> **LO ESENCIAL · Módulo 11 · Multiusuario y operación**
>
> Tres empleados, tres permisos, un sistema. Trazas atribuidas como producto y Git que separa plantilla de cliente.

11.1 · Identidad federada, grupos, cuotas y aislamiento

> Objetivos
>
> - Dar de alta tres usuarios con permisos distintos
> - Aislar datos entre usuarios y poner cuotas

Tres personas, un sistema

El gerente ve todo. La contadora ve contabilidad y contratos, y **aprueba** registros. El vendedor ve inventario y precios, y nada más. Esa frase, convertida en configuración, es este módulo.

Capa	Dónde se define	Qué garantiza
Identidad	proveedor SSO (3.2): usuario, MFA, grupos	quién es
Autorización por grupos	el proxy pasa los grupos; cada servicio los lee de la cabecera	qué puede ver
Permisos en el índice	`permisos && grupos` antes de recuperar (6.3)	qué documentos entran al contexto
Aprobaciones	la cola (10.4) filtra por grupo	qué puede autorizar
Cuotas	por usuario: peticiones/minuto, tokens/día	que uno no sature a los demás
Aislamiento de datos	conversaciones, borradores y trazas con `usuario_id`; nadie lee las de otro	privacidad entre empleados

Cuotas

```
# cuota.py — límite por usuario, en el nodo de servicios, antes de llamar al modelo
LIMITES = {"gerencia": {"rpm": 30, "tokens_dia": 400_000}, "contabilidad": {"rpm": 20, "
    tokens_dia": 250_000},
          "ventas": {"rpm": 10, "tokens_dia": 100_000}}
def permitido(usuario, grupos, tokens_estimados, db):
    lim = max((LIMITES[g] for g in grupos if g in LIMITES), key=lambda l: l["rpm"], default=
    LIMITES["ventas"])
    if db.peticiones_ultimo_minuto(usuario) >= lim["rpm"]: return False, "límite por minuto"
```

```
    if db.tokens_hoy(usuario) + tokens_estimados > lim["tokens_dia"]: return False, "cuota diaria
    "
    return True, ""
```

Las cifras salen de la prueba de carga (4.4): si el equipo aguanta 5 usuarios a 3 s, la suma de `rpm` de todos no puede superar lo que la GPU procesa por minuto.

> PROCEDIMIENTO · **Laboratorio 11.1 · los tres usuarios operando**
>
> Con el SSO, el proxy, el índice con permisos, la cola y las cuotas: cada usuario entra, hace tres consultas propias de su rol, una que **no** le corresponde, y la contadora aprueba una factura. Verifique en las trazas (11.2) que todo quedó atribuido y que la consulta indebida no trajo fichas.
> **Entregable:** `labs/11-01/prueba-usuarios.md` con las 12 consultas y su resultado. Commit: «Capa 11.1: multiusuario».

Qué se degrada en cada perfil

A — Un usuario a la vez: las cuotas sirven para **turnar**, no para repartir

B — Cuotas ajustadas a 3–5 usuarios

C — Holgura

Cuándo NO usar esto **Identidad federada, grupos, cuotas y aislamiento**

- No implemente permisos «en el prompt» («no muestres nómina al vendedor»): el modelo no es un control de acceso.
- No comparta un usuario entre empleados «para simplificar»: se pierde la atribución y la ley (2.3).

PRÁCTICA · **Práctica**

1. Llega un cuarto empleado: auxiliar de bodega. Defina grupo, permisos, cuota.
2. ¿Qué pasa cuando la contadora se va de la empresa? Liste los pasos (con 3.3).

Referencias

1. Módulos 3.2, 6.3 y 10.4 de este curso.
2. OWASP LLM Top 10 — LLM01 (inyección de prompt) explica por qué los permisos no van

en el prompt.

