

TIC SERVICIOS

Central de Licencias

Instalación y operación

Release 1.0.0 · 20 de octubre de 2026

macOS primero. cPanel después.

Fuente autoritativa: archivos Markdown incluidos en docs/. Copie comandos desde Markdown para conservar exactamente las líneas y sustituya los placeholders.

Verificación ejecutada en Linux controlado. Mac, SMTP real, respaldo externo e instalación cPanel requieren validación de destino.

Instalación local en macOS - Central de Licencias

1. Conservar la estructura y comprobar binarios

Extraiga el ZIP fuente en ~/Developer/TIC_SERVICIOS_UNIVERSAL/. Si ambos ZIP contienen packages/license-protocol, es la misma biblioteca: compare SHA-256 antes de reemplazar. No combine versiones distintas.

```
cd ~/Developer/TIC_SERVICIOS_UNIVERSAL/apps/license-center

export PATH="/opt/homebrew/opt/php@8.4/bin:$PATH"

php -v

php --ini

php -m

composer --version

node --version

npm --version

/opt/homebrew/opt/mariadb@10.11/bin/mariadb --version

/opt/homebrew/opt/mariadb@10.11/bin/mariadb -h 127.0.0.1 -P 3307 -u <USUARIO_ADMIN_MARIADB> -p -e
"SELECT VERSION();"

Reemplace <USUARIO_ADMIN_MARIADB> por el administrador de MariaDB 10.11; -p pide la contraseña sin escribirla en el comando. Resultado: PHP 8.4 >=8.4.26, Node 24 LTS y MariaDB 10.11. Si el puerto no responde, use el procedimiento de arranque de su instalación aislada 10.11; no arranque ni modifique MySQL 9.6. No ejecute comandos con placeholders sin sustituirlos.
```

2. Dependencias y configuración

```
composer install --prefer-dist

composer check-platform-reqs

npm ci

npm run build

cp .env.example .env

chmod 600 .env
```

Edita .env: DB_USERNAME y DB_PASSWORD del usuario exclusivo de tic_licencias_dev, DB_PORT=3307, y APP_URL=http://127.0.0.1:8002. La BD _dev ya existe según el consolidado: no la elimine ni cree de nuevo. Si contiene datos previos, use ACTUALIZAR.md y deténgase antes de migrar sin respaldo. Usuario y contraseña de la otra aplicación son diferentes. Escape valores con espacios, # o \$ entre comillas en dotenv; no suba el archivo.

Para una base nueva y APP_KEY vacía, únicamente una vez:

```
php artisan key:generate

php artisan config:clear

php artisan tic:install --local

php artisan tic:owner:create
```

El último comando pide nombre, correo y contraseña dos veces. No hay usuario predeterminado. Si ya hay propietario, el comando rechaza una nueva inicialización. Abra sesión, confirme contraseña, active MFA, escanee el QR, confirme TOTP y guarde los códigos de recuperación fuera del equipo.

3. Ejecutar

```
php artisan serve --host=127.0.0.1 --port=8002
```

Visite <http://127.0.0.1:8002/login>. El otro proyecto usa otro puerto y otra cookie. Los assets ya están compilados; Vite dev no es necesario. Para recompilar cambios: `npm run build`. El servidor Artisan es solo para desarrollo. En otra Terminal:

```
cd ~/Developer/TIC_SERVICIOS_UNIVERSAL/apps/license-center
```

```
export PATH="/opt/homebrew/opt/php@8.4/bin:$PATH"
```

```
php artisan schedule:work
```

Este proceso ejecuta el scheduler local. cPanel usa cron, sin proceso permanente. `TIC_NETWORK_ENABLED=false` mantiene el desarrollo sin llamadas a dominios ajenos.

