

Checkout para Moodle

Documentación técnica · versión 1.3.1 (2026100231) · `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` $\geq$ 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

<code>checkout.php</code>	<code>orders::create()</code>	pedido "pending" + filas de estudiantes (transacción)
	└ <code>orders::start_payment()</code>	\$0 → <code>mark_paid</code> · transferencia → instrucciones · en línea →
pasarela	→ <code>server</code> → <code>notify.php</code>	<code>NO_MOODLE_COOKIES</code> , sin exigir <code>user-agent</code> ; GET+POST+JSON
	└ <code>gateway::order_from_notification()</code> → <code>gateway::verify()</code>	(consulta a la API, nunca confí)
	└ <code>orders::apply_verification()</code>	estado + monto $\geq$ pedido + referencia propia
	└ <code>orders::mark_paid()</code>	lock por pedido; idempotente
	└ primera vez: estado <code>paid</code> , uso de cupón, evento <code>order_paid</code> , comprobante, aviso a ve	
	└ <code>fulfil()</code> : por estudiante → <code>ensure_user()</code> → <code>enrol()</code> / renovación → evento <code>student_e</code>	
<code>result.php</code>	(regreso del navegador)	si sigue pendiente, vuelve a consultar la API; nunca matric
<code>task\reconcile_orders</code>	(cada 15 min)	consulta pedidos en línea pendientes (>5 min); expira: en lí

4. Seguridad

Riesgo	Control
Matrícula por un regreso falsificado del navegador	Solo <code>apply_verification()</code> con datos leídos de la API de la pasarela marca un pedido pagado.
Aviso falso a <code>notify.php</code>	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 <code>commerceOrder / external_reference</code> y la moneda.
Pago por menos	Si el monto informado es menor que el pedido, se registra <code>amount_mismatch</code> y no se matricula.
Avisos repetidos o concurrentes	Lock por pedido (<code>\core\lock</code>) + estado; matrícula idempotente por estudiante.
Avisos bloqueados	<code>notify.php</code> 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 <code>sesskey</code> .
Acceso al panel	<code>require_login</code> + <code>panel::require_section(\$seccion)</code> al inicio de cada página y <code>panel::require_section(\$seccion, 'edit')</code> antes de cada cambio (no basta con ocultar botones). Usuarios del panel y Licencia: solo <code>is_siteadmin()</code> . Los roles y capacidades de Moodle no dan acceso.
Versión gratuita	Los límites se verifican en el servidor: <code>shop::enabled_gateways()</code> (1 medio), <code>orders::max_students()</code> (1), <code>orders::quote()</code> (sin cupón ni tramo), <code>orders::create()</code> rechaza un medio no ofrecido o más de 1 estudiante, <code>panelusers::save()</code> 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 <code>ext_enabled</code> . CORS: se repite el <code>Origin</code> exacto solo si es un sitio autorizado activo (o el propio Moodle), nunca <code>*</code> ; origen no autorizado o <code>null</code> → 403. <code>Iframe</code> : <code>Content-Security-Policy: frame-ancestors 'self'</code> + sitios autorizados (o solo el de la clave) y se quita <code>X-Frame-Options</code> solo en <code>embed.php</code> ; <code>allowframing</code> no se toca. Sin cookies (<code>NO_MOODLE_COOKIES</code>), límite por IP y minuto, el feed contiene solo datos públicos de la vitrina y <code>embed.js</code> escribe todo con <code>textContent</code> y solo enlaza a URLs del mismo Moodle.
Entrada del visitante	RUT con dígito verificador, correo validado, textos limpiados y truncados.

5. Modelo de datos (XMLDB)

Tabla	Contenido	Índices
-------	-----------	---------

local_tiendacursos_course	Oferta por curso: precio, precio normal, horas, días de acceso, visible, destacado, orden.	courseid único
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).	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, error.	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,	kind único

	<code>usermodified</code> . Nunca contraseñas.	
<code>local_tiadacursos_notice</code> (1.2)	Avisos de vencimiento enviados: <code>userid</code> , curso o <code>pack</code> , tipo (<code>soonN</code> / <code>expired</code>), fecha de término avisada, fecha de envío.	<code>userid+courseid+packid+kind+timeend</code> único
<code>local_tiadacursos_site</code> (1.3)	Sitios externos autorizados: nombre, <code>origin</code> (<code>esquema://host[:puerto]</code> , normalizado), <code>sitekey</code> (24 hex, pública), activo, último uso (se escribe a lo más una vez por hora), fechas y <code>usermodified</code> .	<code>sitekey</code> único; active
<code>local_tiadacursos_mailq</code>	Cola de reintento de correos (tipo, intentos, último error, próximo intento). Sin contraseñas.	<code>timenext</code>

Las consultas del panel están paginadas (50 por página) 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` .

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` . El cliente HTTP (`gateway\http`) es reemplazable en pruebas.

