concurrentes	Lock por pedido (`\core\lock`) + estado; matrícula idempotente por estudiante.
Avisos bloqueados	notify.php no exige cookies ni user-agent; alerta de pagos pendientes >1 h; conciliación cada 15 min.
CSRF en el panel	Todas las acciones que cambian datos son POST con `sesskey` .
Acceso al panel	`require_login` + `panel::require_section($seccion)` al inicio de cada página y `panel::require_section($seccion, 'edit')` antes de cada cambio (no basta con ocultar botones). Usuarios del panel y Licencia: solo `is_siteadmin()` . Los roles y capacidades de Moodle no dan acceso.
Versión gratuita	Los límites se verifican en el servidor: `shop::enabled_gateways()` (1 medio), `orders::max_students()` (1), `orders::quote()` (sin cupón ni tramo), `orders::create()` rechaza un medio no ofrecido o más de 1 estudiante, `panelusers::save()` exige licencia, y Pasarelas rechaza activar un segundo medio.
Contraseñas	La contraseña temporal solo viaja en el correo de bienvenida; no se guarda en pedidos, cola ni registros. La contraseña SMTP se enmascara (texto y base64) en errores; el registro SMTP solo conserva líneas del servidor.
Secretos de pasarela y SMTP	Nunca se devuelven al navegador (campo vacío = conservar la guardada; casilla «Borrar la guardada»); los usuarios de solo lectura ven «Guardada (oculta)». La última prueba de conexión guarda solo ok/mensaje/fecha.
Catálogo en sitios externos (1.3)	Solo con licencia y ajuste `ext_enabled` . CORS: se repite el `Origin` exacto solo si es un sitio autorizado activo (o el propio Moodle), nunca `*` ; origen no autorizado o `null` → 403. Iframe: `Content-Security-Policy: frame-ancestors 'self'` + sitios autorizados (o solo el de la clave) y se quita `X-Frame-Options` solo en `embed.php` ; `allowframembedding` no se toca. Sin cookies (`NO_MOODLE_COOKIES`), límite por IP y minuto, el feed contiene solo datos públicos de la vitrina y embed.js escribe todo con `textContent` y solo enlaza a URLs del mismo Moodle.
Entrada del visitante	RUT con dígito verificador (o documento extranjero de 4–20 caracteres alfanuméricos), correo validado, textos limpiados y truncados.
Pedidos automatizados (1.8, pentest TC-01)	`local\antispam` : campo trampa `website` , sello `ltcts` = tiempo + HMAC-SHA256 con la `sesskey` (mínimo 3 s; ligado a la sesión), pedidos por visitante y hora (`ordersperhour` , por defecto 5; visitante = sha1 de IP + sitio), 10 cupones equivocados por hora y 3 pedidos pendientes por correo en 24 h. Contadores en el caché MUC `antispam` (TTL 1 h) y en la tabla de pedidos. Usuarios del panel exentos.
Datos personales en result.php (1.8, TC-02)	La referencia viaja en correos: datos completos solo para la sesión que creó el pedido (`$SESSION->local_tiendacursos_orders`), el comprador con sesión o un usuario del panel; para los demás, solo mientras el pedido está pendiente o pagado hace < 24 h (y < 7 días desde su creación), con el correo enmascarado (`orders::mask_email()`); fuera de esa ventana, vista limitada (estado y referencia). La ventana existe porque varias pasarelas vuelven con un POST de otro sitio, sin la cookie de sesión.
Compra para la cuenta de otra persona (1.8, TC-03)	Si el usuario (RUT o documento) existe con otro correo y no tiene ni tuvo matrícula en los cursos del pedido (`access::state` ≠ NONE), `enrol_student()` no matricula: estado `review` , aviso `mailer::review_notice()` al correo de ventas con ambos correos enmascarados y botón «Matricular de todos modos» (`retry_student()` → `enrol_student(…, force: true)`). Las renovaciones no se detienen.

5. Modelo de datos (XMLDB)

Tabla	Contenido	Índices
`local_tiendacursos_course`	Oferta por curso: precio, precio normal, horas, días de acceso, visible, destacado, orden, modo de venta, y (1.16) `modality` (elearning|presencial|semipresencial), `startdate` / `enddate` (0 = sin fecha), `sendcredentials` (1/0) y (1.17) `sencecode` y `syllabusmode` (free|lead).	courseid único
`local_tiendacursos_lead`	1.17: leads capturados por el formulario de descarga del temario (nombre, apellido, correo, empresa, token, fecha).	courseid, token
`local_tiendacursos_coupon`	Código, tipo percent|amount, valor, curso (0 = todos), usos máx./usados, vencimiento, activo.	code único

`local_tiendacursos_tier`	Tramos: curso (0 = generales), mín., máx. (0 = abierto), %.	courseid
`local_tiendacursos_order`	Pedido: referencia, curso, comprador, cantidad, precio, subtotal, descuentos (grupal y cupón), monto, pasarela, ref. pasarela, estado, matriculado, renovación, fechas. 1.2: `packid` (0 = curso; entonces `courseid` = 0) y `packcourses` (ids de los cursos del pack al comprar). 1.5: `paycurrency`, `payamount` (centavos) y `payrate` (pago en otra moneda: PayPal) y estado `refunded`. 1.8: `currency` (char 3, por defecto CLP; los montos del pedido están en la unidad menor de esa moneda) y `rut` ampliado a 40 (clave de identidad).	reference único; status+timecreated; gateway+gatewayref; userid; courseid; packid
`local_tiendacursos_student`	Estudiante del pedido: datos, userid, cuenta existente, renovación, fin de acceso, estado pending|enrolled|error|review (1.8), error. `rut` ampliado a 40 (1.8).	orderid; userid
`local_tiendacursos_log`	Historial del pedido (sin secretos).	orderid
`local_tiendacursos_puser` (1.1)	Usuarios del panel: `userid` (único), `perms` (JSON sección → none/read/edit), fechas y `usermodified`.	userid
`local_tiendacursos_pack` (1.2)	Pack: nombre, descripción, precio de oferta (0 = suma), oferta hasta, días de acceso, destacado, activo, orden. El precio normal no se guarda: es la suma de las ofertas de sus cursos. Imagen en el área `packimage` (contexto sistema, itemid = pack).	—
`local_tiendacursos_packcourse` (1.2)	Cursos de cada pack con su orden.	packid+courseid único
`local_tiendacursos_tpl` (1.2)	Plantillas de correo personalizadas: `kind`, asunto, título, cuerpo (NULL = cadena del idioma), activo, `usermodified`. Nunca contraseñas.	kind único
`local_tiendacursos_notice` (1.2)	Avisos de vencimiento enviados: userid, curso o pack, tipo (`soonN` / `expired`), fecha de término avisada, fecha de envío.	userid+courseid+packid+kind+timeend único
`local_tiendacursos_site` (1.3)	Sitios externos autorizados: nombre, `origin` (`esquema://host[:puerto]`, normalizado), `sitekey` (24 hex, pública), activo, último uso (se escribe a lo más una vez por hora), fechas y `usermodified`.	sitekey único; active
`local_tiendacursos_mailq`	Cola de reintento de correos (tipo, intentos, último error, próximo intento). Sin contraseñas.	timenext

Las consultas del panel están paginadas (30 por página, `ui::PERPAGE`; las tablas que traen todas sus filas también muestran 30 por página en el navegador, `amd/src/tables.js`) y precargan cursos y conteos en una sola consulta (sin N+1). La exportación se transmite con `\core\dataformat` y un recordset.