4. Preparar pruebas aisladas

Con el administrador de MariaDB cree únicamente las dos bases `_test` y dos usuarios propios desde `scripts/crear_bases_test.sql`. Defina contraseñas nuevas en su cliente SQL, nunca copie una contraseña del manual. No conceda al usuario de pruebas acceso a `_dev`.

```
cp .env.testing.example .env.testing
```

```
chmod 600 .env.testing
```

Edite `.env.testing`: nombre `tic_licencias_test`, usuario exclusivo `_test`, contraseña y puerto 3307. Genere una `APP_KEY` de pruebas distinta con `php artisan key:generate --env=testing`. Luego:

```
php artisan config:clear
```

```
php artisan test
```

```
php vendor/bin/phpstan analyse --no-progress
```

```
php vendor/bin/pint --test
```

Resultado esperado: pruebas verdes en MariaDB. La suite aborta antes de `RefreshDatabase` si `APP_ENV` o la BD no están autorizados. No use `--parallel`: la suite no crea bases con sufijos. No use los usuarios `_dev` para pruebas.

5. Verificaciones de uso

Login, MFA, dashboard, CRUD de usuarios, permiso denegado con consulta, auditoría y configuración deben funcionar. Cree organización, producto `tic.client-core` con módulo demo, plan y una instalación; configure llaves antes del canje. Verifique la cookie `tic_license_center`. Los tiempos de BD y auditoría se conservan en UTC. SMTP se configura y ensaya aparte: `MAIL_MAILER=log` solo es desarrollo y genera enlaces sensibles en logs locales que deben protegerse.

Despliegue cPanel - Central de Licencias

Ficha obligatoria

Copie `FICHA_CPANEL.example.json` a un archivo privado y rellene la evidencia de PHP web, CLI y cron. Ejecute diagnóstico autenticado para PHP web; Terminal y la propia entrada cron para CLI/cron. La ficha no prueba por sí sola la configuración del proveedor: conserve capturas o salidas y revise todos los alias.

No hay homologación de ese hosting en esta entrega.

Rutas conocidas del consolidado: CLI candidato `/opt/alt/php84/usr/bin/php`; deben comprobarse. El dominio `licences.migsoluciones.cl` es solo el piloto observado y se configura en `.env`.

Instalar paquete productivo

En Terminal de cPanel, sustituya `<CPANEL_USER>`, `<APP_DIR>` y `<ZIP_ABSOLUTO>` por valores reales. `pwd`, `whoami`, `command -v php` y el panel Dominios permiten obtenerlos. Ejemplo de `APP_DIR` privado: `/home/<CPANEL_USER>/apps/license-center`; el dominio debe resolver exclusivamente a ese `APP_DIR/public`.

```
whoami
```

```
pwd
```

```
/opt/alt/php84/usr/bin/php -v
```

```
/opt/alt/php84/usr/bin/php -m
```

```
unzip -l <ZIP_ABSOLUTO>
```

Extraiga el ZIP productivo fuera de cualquier raíz web. El ZIP contiene una carpeta `license-center/`; entre en ella antes de verificar. No suba el workspace completo a `public_html`. No cambie `index.php`. Configure Document Root físico `<APP_DIR>/public` para dominio y alias; si no es configurable, solicite un subdominio con raíz configurable al proveedor.

```
cd <APP_DIR>
```

```
/opt/alt/php84/usr/bin/php scripts/verificar-manifiesto.php
```

```
cp .env.example .env
```

```
chmod 600 .env
```

En el panel Bases de datos cree una BD y usuario exclusivos, con el prefijo de su cuenta; conceda permisos únicamente a esa base. En `.env` use `APP_ENV=production`, `APP_DEBUG=false`, `APP_URL=https://dominio-real`, `DB_PORT` del proveedor, `DB_DATABASE` y `DB_USERNAME` completos con prefijo, cookie propia, `SESSION_SECURE_COOKIE=true`, confianza staging o prod y `TIC_NETWORK_ENABLED=true`. No use llaves DEV en piloto/producción. SMTP y respaldos se configuran antes de homologar.