7. Cuentas, matrícula y renovación

- Usuario = RUT normalizado en minúsculas sin puntos (12345678-5), 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. <code>panel.php</code> abre la primera sección permitida. En Administración del sitio quedan solo enlaces al panel, la licencia y la documentación.
Permisos	<code>classes/local/panel.php</code> sobre la clase común <code>norteduc_panel_perms</code> . 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 <code>local/tiendacursos:manage</code> de 1.0 se eliminó; en la actualización sus titulares (no administradores) pasan a usuarios del panel con «Solo lectura» en todo. Bloque: <code>addinstance</code> , <code>myaddinstance</code> .
Eventos	<code>\local_tiendacursos\event\order_paid</code> , <code>student_enrolled</code> , <code>credentials_sent</code> , <code>panel_user_updated</code> (alta, cambio o baja de un usuario del panel, con sus permisos).
Tareas	<code>reconcile_orders</code> (*15), <code>retry_mail</code> (*10), <code>check_updates</code> (diaria), <code>send_expiry_notices</code> (diaria 9:00).
Observadores	<code>user_deleted</code> (acceso al panel y avisos), <code>course_deleted</code> (sale de los packs y se borra su oferta).
Archivos	<code>local_tiendacursos_pluginfile()</code> sirve la imagen de los packs (pública, como la vitrina).
Hooks	<code>core\hook\navigation\primary_extend</code> (menú), <code>core\hook\output\before_footer_html_generation</code> (catálogo en portada).
Estado del sistema	<code>core\check</code> «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 (<code>usermodified</code>) 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	<code>local_tiendacursos/course_cards</code> , compartida por vitrina, bloque y portada.
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	<code>plan=anual</code> , <code>exp=AAAA-MM-DD</code>	Todo; aviso en el panel desde 30 días antes y correo a <code>get_admins()</code> 30, 15, 3 y 1 día antes (tarea diaria <code>check_updates</code> , una vez por hito y fecha: <code>nlic_reminded_<exp>_<n></code> ; solo el más cercano si se saltaron), 15 días de gracia; actualizaciones mientras esté vigente.
Perpetua	<code>plan=perpetua</code> o sin <code>plan</code> (códigos TC-... anteriores a 1.1)	Todo, sin vencimiento; actualizaciones para siempre.

Las funciones bloqueadas se muestran con `lock_html()` («Comprar licencia» → <https://www.norteduc.cl/tienda/#complementos>, fija en el código). «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_tiadacursos\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
<code>/local/tiadacursos/feed.php[?k=CLAVE]</code>	JSON público (GET/HEAD/OPTIONS; otro método → 405). Cabeceras: <code>ETag</code> , <code>Last-Modified</code> , <code>Cache-Control: public, max-age=120</code> , <code>Vary: Origin</code> , <code>X-Content-Type-Options: nosniff</code> , <code>X-Robots-Tag: noindex</code> ; 304 con <code>If-None-Match</code> / <code>If-Modified-Since</code> . Errores JSON: 404 <code>disabled</code> , 403 <code>origin</code> / <code>badkey</code> / <code>keyrequired</code> , 429 <code>ratelimit</code> (<code>Retry-After: 60</code>).
<code>/local/tiadacursos/embed.js</code>	Script estático sin dependencias: lee los atributos <code>data-</code> (<code>key</code> , <code>container</code> , <code>type</code> , <code>category</code> , <code>featured</code> , <code>limit</code> , <code>color</code> , <code>text</code> , <code>bg</code> , <code>font</code> , <code>radius</code> , <code>minwidth</code> , <code>target</code> , <code>button</code> , <code>heading</code>), espera a que el bloque esté cerca de verse (<code>IntersectionObserver</code>), pide el feed con <code>credentials: 'omit'</code> y pinta tarjetas en un <i>shadow root</i> . Colores y fuentes se validan (sin <code>url()</code> , ; ni llaves). Si el feed falla, muestra un enlace a la vitrina y un aviso en la consola.
<code>/local/tiadacursos/embed.php[?k=&cat=&type=&featured=&limit=&target=_top]</code>	Vitrina con <code>pagelayout('embedded')</code> , sin cookies; enlaces con <code>target=_blank</code> <code>rel=noopener</code> (o <code>_top</code>). Publica su alto al padre con <code>postMessage({tiadacursos:'height', h})</code> ; el código del <code>iframe</code> solo escucha mensajes cuyo <code>origin</code> es el Moodle.

<code>/local/tiendacursos/image.php?course=ID pack=ID&v=T</code>	Portada pública de un curso o pack a la venta (sirve aunque el sitio use <code>forcelogin</code>), caché pública inmutable.
--	--

Formato del feed (versión 1)

```
{
  "version": 1,
  "site": {"name": "...", "url": "https://aula.example.com/", "showcase": ".../local/tiendacursos/index.p",
    "currency": "CLP", "locale": "es-CL"},
  "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": nu
      "access_days": 90, "price": {"currency": "CLP", "normal": 189700, "offer": 149900, "offer_until": "2
      "current": 149900, "formatted": "$149.900", "formatted_normal": "$189.700", "savings": 39800, "feat
      "courses": [{ "id": 5, "name": "..."}], "image": ".../image.php?pack=1&v=...", "url": ".../checkout.p
    {"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.

12. Pruebas y calidad

- 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.