---

6. Pasarelas

Flow (API v2)

- Base: integración `https://sandbox.flow.cl/api`, producción `https://www.flow.cl/api`.
- Firma: parámetros ordenados por nombre, concatenados `nombre+valor`, HMAC-SHA256 con la Secret Key, en `s`.

- `POST /payment/create` (commerceOrder = referencia, urlConfirmation, urlReturn, paymentMethod 9) → redirección a `url?token=…`.
- Verificación: `GET /payment/getStatus` (token) o `/payment/getStatusByCommerceId`. Estados 1 pendiente, 2 pagado, 3 rechazado, 4 anulado.
- Prueba de conexión: `getStatusByCommerceId` de una orden inexistente (solo lectura).

Mercado Pago (Checkout Pro)

- `POST /checkout/preferences` con external_reference, back_urls, auto_return y notification_url.
- Integración usa `sandbox_init_point`; producción `init_point`.
- Webhook (`type=payment&data.id`) o IPN (`topic=payment&id`) → `GET /v1/payments/{id}`; sin id se busca en `/v1/payments/search?external_reference=`.
- Prueba de conexión: `GET /users/me`.

Agregador o integración directa (1.5)

Flow y Mercado Pago son **agregadores**: un contrato con ellos y cobran con varios medios (Flow ofrece Webpay, Khipu, MACH…: es «Webpay vía Flow»). Desde 1.5 hay además **integraciones directas** con contrato propio del comercio: Webpay Plus (Transbank), Khipu, MACH, Getnet y PayPal. Cada una es una clase `gateway\<nombre>` que extiende `gateway\base` (`start`, `verify`, `order_from_notification`, `test_connection`, `return_hint`, `cancel`, `can_refund` / `refund`, `receipt`, `register_urls`, `sandbox_defaults`, `extra_settings`). Orden de `manager::NAMES` (el de la versión gratuita): flow, webpay, khipu, mach, getnet, paypal, mercadopago, transfer.

Pasarela	API	Cómo se confirma	Reembolso des
Webpay Plus	REST v1.2 `/rswebpaytransaction/api/webpay/v1.2/transactions`; integración `webpay3gint.transbank.cl`, producción `webpay3g.transbank.cl`; cabeceras `Tbk-Api-Key-Id` / `Tbk-Api-Key-Secret`.	Sin aviso al servidor. En el retorno (`token_ws` por GET o POST) se hace `PUT …/{token}` (commit); si falla (ya confirmado), `GET …/{token}` (estado, 7 días). Pagado solo con `status=AUTHORIZED`, `response_code=0`, `buy_order` = referencia y monto completo. `TBK_TOKEN` (anulado) o solo `TBK_ORDEN_COMPRA` (tiempo agotado) → rechazado. La tarea de conciliación hace commit/estado de los pendientes.	`POST …/{token}` → REVERSED o NI response_code 0.
Khipu	Pagos instantáneos v3 `https://payment-api.khipu.com/v3`, cabecera `x-api-key` (mismo host para la cuenta de desarrollo y la real).	`POST /payments` (transaction_id = referencia, `notify_api_version 3.0`). Webhook JSON con `x-khipu-signature: t=…,s=…`, `s = Base64(HMAC-SHA256(t + "." + cuerpo, secreto))` sobre el cuerpo sin tocar; luego `GET /payments/{id}`: `status=done` (salvo status_detail rechazado/abuso/reversado/devuelto), moneda CLP, transaction_id y monto. Abandono (cancel_url) → `DELETE /payments/{id}`.	`POST /payments` recaudación en cl la rendición).
MACH	MACH para comercios: Sandbox `biz-sandbox.soymach.com`, producción `biz.soymach.com`, Bearer API Key, caja digital `terminal_id`.	`POST /payments` (upstream_id = referencia, idempotente con pago PENDING) → `payment_url` (QR o app). Webhook del portal (`business-payment-completed|failed|expired|reversed`) con token opcional en `Authorization` comparado con `hash_equals`; luego `GET /payments/{token}`: COMPLETED o CONFIRMED = pagado; FAILED/EXPIRED/CANCELED/REVERSED = rechazado.	COMPLETED (prin `/reverse`; CONF `/refund` con `id`
Getnet	Web Checkout (plataforma Placetopay/Evertec): pruebas `checkout.test.getnet.cl`, producción `checkout.getnet.cl`; `auth = {login, tranKey =`	`POST /api/session` (reference, CLP, returnUrl, cancelUrl, ipAddress, userAgent, expiración 1 h) →	`POST /api/reve` `internalRefere`

	`Base64(SHA-256(nonce + seed + secretKey)), nonce Base64, seed ISO 8601}`.	`processUrl`. Notificación única sin reintentos con `signature` = `sha256:` SHA-256(requestId + status + date + secretKey) (o SHA-1 sin prefijo); luego `POST /api/session/{requestId}`: APPROVED con referencia y moneda CLP. Un intento rechazado deja la sesión PENDING (el comprador puede reintentar en Getnet); la conciliación cubre el aviso perdido.	aprobado (probac pruebas).
PayPal	Orders v2 + OAuth2 client_credentials; sandbox `api-m.sandbox.paypal.com`, live `api-m.paypal.com`.	`POST /v2/checkout/orders` (intent CAPTURE, custom_id = referencia, monto convertido) → enlace `payer-action`; retorno `?token=ORDER_ID` → `GET` y, si APPROVED, `POST …/capture` con `PayPal-Request-Id`. Webhook verificado con `POST /v1/notifications/verify-webhook-signature` (cuerpo original insertado sin re-serializar). Pagado solo si la captura COMPLETED tiene exactamente la moneda y el monto guardados en el pedido.	`POST /v2/payments/ca`

PayPal y la moneda

PayPal no acepta CLP. (1.8: «pesos» = unidades de la moneda de la tienda; si la tienda está en la moneda de PayPal no hay conversión, `paypal::needs_rate()`; con la tienda fuera de CLP el automático usa open.er-api.com `rates[tienda]`, sin rangos fijos, y guarda `paypal_rateshop`.) Ajustes `paypal_currency` (USD, EUR, GBP, CAD o AUD), `paypal_ratemode` (`auto` | `manual`; 1.7.0), `paypal_rate` (tasa base, pesos por unidad), `paypal_margin` (0–15 %), `paypal_ratedate`, `paypal_ratesource`, `paypal_ratecurrency`, `paypal_ratecheck` y `paypal_rateerror`. `local\fxrate`: en modo automático la tarea diaria `task\update_fx_rate` (10:xx) y cada guardado de ajustes consultan mindicador.cl (dolar, euro) y, si falla, open.er-api.com; la tasa se acepta solo dentro del rango de la moneda (USD 500–2000, EUR 500–2200, GBP 600–2600, CAD 350–1500, AUD 300–1400) y con una variación ≤ 15 % frente a la anterior de la misma moneda; si no, se conserva la anterior y queda el motivo en `paypal_rateerror` (y en el registro `fx_rate`). Más de 3 días sin actualizar → aviso `warn_fx_stale` en Pasarelas. Tasa efectiva = base / (1 + margen/100). `paypal::convert()` = techo(CLP × 100 / tasa efectiva) en centavos (nunca cobra menos). Al ir a PayPal la orden lleva `experience_context.locale` según el idioma del aula (1.6.1). Al iniciar el pago el pedido guarda `paycurrency`, `payamount` (centavos) y `payrate`; `amount` sigue en CLP. El comprador ve el monto convertido junto a PayPal y en el resumen antes de pagar; Ventas muestra lo cobrado y el tipo de cambio.