En instalación nueva, `APP_KEY` vacía:

```
/opt/alt/php84/usr/bin/php artisan key:generate
```

```
/opt/alt/php84/usr/bin/php artisan config:clear
```

```
/opt/alt/php84/usr/bin/php artisan tic:preflight --profile=<FICHA_JSON_ABSOLUTA>
```

```
/opt/alt/php84/usr/bin/php artisan tic:install --profile=<FICHA_JSON_ABSOLUTA>
```

```
/opt/alt/php84/usr/bin/php artisan tic:owner:create
```

```
/opt/alt/php84/usr/bin/php artisan optimize
```

La ficha exige versiones PHP iguales y extensiones del perfil, cron compatible, inodos y Document Root correctos. El preflight también comprueba MariaDB, espacio, permisos y respaldo externo configurado. No instala extensiones ni corrige permisos del sistema. Corrija cada fallo antes de migrar. Solo storage y bootstrap/cache deben ser escribibles por PHP; no use 777. La central prod necesita cuenta separada verificada.

Cron, SMTP y validación H5

Configure una entrada por aplicación, con rutas absolutas y registro privado. Sustituya APP_DIR antes de pegar:

```
* * * * * cd <APP_DIR> && /opt/alt/php84/usr/bin/php artisan schedule:run >>
```

```
<APP_DIR>/storage/logs/scheduler.log 2>&1
```

Si el proveedor permite mínimo cinco minutos: */5 * * * * y TIC_CRON_MINUTES=5 en .env, luego config:clear y optimize. El worker es acotado, database, timeout 30 s, retry_after 90 s. PCNTL es obligatorio en CLI para aplicar timeout. No lance workers permanentes ni Redis.

Use MAIL_MAILER=smtplib, host/usuario/contraseña propios, puerto/TLS del proveedor y remitente autorizado. Ensaye recuperación de contraseña y recepción efectiva, SPF/DKIM/DMARC, cola fallida y reintento. El modo log no demuestra entrega.

Un monitor independiente consulta /health/monitor con header secreto TIC-Monitor-Token; valor aleatorio >=32 caracteres en .env. El cuerpo solo informa antigüedad del scheduler y cantidad de fallos. Configure alerta externa; detener el cron vigilado debe producir alerta. Nunca use el mismo cron para vigilarse a sí mismo.

Pruebe: /, login, MFA, panel, API, CSS/JS, cookies Secure/HttpOnly y callbacks HTTPS. Compruebe 404/403 para /.env, /vendor/autoload.php, /storage/logs/laravel.log, /backups/ y /storage/app/private/tic/installation.key en todos los dominios y alias. Revise cuota, inodos, límites LVE, TLS y cuenta de aislamiento. Ejecute respaldo externo y restauración en staging con tiempo medido. Solo después complete aceptación H5. Conserve el release anterior y su SHA-256.

Licencias - Central de Licencias

La central no se autolicencia. Gestione organizaciones, productos/módulos, planes, instalaciones y estados desde el panel con MFA y permisos. Producto del cliente: tic.client-core, módulo demo, entitlements core y demo ordenados. Los cupos nulos significan sin límite. Planes con vencimiento requieren duración; PERPETUAL usa expires_at null.

Raíz y llaves por entorno

La clave raíz privada nunca se aloja en el hosting. Use scripts/offline-root.php del workspace en un equipo custodio con PHP, sodium y la biblioteca instalada:

```
php scripts/offline-root.php create staging /ruta/privada/nueva-staging
```

Distribuya solo anchor.json por canal verificado. Complete root_id/public_key del entorno en config/tic_trust.php de ambas aplicaciones. Respalde la raíz fuera del hosting bajo custodia separada y no la incluya en releases. En la central, configure TIC_SIGNING_KEY_DIRECTORY privado; prepare las llaves license/event y el payload público:

```
php artisan tic:keys prepare staging --file=/ruta/privada/payload-1.json
```

Lleve ese payload al custodio offline, compruebe entorno, public keys, key_id, versión, usos y ventanas, y firme:

```
php scripts/offline-root.php sign /ruta/privada/nueva-staging /ruta/payload-1.json /ruta/keyset-1.json
```

De vuelta en central: php artisan tic:keys import staging --file=/ruta/keyset-1.json. config:clear/optimize tras editar anclas. El endpoint público publica solo keyset firmado. Para DEV existe tic:keys dev-init dev --root-dir=/ruta/externa; está bloqueado fuera de local/testing.