Moneda de la tienda (1.8)

`local\currency`: ajuste `currency` (CLP por defecto; ARS, BOB, BRL, COP, CRC, DOP, GTQ, MXN, PEN, PYG, UYU, USD, EUR). Todos los montos (ofertas, packs, cupones de monto, pedidos) son enteros en la **unidad menor** de la moneda (CLP, COP y PYG sin decimales; el resto, centavos), así que los datos de 1.7 en CLP no cambian. Los formularios convierten con `currency::parse()` / `to_input()` (trait `form\money_fields`); `shop::money()` formatea con la moneda del pedido. `currency::CLP_ONLY` (Flow, Webpay, Khipu, MACH, Getnet) se excluye de `shop::enabled_gateways()` si la tienda no está en CLP. El feed de sitios externos incluye `currency` y `savings_formatted`.

Identidad del comprador (1.8)

`local\identity`: clave = RUT normalizado (Chile) o `{país}-{tipo}-{número}` en minúsculas (`pe-pasaporte-pe1234567`; tipos pasaporte, dni, cedula, otro); nunca colisionan. Es el `username` de las cuentas nuevas e `idnumber` = RUT con formato o el número del documento. Si algún curso del pedido tiene una instancia del bloque `block_sence`, todos los estudiantes deben ser chilenos (`identity::sence_course()`, error `err_sence_rut`).

Disponibilidad por país (1.8)

`base::operates()` (Flow/Webpay/Khipu/MACH/Getnet: CL; Mercado Pago: país de la cuenta leído en «Probar conexión», `mercadopago_accountcountry`; el resto: todos), ajustes `{gw}_countrymode` (home | all | list; por defecto home, salvo PayPal y Stripe: all) y `{gw}_countries` (CSV ISO). `available_for($country)` filtra en el checkout y en `orders::create()`; `country_warnings()` avisa en Pasarelas sin bloquear. País de la tienda: `shopcountry`.

Stripe (1.8)

`gateway\stripe`: Checkout Sessions alojado (POST /v1/checkout/sessions, `Stripe-Version: 2024-06-20`, `Idempotency-Key: cs-{referencia}`, monto en la unidad de Stripe: monedas sin decimales según su lista, p. ej. CLP; COP con 2). Verificación por API (GET de la sesión: `client_reference_id`, `payment_status`, monto y moneda). Webhook opcional con `Stripe-Signature`: HMAC-SHA256 de `t.cuerpo` con el `whsec`, tolerancia 300 s; aun firmado, el estado se lee de la API. Reembolso por `payment_intent`. «Probar conexión»: GET /v1/account, avisa clave de prueba en Producción y viceversa.

Credenciales públicas de prueba

Webpay y Getnet traen las credenciales públicas de integración que publican Transbank (código 597055555532) y Getnet (getnet.cl/developers): se usan solo en modo Integración y solo si el campo está vacío (`sandbox_defaults()`); nunca en Producción.

Reembolsos y abandono

Ventas › detalle de un pedido pagado muestra «Reembolsar el pago» cuando la pasarela tiene API (confirmación del navegador, POST + sesskey + permiso Editar en Ventas, lock por pedido). El pedido pasa a `refunded` (nuevo estado) y ningún aviso posterior lo reabre; la matrícula se conserva (el administrador decide). Flow y Mercado Pago: reembolso desde su panel. Anular un pedido pendiente o el abandono del comprador cierran también el pago en la pasarela cuando la API lo permite (Khipu, MACH, Getnet).

Transferencia

No es en línea: el pedido queda pendiente hasta que un administrador confirma el pago en el panel.

Cada pasarela guarda su modo en `<gw>_mode` (sandbox|production) y sus credenciales en `<gw>_<campo>_sandbox` y `<gw>_<campo>_prod`. Desde 1.5 los campos secretos se guardan **cifrados** con `\core\encryption` (clave del sitio en `$CFG->dataroot/secret/key`); la actualización cifra los guardados en texto plano por versiones anteriores. Si la clave del sitio cambia o se pierde (por ejemplo al migrar el moodledata sin `secret/`), los secretos no se pueden leer: la pasarela aparece «Faltan datos» y hay que escribirlos de nuevo. El cliente HTTP (`gateway\http`, GET/POST/PUT/DELETE) es reemplazable en pruebas. notify.php entrega a cada pasarela el cuerpo original (`_raw`) y las cabeceras (`_headers`) para verificar firmas.

Fuentes oficiales

- Transbank: transbankdevelopers.cl › Referencia Webpay y Documentación Webpay Plus (fuente en github.com/TransbankDevelopers/transbank-developers-docs) y el SDK oficial github.com/TransbankDevelopers/transbank-sdk-php (endpoints v1.2 y credenciales de integración).
- Khipu: docs.khipu.com › API Pagos Instantáneos v3.0 (OpenAPI), Notificaciones (Webhooks), Autenticación y Pruebas y ambientes.
- MACH: docs.business-api.soymach.com (MACH para comercios API 1.1.0: BusinessPayments, Webhooks, Sandbox).
- Getnet: getnet.cl/developers (ambiente de pruebas, credenciales y tarjetas de prueba; manual de integración API) y docs.placetopay.dev › Checkout (Authentication, Session API, Notification).
- PayPal: developer.paypal.com › Orders v2, Payments v2, Webhooks v1 (verify-webhook-signature), Authentication; especificaciones OpenAPI en github.com/paypal/paypal-rest-api-specifications.

---

7. Cuentas, matrícula y renovación

- Usuario = RUT normalizado en minúsculas sin puntos (`12345678-5`) o, desde 1.8, la clave del documento extranjero, auth manual, idnumber = RUT con formato, institución = empresa, `auth_forcepasswordchange`.
- Estado de acceso (`local\access`): activo, vencido (timeend ≤ ahora), suspendido (matrícula o instancia deshabilitada con fecha vigente) o sin matrícula; considera cualquier método de matriculación.
- Renovación: `update_user_enrol(instancia existente, ACTIVE, ahora, ahora + días)`, habilita la instancia si estaba deshabilitada y reasigna el rol estudiante si faltaba. No crea matrículas duplicadas.
- Packs (`orders::enrol_until()`): la fecha de término del estudiante se fija una vez (`student.timeend`) y se aplica a cada curso del snapshot `packcourses`: vencido → se renueva la misma matrícula; vigente → se conserva y solo se extiende si terminaba antes (nunca se acorta; sin término queda sin término); sin matrícula → matrícula manual; suspendido → error del estudiante (reintentable desde Ventas). Reintentar da las mismas fechas. Validación previa: suspendido en algún curso o vigente en todos = no puede comprar.
- Precio del pack: oferta si `offerprice` > 0 y (sin término o ahora ≤ `offeruntil`), si no la suma. Tramos: los generales (courseid 0). Cupones: solo los «para todos los cursos» (`coupon_notpack`).

- Se vende con licencia de pago, activo, ≥ 2 cursos visibles y precio > 0 (`packs::sellable()`, verificado también en `orders::create()`).

8. Correo

- `local\transport`: `email_to_user` (respeta smtphosts, noreply, divertallemailsto) o SMTP propio con `moodle_phpmailer` (Date, Message-ID con el dominio del remitente, HTML + texto, Reply-To).
- `local\mailer`: plantillas con marca del sitio (logo, nombre y `brandcolor` del tema; respaldo Boost). Si el envío falla, cola con reintento escalonado (15 min × intentos, máx. 6 h, 12 intentos).
- `local\templates` (1.2): 9 tipos (`welcome_new`, `welcome_existing`, `renewal`, `receipt`, `transfer`, `admin_sale`, `admin_pending`, `expiry_soon`, `expired`) con destinatario, si es opcional y si requiere licencia. Sin fila en `_tpl` se usa la cadena del idioma del destinatario (`get_string_manager()->get_string(..., $lang)`), escrita con variables `{nombre}` al pasarle un objeto de marcadores. Variables permitidas por tipo y obligatorias validadas al guardar (texto limpio con `clean_text`); los valores se escapan con `s()`; un `href` que termina en una variable (también `%7Bvar%7D` que deja el editor) recibe el valor exacto. Los opcionales desactivados no se envían ni se reintentan.
- `local\expiry` + tarea `send_expiry_notices` (diaria 9:00, solo con licencia): candidatos = estudiantes matriculados por la tienda con término entre hace 3 días y hoy + máx(días); el término real se lee de `user_enrolments` (en packs, el menor de sus cursos). Envía el umbral más pequeño alcanzado y registra todos los mayores; «vencido» solo dentro de 3 días; registro único por persona, curso/pack, tipo y término (idempotente). Sin cola: si falla, se intenta al día siguiente.

9. Integración con Moodle

Elemento	Detalle
Panel	Páginas propias con el formato del panel de Registro de avances (`templates/panel*.mustache`, `classes/local/ui.php`, AMD `tables` y `navscroll`): sales, courses, coupons, tiers, gateways, mail, config, users, activate, docs; 1.2: packs y templates. `panel.php` abre la primera sección permitida. En Administración del sitio quedan solo enlaces al panel, la licencia y la documentación.
Permisos	`classes/local/panel.php` sobre la clase común `norteduc_panel_perms`. Secciones y nivel de un usuario nuevo: sales read, courses read, packs read, coupons read, tiers read, gateways none, mail none, templates read, config none. Actualización a 1.2: templates = edit si mail era edit, si no read; packs = nivel de courses. La capacidad `local/tiendacursos:manage` de 1.0 se eliminó; en la actualización sus titulares (no administradores) pasan a usuarios del panel con «Solo lectura» en todo. Bloque: `addinstance`, `myaddinstance`.
Eventos	`\local_tiendacursos\event\order_paid`, `student_enrolled`, `credentials_sent`, `panel_user_updated` (alta, cambio o baja de un usuario del panel, con sus permisos).
Tareas	`reconcile_orders` (*/15), `retry_mail` (*/10), `check_updates` (diaria), `send_expiry_notices` (diaria 9:00), `update_fx_rate` (diaria 10:xx, solo con tipo de cambio automático; sin datos personales).
Observadores	`user_deleted` (acceso al panel y avisos), `course_deleted` (sale de los packs y se borra su oferta).
Archivos	`local_tiendacursos_pluginfile()` sirve la imagen de los packs (pública, como la vitrina).
Hooks	`core\hook\navigation\primary_extend` (menú), `core\hook\output\before_footer_html_generation` (catálogo en portada).
Estado del sistema	`core\check` «Checkout para Moodle» (código no válido, varios medios en versión gratuita, pasarelas ofrecidas, modo, última prueba, cron, pendientes, correo, nueva versión).
Privacidad	Proveedor completo: metadatos de pedidos, estudiantes, usuarios del panel, avisos de vencimiento, plantillas editadas (`usermodified`) y envío a Flow/MP; exporta, anonimiza los pedidos (los montos se conservan para contabilidad) y borra el acceso al panel. El bloque no guarda datos.
Plantilla	`local_tiendacursos/course_cards`, compartida por vitrina, bloque y portada; 1.10: `local_tiendacursos/browse` (barra lateral) la envuelve.
Idiomas	es y en completos.
Copia de seguridad de cursos	La oferta, los tramos por curso y los pedidos viven en tablas de la tienda y **no viajan** en el respaldo del curso: al restaurar en otro sitio, vuelve a configurar la oferta. Las matrículas sí viajan como matrículas manuales normales.
Desinstalación	Solo borra sus tablas y ajustes; las cuentas y matrículas creadas se conservan.

10. Licencia, planes y actualizaciones

Código común de los complementos Norteduc (`norteduc_license`, `norteduc_panel_perms`, `norteduc_updates`, copias idénticas verificadas con `shared/sync.py --check`). Código de activación Ed25519 de un solo dominio (producto `tc`), activado una vez en línea con comprobante firmado.

Plan	Payload	Comportamiento
Gratis	sin código, o código inválido / de otro dominio / anual vencida tras la gracia	Nunca cerrada: 1 medio de pago (el primero activo en el orden Flow, Mercado Pago, transferencia), 1 estudiante por compra, sin cupones ni tramos, solo el administrador del sitio. Los datos de pago se conservan inactivos.
Anual	`plan=anual`, `exp=AAAA-MM-DD`	Todo; aviso en el panel desde 30 días antes y correo a `get_admins()` 30, 15, 3 y 1 día antes (tarea diaria `check_updates`, una vez por hito y fecha: `nlic_reminded_<exp>_<n>`; solo el más cercano si se saltaron), 15 días de gracia; actualizaciones mientras esté vigente.
Perpetua	`plan=perpetua` o sin `plan` (códigos TC-… anteriores a 1.1)	Todo, sin vencimiento; actualizaciones para siempre.

Las funciones bloqueadas se muestran con `lock_html()` («Comprar licencia» → desde 1.8 https://www.norteduc.cl/complementos-moodle/checkout/#licencias, fija en el código). Página Licencia (`license::plan_button()`, 1.8): gratis → «Ver licencias anual y perpetua»; anual → «Renovar licencia anual» y «Pasar a licencia perpetua» (`license::UPGRADE_URL` = https://www.norteduc.cl/producto/mejora-licencia-perpetua-checkout-para-moodle/?dominio={host}); perpetua → ninguno. La mejora se valida en norteduc.cl contra `POST /api/mejora/verificar` de licencias.norteduc.cl (anual vigente o en gracia del mismo dominio) y al pagarse el servidor vuelve a firmar la misma licencia como perpetua (mismo `lid` y dominio, código nuevo; la activación existente sigue válida). «Contactar a soporte» (`support_button()`, mailto contacto@norteduc.cl con asunto y datos técnicos) va en el pie del panel y en Licencia, en todas las versiones. La tarea `check_updates` lee `https://licencias.norteduc.cl/version/tc.json` y el panel muestra el aviso de nueva versión (descarga POST con el código si la licencia está vigente; enlace `download` público en la gratuita; «renueva» si la anual venció).

11. Catálogo para sitios externos (1.3)

Clase `\local_tiendacursos\local\sites` (política de acceso, claves, feed, caché, límite) y la sección del panel `sites.php` (permiso de sección `sites`, nuevo = «Sin acceso»; la 1.3 no migra niveles). Función de pago `sites` en `license::PAID_FUNCTIONS`.