Rotación: prepare --rotate=license o --rotate=event; la llave anterior queda retired con ventana de emisión de 14 días y se agrega una activa. Los documentos históricos se verifican según issued_at, no por la fecha del servidor. Conserve todas las claves públicas históricas en keysets. En compromiso, prepare una nueva versión, cambie status del key_id afectado a compromised en el payload revisado por el custodio, firme offline e importe. Nunca sustituya la raíz por una recibida de red sin procedimiento de confianza. Mantenga llaves de licencia y evento separadas y backups cifrados.

Canje y estados

El código TIC1 aleatorio tiene 130 bits, hash-only, 72 h, cinco intentos y un uso. Solo se muestra al emitir, no queda en logs. ACTIVATE parte de PENDING; REKEY conserva licencia y cambia fingerprint después de proof. Una respuesta perdida se recupera usando mismo fingerprint/request_id/cuerpo. La transacción guarda revisión, documento, Outbox y auditoría; no hay confirmación sin ACK firmado.

ACTIVE/SUSPENDED/REVOKED son cambios POST con motivo, permiso, MFA y reautenticación. REVOKED es terminal; una contratación posterior crea otra licencia. El panel separa estado deseado, revisión emitida, confirmada y último error. DELIVERED no implica CONFIRMED. El despachador pinnea DNS público y rechaza redirects, IP internas y TLS inválido. El cliente también recupera pendientes por sync autenticado.

Restauración y cierre

CENTRAL_BEHIND_CLIENT exige conciliación del historial externo antes de emitir; no se reduce revisión del cliente. Conserve expedientes, anclas y claves públicas, emitidos, Outbox, idempotencia y claves online recuperables. Para cierre definitivo: congelar altas/cambios comerciales, inventariar contratos, entregar último documento válido, keysets y evidencia al titular, acordar continuidad/escrow y responsables antes de apagar servicios. Una PERPETUAL no se revoca por el cierre; no se liberan llaves privadas universales. El plan contractual y soporte definitivo se aprueba en H6.

Comandos de Central de Licencias

Ejecute desde apps/license-center con PHP 8.4; en cPanel anteponga el binario absoluto verificado. Los placeholders deben sustituirse antes de copiar.

Comando	Uso
php artisan tic:preflight --local	Comprueba CLI y BD de local/testing; no homologa web
php artisan tic:preflight --profile=/ruta/ficha.json	Preflight bloqueante del hosting
php artisan tic:install --local	Migraciones iniciales y permisos; requiere APP_KEY y BD configuradas
php artisan tic:owner:create	Primer propietario por diálogo privado
php artisan tic:permissions:sync	Registra permisos del código y roles base
php artisan tic:backup:key /ruta/externa/nueva.key	Genera 32 bytes sin mostrarlos; directorio existente privado
php artisan tic:backup --external	Cifra BD/archivos/config y copia a destino externo
php artisan tic:restore /ruta/copia.ticbak --verify-only	Autentica sin tocar BD
php artisan tic:restore /ruta/copia.ticbak	Requiere mantenimiento y confirmación del nombre de BD
php artisan tic:upgrade --profile=/ruta/ficha.json	Preflight, respaldo externo, mantenimiento y migración
php artisan tic:mfa:recover correo --approver=otro-correo	Segundo propietario, contraseña, TOTP y expediente
php artisan schedule:run	Una iteración del scheduler
php artisan queue:failed	Bandeja durable de correo fallido
php artisan queue:retry <ID>	Reintentar una tarea revisada
php artisan tic:maintenance	Limpiar sesiones/idempotencia/auditoría según retención
php artisan migrate:status	Inventario de migraciones
php artisan migrate --pretend	SQL pendiente para revisión
php artisan test	Suite sobre tic_licencias_test y APP_ENV=testing
php vendor/bin/phpstan analyse	Análisis estático con Larastan
npm ci && npm run build	Assets reproducibles solo en máquina de compilación

Licencias central: tic:keys dev-init dev --root-dir=/custodia/dev, tic:keys prepare <entorno> --file=/payload.json, tic:keys import <entorno> --file=/keyset.json, tic:keys export <entorno> --file=/keyset.json, tic:outbox:dispatch --limit=20, tic:license:dev <identidad.json> <directorio-nuevo> (solo local/testing).