Punto de entrada	Qué hace
`/local/tiendacursos/feed.php[?k=CLAVE]`	JSON público (GET/HEAD/OPTIONS; otro método → 405). Cabeceras: `ETag`, `Last-Modified`, `Cache-Control: public, max-age=120`, `Vary: Origin`, `X-Content-Type-Options: nosniff`, `X-Robots-Tag: noindex`; 304 con `If-None-Match` / `If-Modified-Since`. Errores JSON: 404 `disabled`, 403 `origin` / `badkey` / `keyrequired`, 429 `ratelimit` (`Retry-After: 60`).
`/local/tiendacursos/embed.js`	Script estático sin dependencias: lee los atributos `data-` (key, container, type, category, featured, limit, color, text, bg, font, radius, minwidth, target, button, heading y, desde la 1.4, style = compact|full; sin style usa `site.card_style` del feed; desde la 1.4.1, filters = 1|0; desde la 1.6, layout = sidebar|top, sticky-top y max-width). Con `layout=sidebar` el buscador y las categorías van en una barra lateral pegajosa; el diseño responde al ancho del bloque (container queries `tcw`/`tcm`, con respaldo por ResizeObserver): barra lateral desde 768 px de ancho del bloque y tarjetas de a 3, 2 o 1 por fila según el ancho del área de tarjetas (760 / 480 px). Espera a que el bloque esté cerca de verse (IntersectionObserver), pide el feed con `credentials: 'omit'` y pinta tarjetas en un *shadow root*. Colores y fuentes se validan (sin `url(`, `;` ni llaves). Si el feed falla, muestra un enlace a la vitrina y un aviso en la consola. El generador del panel (`sites::codes()`) agrega `?v=<version>` (`sites::embed_version()` = `$plugin->version` instalada) para que, tras una actualización, el navegador descargue el script nuevo en vez del guardado en caché; la vista previa añade `&p=`

	`<marca de tiempo>`. Con dos contenedores de igual id en la página, el segundo script ya no reemplaza al primero (1.6).
`/local/tiendacursos/embed.php[?k=&cat=&type=&featured=&limit=&target=_top]`	Vitrina con `pagelayout('embedded')`, sin cookies; enlaces con `target=_blank rel=noopener` (o `_top`). Publica su alto al padre con `postMessage({tiendacursos:'height', h})`; el código del iframe solo escucha mensajes cuyo `origin` es el Moodle.
`/local/tiendacursos/image.php?course=ID|pack=ID&v=T`	Portada pública de un curso o pack a la venta (sirve aunque el sitio use `forcelogin`), caché pública inmutable.

Formato del feed (versión 1)

```
{
  "version": 1,
  "site": {"name": "...", "url": "https://aula.example.com/", "showcase": ".../local/tiendacursos/index.php",
           "currency": "CLP", "locale": "es-CL", "card_style": "compact"},
  "labels": {"buy": "Comprar", "pack": "Pack", "hours": "{n} horas", "days": "Acceso por {n} días", ...},
  "items": [
    {"type": "pack", "id": 1, "name": "...", "category": "", "summary": "texto plano (≤300)", "hours": null,
     "access_days": 90, "price": {"currency": "CLP", "normal": 189700, "offer": 149900, "offer_until": "2026-12-31T23:59:00-
     "current": 149900, "formatted": "$149.900", "formatted_normal": "$189.700"}, "savings": 39800, "featured": true,
     "courses": [{"id": 5, "name": "..."}], "image": ".../image.php?pack=1&v=...", "url": ".../checkout.php?pack=1",
     "info_url": ".../course.php?pack=1"},
    {"type": "course", "id": 3, "category_id": 2, ...}
  ]
}
```

Primero los packs (como en la vitrina) y luego los cursos; los packs traen `category_ids` (categorías de sus cursos) para filtrar igual que la vitrina. El feed y el iframe usan siempre el idioma del sitio (`force_current_language($CFG->lang)`), así la caché compartida nunca queda en el idioma de un visitante. Caché MUC `feed` (60 s, clave por idioma) que se purga al guardar ofertas, packs o los ajustes; el ETag depende solo del contenido y `Last-Modified` es la hora en que cambió ese ETag (ajuste `ext_lm_feed_<idioma>`), así también cambia al borrar un curso, renombrar una categoría o cambiar una portada. Caché MUC `ratelimit`: una entrada por IP (`ventana:contador`, ventana de un minuto, sobrescrita; no crece); el conteo no es atómico (tolerancia aceptable para un límite anti-abuso). Detrás de un proxy o CDN hay que configurar `getremoteaddrconf` / `reverseproxy` para que `getremoteaddr()` entregue la IP real. El 429 se envía después de las cabeceras CORS para que la página autorizada pueda leerlo.

Reglas de acceso (`sites::decide()`)

1. Publicación apagada o sin licencia → 404 `disabled`; `frame-ancestors 'self'`.
2. `Origin: null` → 403.
3. Con `k`: debe ser la clave de un sitio activo; si llega `Origin`, debe ser ese sitio (con o sin «www.», mismo esquema y puerto).
4. Sin `k` y `ext_requirekey` → 403. Sin `k` ni exigencia: con `Origin`, debe ser un sitio activo; sin `Origin` (servidor, curl) se responde.

1.4 agrega al feed, sin cambiar su versión (campos nuevos, compatibles): `info_url` en cada ítem, `site.card_style` y `labels.info`.

11b. Tarjeta compacta y ficha pública del curso (1.4)

- **Tarjeta**: plantilla `local_tiendacursos/course_cards` (vitrina, portada, bloque, iframe) con la variable `compact` que entrega `shop::cards_context()` según el ajuste `cardstyle` (compact por defecto, también en sitios que actualizan; full = tarjeta de la 1.3). Cada tarjeta trae `infourl` (`shop::info_url()`) y `savings`. El resumen solo se calcula con el estilo full. El bloque no cambia: usa la misma plantilla.
- **Ficha**: `course.php?id=ID` o `?pack=ID`, pública (sin `require_login`, como la vitrina y el checkout), contexto de sistema, layout `standard` sin encabezado del tema (el nombre del curso es el único h1). Datos en `\local_tiendacursos\local\coursepage`: solo cursos de `shop::catalogue()` (oferta visible con precio, curso visible) y packs de `packs::catalogue()` (licencia de pago); en otro caso, HTTP 404 sin datos del curso.
- **Descripción**: `course.summary` con `file_rewrite_pluginfile_urls()` (área `course/summary`) y `format_text()` en el contexto del curso (filtros incluidos). Se eligió el resumen de Moodle como única fuente: el plugin no guarda texto propio por curso y así el cliente mantiene un solo texto. Los packs usan su descripción propia. *Nota:* con `forcelogin`, las imágenes del resumen pasan por `pluginfile.php` y piden sesión (igual que las portadas de la vitrina); el Open Graph usa `image.php` cuando la publicación externa está activa.

- **Temario**: nombres de `course_sections` con `section > 0`, `visible = 1`, nombre no vacío y sin `component` (subsecciones delegadas de 4.5+); se muestra con 2 o más. Ajuste `ficha_syllabus`. Nunca se publica el contenido de las secciones.
- **SEO**: `coursepage::head_html()` agrega (solo en esa petición, vía `$CFG->additionalhtmlhead`) meta description (≤160), canonical, Open Graph, twitter:card y JSON-LD `Course` (pack: `Product`) con `offers` en CLP y `courseWorkload`; JSON con `JSON_HEX_TAG` para que no pueda cerrar el script.
- **Móvil**: el recuadro de precio pasa bajo el título; barra fija inferior con precio y botón que se oculta mientras el recuadro o la llamada final están en pantalla (IntersectionObserver; sin JS queda visible).
- **Licencia**: tarjeta y ficha son de la versión gratuita (lo público del catálogo base es gratis); la ficha de packs, igual que los packs, requiere licencia de pago.

11d. Buscador y filtros en barra lateral (1.10)

- **Dónde**: vitrina (`index.php`, id `ltcb-showcase`), catálogo de la portada (`hook_callbacks::before_footer()`, id `ltcb-frontpage`) y bloque (`block_tiendacursos` 1.3.0, id `ltcb-block-<instancia>`). Todos llaman a `\local_tiendacursos\local\browse::render($ctx, $place, $options)`, que devuelve la plantilla `course_cards` sin cambios cuando la barra no aplica (desactivada para el lugar, bloque con `filters = 0`, formato lista o menos de 2 tarjetas) y si no `local_tiendacursos/browse` + el módulo AMD `local_tiendacursos/browse`.
- **Datos por tarjeta**: `shop::cards_context()` y `shop::pack_cards()` agregan con `browse::card_data()` categorías (`f_cats`; packs: las de sus cursos), precio en unidad menor (`f_price`), descuento (`f_offer`), horas (`f_hours`; packs: suma), fecha de creación (`f_created`), tipo (`f_kind`) y etiquetas (`f_tags`, `core_tag_tag::get_items_tags()` en una consulta, solo con `usetags` y el filtro activo). `browse::with_text()` arma `f_text` (nombre, categorías, etiquetas, cursos del pack y 300 caracteres del resumen en texto plano; una consulta). La plantilla de tarjetas los imprime como atributos `data-*` escapados solo cuando `browse` es verdadero.
- **Facetas** (`browse::facets()`): categoría con ≥ 2 categorías; precio con `price_ranges()` (≥ 2 precios distintos; cortes en los cuartiles redondeados hacia arriba a 2 cifras con `nice()`, máximo 4 tramos, valor `lo-hi` con lo exclusivo, hi inclusivo, hi vacío = sin tope); duración con `hour_ranges()` (8/20/40/80 h, solo tramos con cursos, ≥ 2); ofertas cuando algunos y no todos tienen descuento; tipo cuando hay cursos y packs; etiquetas (las 20 más usadas). Cada una se apaga con `browse_f_<filtro>`.
- **Navegador** (`amd/src/browse.js`, sin dependencias): filtra ocultando tarjetas (`hidden`), ordena moviendo los nodos, recalcula el conteo de cada opción aplicando los demás filtros, búsqueda normalizada (NFD sin diacríticos, todas las palabras) con espera de 220 ms, chips de filtros activos, contador `aria-live`, estado vacío. Con `urlsync` (vitrina y portada) lee y escribe `q, cat, price, hours, offer, type, tag, sort` con `history.replaceState` (valores validados contra las opciones existentes). El ancho del bloque (`ResizeObserver`, ≥ 720 px) decide entre barra lateral `position: sticky` (tope = barra fija del tema + 16 px, o 16 px si un contenedor del tema es el que se desplaza; los ancestros con `overflow` que no se desplazan pasan a `overflow: clip` para no anular el sticky) y cajón: el panel pasa a `role="dialog"` + `aria-modal`, se mueve al final de `body` mientras está abierto (escapa de cajones y contextos de apilamiento del tema), atrapa el foco, se cierra con Esc, la ×, el fondo o «Ver N resultados» y devuelve el foco al botón. Las tarjetas usan una container query (`ltcbmain`: 1 columna, 2 desde 440 px, 3 desde 700 px).
- **Ajustes** (`setup::DEFAULTS`, todos 1): `browse_showcase`, `browse_frontpage`, `browse_block`, `browse_f_search|category|price|hours|offer|type|tags|sort`. Bloque: `config_filters` ('' = ajuste general, '1', '0'). `frontpage_count` y la cantidad del bloque aceptan 0 = todos (tope 200). La vitrina agrega la clase `ltc-shop-wide` al body para ensanchar `.main-inner` en Boost (1320 px).
- **Sin JavaScript** la barra no se muestra y quedan todas las tarjetas. Sin cambios de base de datos (el paso de actualización 2026100700 solo escribe los ajustes nuevos).

11e. Portadas del catálogo: carga diferida y WebP (1.11)

En la vitrina, el catálogo de la portada y el bloque, cada tarjeta pinta la portada del curso. La imagen de presentación del curso (`course/overviewfiles`) suele ser grande (1200 px, 0,1–1,3 MB), y como la tarjeta mide unos cientos de píxeles, descargarla a tamaño completo era el principal costo del catálogo.

- **Carga diferida.** La portada de la tarjeta pasó de ser un fondo CSS (que no admite `loading` y siempre descarga el original completo) a un `<img class="ltc-cover-img" loading="lazy" decoding="async" width="600" height="338">` con `object-fit:cover` dentro de `.ltc-cover` (mismo recorte 16:9 de antes, sin desplazamiento de contenido). Las portadas fuera de la pantalla no se descargan hasta acercarse. La **ficha** del curso (`course.php`) mantiene su portada grande en `eager` (solo se le añadió `decoding="async"`): no es el problema y conviene que cargue de inmediato.
- **Derivado redimensionado y en WebP** (`local\covers`). Se genera una copia de la portada de a lo más `MAXWIDTH = 600` px (sin ampliar nunca), en **WebP** con calidad 78 cuando el servidor tiene `imagewebp()`, o en **JPEG** (calidad 82) / PNG si no. Si no hay GD, la tarjeta usa la URL del **original**: nunca se rompe en un Moodle sin GD o sin WebP.

- **Caché por contenthash.** El derivado se guarda con `file storage` en el área `cover` del contexto del sistema; el nombre es `<contenthash>-<ancho>.<ext>`. Al ser direccionable por contenido, se genera una sola vez, luego se lee de la caché y se **invalida solo** en cuanto cambia la imagen del curso. El `itemid` es el id del curso (los packs comparten el `itemid` 0); al regenerar un curso se borran sus derivados anteriores (no se acumulan). `shop::cards_context` / `pack_cards` entregan `image` (el derivado, para las tarjetas) e `imagefull` (el original, para la ficha); `course_image()` sigue devolviendo la URL del original para quien la use.
- **Servido** por `local_tiendacursos_pluginfile` (área `cover`): público como las portadas de la vitrina, con `Cache-Control: public, immutable, max-age=7776000` (es direccionable por contenido). Al crear el archivo se fija el `mimetype` explícito (`image/webp|jpeg|png`) y al servir se valida por prefijo `image/` en vez de `is_valid_image()`, porque algunas versiones de Moodle (4.5) no mapean la extensión `.webp` y la guardarían como `document/unknown`.
- **Seguridad.** El nombre lo da la API de archivos (sin travesía de rutas); solo se crean derivados de las portadas que la propia tienda muestra; se valida tipo y tamaño; contexto del sistema, como las imágenes de los packs.
- Sin cambios de base de datos (el paso de actualización 2026100702 es un savepoint; los derivados son una caché que se reconstruye sola).

Medición real (aula con 68 cursos, servidor con WebP): las portadas del catálogo pasaron de ~15 MB a ~1,1 MB (93 % menos); en otra aula, de ~5 MB a ~0,12 MB (98 %). Sin WebP (solo JPEG) la reducción típica ronda el 55 %. Verificado con Playwright: las portadas fuera de la pantalla no se descargan hasta desplazarse, sin peticiones a los originales, sin errores de consola y sin violaciones de accesibilidad propias.