No se documentan comandos inexistentes para borrar usuarios, licencias o módulos. No use migrate:fresh/refresh/reset/rollback o db:wipe fuera de las bases de prueba autorizadas. tic:restore --test es exclusivamente QA y tiene guarda propia.

Respaldo y recuperación - Central de Licencias

Configurar custodia

Cree un directorio privado fuera del proyecto (0700) y ejecute `php artisan tic:backup:key /ruta/privada/backup.key`. Configure `TIC_BACKUP_KEY_FILE`, `TIC_BACKUP_DIR` y binarios MariaDB 10.11. Duplique la llave en custodia externa offline: perderla hace irrecoverable el respaldo. La raíz de licencias se custodia separadamente offline, nunca entra al hosting ni al ZIP.

Formato TICBACKUP2: tar.gz cifrado con AES-256-GCM por bloques de hasta 1 MiB; nonce por bloque, índice autenticado y marcador final autenticado. Detecta modificación, reorden y truncamiento antes de extraer/restaurar. Incluye SQL consistente de `mariadb-dump` (InnoDB, single-transaction), `.env` con `APP_KEY`, `config`, `storage/app/private/tic` y adjuntos `demo`, y llaves online externas configuradas. No incluye `vendor` ni `assets`: conserve el release inmutable correspondiente. No ejecute respaldo durante DDL; mantenimiento y una ventana sin escrituras aseguran consistencia entre SQL y archivos.

El temporal privado sin cifrar se elimina en `finally`. Reserve espacio para SQL + tar + cifrado y dimensione cuotas. El archivo SQL de restauración se envía al cliente MariaDB; pruebe el tamaño objetivo y límites de memoria/timeout en cPanel antes de aceptar cargas reales.

Copia externa y retención

`TIC_BACKUP_REMOTE=usuario@servidor-externo:/ruta/privada/` y `TIC_BACKUP_SSH_KEY=/ruta/ssh-dedicada`. SCP usa `BatchMode` y `StrictHostKeyChecking=yes`; dé de alta el fingerprint del servidor verificado por canal independiente. Nunca desactive verificación de host. Pruebe `tic:backup --external`. Un éxito SCP prueba transferencia, no restauración ni independencia de cuenta: verifique `checksum`/lectura en destino y ejecute ensayo desde esa copia.

Política propuesta: copia diaria y antes de cada actualización; 7 diarias, 4 semanales y 12 mensuales. La retención se ejecuta en el repositorio externo por su administrador, nunca borrando automáticamente la única copia local. Responsable y destino concreto se completan al desplegar. La transferencia a una cuenta externa real no se ha ejecutado en este entorno; es una condición operativa pendiente, no una homologación concedida por este manual.

Restaurar

Prepare una app aislada del mismo release con la llave AES fuera del proyecto y el `.env` de destino para conectarse. Verifique primero:

```
php artisan tic:restore /ruta/copia.ticbak --verify-only
```

```
php artisan down --retry=60
```

```
php artisan tic:restore /ruta/copia.ticbak
```

El comando solicita el nombre exacto de la BD, recupera SQL, `.env/APP_KEY`, `config` y archivos, revoca sesiones y tokens, y limpia cachés. No cambie `APP_KEY`. En un destino distinto revise `DB_*` del `.env` recuperado ANTES del siguiente comando para evitar conectar a la instalación original; conserve el `.env` inicial en un respaldo privado. Revise permisos 0600 de secretos. El comando no cambia usuarios/grants del servidor. Durante QA, `--test` solo permite `tic_cliente_test/tic_licencias_test` y `testing`.

Después de restaurar la central, compare revisiones confirmadas por clientes con `issued_documents` y archivos externos. `CENTRAL_BEHIND_CLIENT` alerta la discrepancia: no emita una revisión vieja como corrección ni reduzca derechos. Recupere documentos/llaves/outbox/idempotencia de la copia más nueva; un custodio debe reconstruir el historial faltante antes de volver a emitir. No invente una confirmación de ACK.

Compare cuentas, registros, licencia, llaves públicas y checksums de adjuntos; ejecute `smoke tests` y mida RTO real. Recuperar BD no recupera DNS ni TLS. Una caída de central no implica caída del cliente y

un certificado vencido no vence una licencia. RPO/RTO contractuales requieren aprobación H6; no se prometen tiempos ni medidas.

Actualizar Central de Licencias

1. Lea CHANGELOG, compatibilidad y SQL de migraciones del release. Verifique ZIP y manifiesto. Conserve el ZIP anterior fuera de public/ y espacio para ambos.

2. Con el código anterior todavía activo: ejecute `preflight` y `php artisan tic:backup --external`. Descargue una copia cifrada a una cuenta diferente y verifique restauración en una instalación de ensayo. Registre tiempos.

3. `php artisan down --retry=60`. Conserve `.env`, storage completo, `config/tic_trust.php` y llaves externas. No use `key:generate`.

4. Extraiga el release nuevo en carpeta privada de staging. Verifique su manifiesto antes de personalizar. Sin symlinks, sustituya código, vendor y `public/build` con mantenimiento activo; preserve los elementos del punto 3. No superponga `node_modules` ni cachés de otra máquina. No cambie Document Root.

5. Ejecute `php artisan optimize:clear`, `php artisan tic:upgrade --profile=/ruta/ficha.json` y `php artisan migrate:status`. El comando respalda de nuevo y mantiene el modo mantenimiento para revisión.

6. Verifique login, permisos, assets, endpoints, jobs y licencias. `php artisan up` solo cuando el smoke test pase. Active cron y confirme latidos, correo y sincronización.

Rollback de código: vuelva al ZIP anterior solo si esquema y migraciones son compatibles; restaure `.env/storage` sin regenerar `APP_KEY`. Compensación de datos: migración correctiva revisada, no rollback destructivo automático. Restauración completa: procedimiento independiente de `RESPALDO_RECUPERACION.md`, con conciliación de operaciones externas. No confirme un correo o evento por haber recuperado una fila antigua. Conserve al menos un release anterior.

La alternativa `releases/current` requiere comprobar symlinks y protección Apache con el proveedor. No es el método predeterminado.

Operación diaria - Central de Licencias

Revise dashboard, auditoría, diagnóstico, latido del scheduler y failed_jobs. El diagnóstico exige system.diagnose y MFA; no muestra contraseñas ni .env. El monitor independiente requiere token. El heartbeat de BD solo da evidencia al monitor, no puede alertar si nadie lo consulta.

Colas: database, worker acotado, tres intentos y backoff del correo 60/300/900 s. Investigue queue:failed antes de queue:retry. Un reset de contraseña queda en cola, no se declara recibido por encolarlo. Mantenga SMTP real y prueba de recepción. Los logs rotan a 14 días, auditoría propuesta 365; confirme retención con responsable. El mantenimiento elimina idempotencia a 30 días y sesiones expiradas.

MFA: obligatoria en propietarios/administradores y todos los operadores de la central. Guarde códigos de recuperación fuera de la app. La recuperación excepcional exige otro propietario con contraseña y TOTP y referencia de expediente; propietario único usa sus códigos guardados. No hay bypass universal. Desactivar cuenta o cambiar contraseña revoca sesiones/tokens.

Revise PENDING/DISPATCHING/DELIVERED/CONFIRMED/REJECTED/DEAD de Outbox. DELIVERED no significa aplicado: solo ACK firmado aumenta revisión confirmada. Tras ocho fallos pasa a DEAD; sigue recuperable por polling. Antes de reprogramar un DEAD investigue origen/TLS/llaves. Nunca cambie status por SQL sin expediente y revisión de consistencia.

Pruebe recuperación trimestral y después de cambios mayores. Mantenga calendario de PHP/Laravel, cuota/inodos, TLS, DNS y custodia de secretos. Producción exige responsable nominal, soporte y alertas acordados.