11c. Suscripciones y membresías (1.9)

Tablas: `local_tiendacursos_plan` (kind course|membership, courseid, scope all|categories, categories csv, amount en unidad menor, currency, period month|quarter|year, trialdays 0–90, active, gwplans JSON con los ids de plan de cada pasarela por clave `gw:md5(precio, moneda, período, prueba, modo)`), `local_tiendacursos_sub` (reference SB…, planid, userid, datos del suscriptor, gateway, gatewayref, gwcustomer, status pending|trial|active|pastdue|suspended|cancelled|expired, amount/currency de la tienda y chargeamount/chargecurrency cobrados, withtrial, trialend, periodend, nextcharge, graceuntil, failcount, notified bitmask de avisos, lastsync, timeactivated/timecancelled/timeended), `local_tiendacursos_subpay` (cobros, único subid+gwpayid: idempotencia) y `local_tiendacursos_subcourse` (por curso: prevend −1 sin matrícula previa / 0 ilimitada / fecha, grantedend). Paso de upgrade 2026100690.

Núcleo: `local\subscriptions`. `create()` inserta la suscripción pendiente y llama a `sub_start()` de la pasarela (URL de aprobación). `returned()` (con lock) completa lo que la pasarela necesita al volver y `sync()` lee el estado de la API (`sub_fetch()` → estado normalizado + cobros) y `apply()` lo aplica: verifica plan, precio, moneda y monto de cada cobro (`mismatch()`: si no coincide no se aplica nada y queda `sub_mismatch` en el historial), registra cobros idempotentes, calcula gracia, otorga o devuelve acceso y envía cada aviso una sola vez. Una cancelación local nunca se deshace por una sincronización; un reembolso es definitivo.

Acceso: `access_end()` = fin del período (o de la prueba) + gracia (`sub_gracedays`, 7); morosa → `graceuntil`; cancelada → fin del período. `grant()` usa `orders::enrol_until()` (nunca acorta) y guarda prevend/grantedend; `revoke()` devuelve la fecha previa sin tocar matrículas ilimitadas ni extendidas por otra compra (±1 h). Suspender = fin de matrícula ahora: el avance no se borra.

Pasarelas (métodos `supports_subscriptions, sub_start, sub_return, sub_fetch, sub_cancel, sub_reactivate, sub_from_notification, sub_manage_url, sub_refund` en `gateway\base`):

- **Flow**: `plans/create` (planId estable por sitio/plan/precio/período/prueba/modo, urlCallback `notify.php?gw=flow&sub=1`), `customer/create` + `customer/register` (tarjeta), vuelta con token → `customer/getRegisterStatus` → `subscription/create`; estado con `subscription/get` (status, morose, invoices); cancelación `subscription/cancel at_period_end`; cambio de tarjeta con un nuevo `customer/register`. El aviso del plan no identifica la suscripción: se releen las suscripciones Flow abiertas. Solo CLP.
- **Mercado Pago**: `POST /preapproval` sin plan (auto_recurring, free_trial, back_url, status pending); estado con `GET /preapproval/{id}` + `/authorized_payments/search` (si falla, `summarized`); semáforo amarillo/rojo o paused → morosa; cancelación `PUT status cancelled`; avisos `subscription_preapproval` y `subscription_authorized_payment` con firma `x-signature` (HMAC-SHA256 de `id;request-id;ts`) si se configura la clave secreta (credencial `whsecret`); reembolso `POST /v1/payments/{id}/refunds`.
- **PayPal** (Subscriptions API v1): producto por modo (`paypal_subproduct_{modo}`), plan con ciclo TRIAL (DAY) + REGULAR (MONTH 1/3 o YEAR 1) a precio fijo convertido con `paypal::convert()` (tipo de cambio + margen), `payment_failure_threshold` 3; `POST /v1/billing/subscriptions` (custom_id = referencia, enlace approve); estado con `GET /v1/billing/subscriptions/{id}` + `/transactions`. En prueba = tiene ciclo TRIAL y ningún ciclo REGULAR completado (PayPal marca el ciclo TRIAL como completado al empezar; verificado en sandbox). SUSPENDED →

morosa; reactivar = `/activate`; cancelar `/cancel`; reembolso `/v1/payments/sale/{id}/refund`. Webhooks `BILLING.SUBSCRIPTION.*` y `PAYMENT.SALE.*` verificados con `verify-webhook-signature`. El monto en USD queda fijo por suscripción; un cambio de precio o tipo de cambio crea otro plan PayPal que solo usan las nuevas.

notify.php pregunta primero `subscriptions::notification()`; si el aviso es de suscripciones se sincronizan las suscripciones que nombra (el aviso nunca se cree por sí mismo) y responde OK. **Tareas:** `task\sync_subscriptions` (cada hora: sincroniza hasta 200, suspende al vencer la gracia, termina canceladas, cierra pendientes de más de 2 días, aviso de fin de prueba 3 días antes, sincroniza membresías) y `task\sync_memberships` (adhoc, al crear/actualizar cursos o categorías). `observer`: cuenta eliminada → cancela en la pasarela y anonimiza; curso eliminado → desactiva sus planes.

Páginas: `plans.php` y `subscriptions.php` (sección de permisos `subs`, licencia de pago, acciones POST con sesskey y permiso Editar; exportación Excel/CSV sin fórmulas), `subscribe.php` (público: campo trampa, tiempo mínimo, límite por visitante y 3 pendientes por correo al día; RUT de otra cuenta con otro correo exige iniciar sesión; datos del resultado solo para la sesión que la inició), `mysubscriptions.php` (solo las del usuario; enlace en el menú de usuario con el hook `core_user\hook\extend_user_menu`). Feed JSON: ítems `type: "plan"` (kind, period, trial_days, courses_count, category_ids); `embed.js` acepta `data-type="plans"`.

Privacidad: metadata de sub/subcourse y de datos enviados a PayPal; exportación de suscripciones con sus cobros; borrado = cancelar + anonimizar.

---

12. Pruebas y calidad

- 1.9: PHPUnit 159 pruebas en Moodle 4.5 LTS y 5.0 (nuevo `subscriptions_test`: ciclo completo con Flow simulado — tarjeta, prueba, cobro, cobro fallido con gracia, suspensión sin perder la matrícula, reactivación, cancelación al fin del período y término—, prueba gratis una vez por persona, membresía que sigue sus categorías sin acortar matrículas ilimitadas, rechazo de plan/monto distintos, PayPal simulado con conversión, plan único por precio, reembolso definitivo y estados, Mercado Pago (estados y firma x-signature), MRR y tasa de cancelación, versión gratuita y cuenta eliminada). **Punta a punta real en PayPal Sandbox** (aula demo, Moodle 4.5): suscripción mensual aprobada con la cuenta personal de pruebas → activa, cobro USD 10,23 registrado, webhooks firmados recibidos y verificados, cancelación desde «Mis suscripciones» (CANCELLED en PayPal, acceso hasta el fin del período), reembolso del cobro desde el panel; y membresía con 7 días de prueba → estado «En prueba». Flow y Mercado Pago: probados con respuestas simuladas de la API (sin credenciales de prueba de suscripciones disponibles). Playwright en 1366 y 390 px: vitrina, ficha, suscripción, Mis suscripciones, Planes, formulario, Suscripciones y detalle con axe 0 infracciones (con un color de marca accesible; el verde del tema del aula demo no alcanza contraste AA en sus botones, igual que en las páginas existentes).
- 1.8: PHPUnit 151 pruebas en Moodle 4.5 LTS y 5.0 (nuevo `intl_test`: monedas y conversión de montos, claves de identidad, pedido de un comprador extranjero, regla SENCE, pasarelas por país y moneda, PayPal con la tienda en otra moneda, Stripe con respuestas simuladas y firmas válidas/alteradas/vencidas, antispam y estudiante por revisar). Prueba de punta a punta en Moodle 5.0 con la tienda en soles: comprador de Perú con pasaporte, PayPal Sandbox en USD con tipo de cambio automático, matrícula de la cuenta `pe-pasaporte-…` y reembolso. Checkout y Pasarelas con axe 0 infracciones en 1366 y 390 px y sin scroll horizontal.
- 1.5: PHPUnit nuevo `direct_gateways_test` con respuestas simuladas de cada API (Webpay, Khipu, MACH, Getnet, PayPal: creación, verificación de estado/referencia/monto/moneda, firmas válidas y alteradas, idempotencia, abandono, reembolsos, prueba de conexión, cifrado de secretos, regla de la versión gratuita, conversión de PayPal). Pruebas reales de punta a punta en el ambiente de integración de **Webpay Plus** (tarjeta aprobada, rechazada, anulación, segundo retorno, reembolso REVERSED) y en el ambiente de pruebas de **Getnet** (aprobada, rechazada, reversa), con el navegador volviendo al Moodle de pruebas. Khipu, MACH y PayPal piden cuenta de comercio (cuenta de desarrollo / registro en el portal / cuenta developer) y quedaron probados con simulaciones.
- 1.4: PHPUnit 120 pruebas en Moodle 5.0 (4 nuevas en `coursepage_test`: estilo de tarjeta, ficha de curso con temario, matriculado y cursos que no se venden, ficha de pack con licencia y sin ella, etiquetas SEO escapadas; feed con `info_url`). Playwright en 1366 y 390 px: vitrina, portada, bloque, fichas (curso, curso corto, pack, 404), iframe y sitio externo con axe 0 infracciones (WCAG 2.1 AA + buenas prácticas), sin scroll horizontal ni errores de consola.
- 1.3: PHPUnit 106 pruebas en Moodle 4.5 LTS y 5.0 (13 nuevas en `sites_test`: normalización de direcciones, «www.», decisión CORS, claves y exigencia, versión gratuita, `frame-ancestors`, contenido del feed sin cupones, tramos, cursos ocultos ni compradores, ETag, 304, límite por IP, validación al guardar, permisos de sección y códigos generados). Prueba real con un sitio en otro origen (feed, código para pegar, iframe) y compra hasta confirmar el pago; un origen no autorizado queda bloqueado por CORS y `frame-ancestors`. Actualización 1.2 → 1.3 en 4.5 y 5.0.
- 1.2: PHPUnit 90 pruebas + 2 del bloque en Moodle 4.5 LTS y 5.0: packs (precios, reglas de venta, tarjetas, compra grupal con tramo y cupón, misma fecha en todos los cursos, un correo, extensión y renovación, filtros, borrar/inactivar), plantillas (texto original = cadenas, validación de variables, texto propio usado y sin contraseñas guardadas, activar y desactivar, prueba, migración de permisos) y avisos (secuencia 7/1/vencido, sin repetir, hueco,

renovación, pack, gratis, privacidad). Actualización 1.1 → 1.2 probada en 4.5 (gratis) y 5.0 (licencia, usuarios del panel) sin pérdida de datos.
- 1.1: PHPUnit 71 pruebas en Moodle 4.5 LTS y 5.0 (las de 1.0 más: permisos por sección, solo el administrador gestiona usuarios, roles de Moodle sin acceso, versión gratuita sin usuarios del panel, 1 medio de pago, 1 estudiante, cupones y tramos ignorados pero conservados, vencimiento anual con gracia, ayudantes de actualización, aviso de nueva versión y las pruebas de las clases comunes de licencia) + 1 del bloque.
- Actualización probada desde 1.0.2 (con pedidos, cupones, tramos, credenciales y un gestor) en Moodle 4.5 y 5.0.
- moodle-cs (moodle-extra): 0 errores.
- Herramientas de desarrollo (datos demo) en `dev/`, fuera del ZIP.

Novedades 1.16 — modalidad, fechas, credenciales e iframes por modalidad

- **Modalidad de la oferta**: columna `modality` (elearning|presencial|semipresencial) en `local_tiendacursos_course`; `shop::offer_modality()` la normaliza, `quotes::MODALITIES` suma `semipresencial` (también en el formulario de cotización). El formulario de la oferta reutiliza `quotes::modality_options()`. Cualquier modalidad admite cualquier modo de venta (`mode`).
- **Fechas presenciales**: columnas `startdate` / `enddate` (date_selector opcional). Se muestran en la ficha (`coursepage`) para presencial/semipresencial; `courseMode` de schema.org = online/onsite/blended.
- **Credenciales**: columna `sendcredentials` (1/0, def. 1; el formulario la oculta y la deja en 1 para e-learning). `shop::sends_credentials()`. En `mailer::welcome()`, si ningún curso del pedido envía credenciales, se usa la plantilla `enrol_presencial` (sin contraseña, sin evento `credentials_sent`); la matrícula no cambia.
- **Ficha según compra**: las líneas de acceso/matrícula y la tarjeta CTA dependen de `showbuy` (si el curso se compra), no de la modalidad; un curso que solo se cotiza las omite.
- **Iframes por modalidad**: `embed.php` acepta `mod=presencial|online|elearning|semipresencial` (con filtro de modalidad active se muestran cursos, no packs); `shop::catalogue()` filtra por `o.modality`; `sites::codes()` acepta `mod` y el panel genera dos iframes extra (presencial / online).
- **Upgrade**: savepoint `2026103000` agrega las cuatro columnas (aditivo, sin pérdida).

Novedades 1.17 — código SENCE por curso y temario PDF con captura de leads

- **Código SENCE por oferta**: columna `sencecode`; `shop::offer_sence($offer,$idnumber)` devuelve el campo si está, si no el método heredado (idnumber + ajuste global). La ficha lo muestra desde ahí.
- **Temario PDF**: área de archivos `local_tiendacursos / syllabus` (contexto del curso, itemid = courseid, 1 PDF). El formulario de la oferta usa un `filemanager`; `courses.php` hace `file_prepare_draft_area` / `file_save_draft_area_files`. `shop::syllabus_file()`.
- **Descarga**: endpoint público `syllabus.php?id=`. En modo `free` entrega el PDF directo; en modo `lead` muestra un formulario (nombre, apellido, correo, empresa) con anti-spam (`antispam` honeypot + marca de tiempo + `too_many_leads()`), guarda el lead en `local_tiendacursos_lead` con un `token` de un solo uso, envía el correo con el PDF adjunto y redirige a la página de gracias que inicia la descarga con ese token. `pluginfile` sirve el área `syllabus` solo a usuarios del panel (vista previa del formulario).
- **Correo con adjunto**: `transport::send()` admite `$attachpath` / `$attachname` (ruta temporal bajo dataroot) tanto en modo Moodle (`email_to_user`) como SMTP (`addAttachment`); `mailer::syllabus_lead()` compone la plantilla `syllabus_lead` y adjunta el PDF.
- **Privacidad**: `local_tiendacursos_lead` declarada en el provider (datos de un visitante, sin userid). **Upgrade**: savepoint `2026103002` (dos columnas + tabla, aditivo).