11.2 · Trazas atribuidas como producto vendible

> Objetivos
>
> - Registrar quién, qué, con qué fuente y a qué costo
> - Generar el reporte mensual de uso

La traza es el producto

Un cliente no puede ver «la IA». Puede ver un registro que diga: *el 14 de agosto a las 10:32, la contadora consultó la garantía del contrato 2026-014; el sistema recuperó la cláusula 7 (página 3); respondió citándola; costó 3 400 tokens y 4,1 segundos.* Eso se puede auditar, facturar y mejorar. Sin eso, el sistema es una caja negra que un día alguien apaga.

Qué lleva cada traza

```
{"id":"…tr_01J","cuando":"2026-08-14T10:32:07-05:00","usuario":"contadora","grupos":["
    contabilidad"],
 "tarea":"consulta","estado_final":"respondida",
 "pasos":[{"n":1,"rol":"consultor","modelo":"<modelo>@<version>","tokens_in":3100,"tokens_out
    ":300,"ms":4100,
          "fuentes":["contrato-2026-014#7"],"herramientas":["buscar_documentos"]}],
 "costo":{"tokens":3400,"segundos":4.1},
 "aprobaciones":[], "errores":[]}
```

Reglas: la traza guarda **identificadores** de fuentes, no su contenido (privacidad); guarda la **pregunta** solo si la política de datos lo permite (2.3), y con retención de 12 meses; **nunca** guarda datos sensibles en claro.

Lo que sale de las trazas

Para quién	Reporte	Frecuencia
Gerente	uso por persona, tareas completadas, ahorro estimado de horas	mensual
Contadora	qué se registró automático y qué aprobó ella	semanal
Implementador	tasas de fallo por estado, latencias, cuota consumida	diaria (canario)
Auditoría	quién vio qué documento	a demanda

El reporte mensual es una **línea de facturación**: «mantenimiento con reporte de uso».

> PROCEDIMIENTO · **Laboratorio 11.2**
>
> Instrumente el ejecutor (10.3) y el consultor para escribir la traza de arriba en la base. Genere `reporte_mensual.md` con un script: usuarios activos, tareas por estado, tokens por rol, las 5 consultas más frecuentes (sin texto sensible), aprobaciones.
> **Entregable:** `traza.py`, `reporte.py`, y un reporte con datos del laboratorio. Commit: «Capa 11.2: trazas».
> Nada. Las trazas cuestan una escritura en base por paso.

Cuándo NO usar esto **Trazas atribuidas como producto vendible**

- No guarde el contenido de los documentos en la traza «por si acaso».
- No mida «satisfacción» con un pulgar: mida tareas completadas sin corrección humana.

PRÁCTICA · **Práctica**

1. Diseñe la consulta SQL para «quién vio el contrato 2026-014 en julio».
2. ¿Qué campos de la traza se anonimizan a los 12 meses y cuáles se borran?

Referencias

1. OpenTelemetry — trazas distribuidas (el formato se puede adaptar): https://opentelemetry.io/docs/
2. Langfuse — trazado de aplicaciones con modelos, autoalojable: https://langfuse.com/docs `[VOLÁTIL — verificado: 2026-09]`

11.3 · Git para el servicio: plantilla, derivados y secretos fuera

> Objetivos
>
> - Separar repo plantilla de repos por cliente
> - Sacar los secretos del repo

Un repositorio plantilla, un repositorio por cliente

Todo lo que se construyó es texto (0.2). Se organiza en dos niveles:

```
ia-local-pymes-plantilla/        ← el producto: se mejora aquí├──
 docker-compose.yml├──
 config/            .env.example  maquina.yaml  esquemas/  contratos/├──
 skills/            extraer-factura/  redactar-acta/├──
 labs/              (los de este curso)├──
 canario/├──
 docs/              entrega/  runbooks/└──
 VERSION

el-tornillo/                     ← derivado: plantilla + lo del cliente├──
 (todo lo anterior, como submódulo o rama)├──
 config/.env        ← NO SE VERSIONA├──
 config/politica-datos.md├──
 set/               ← 40 documentos anotados del cliente (privado)└──
 CLIENTE.md
```

Cambios en la plantilla se **propagan** a los derivados (rebase o actualización del submódulo) con una prueba: el canario del cliente sigue en verde.

Qué se versiona y qué no

Se versiona	No se versiona
código, esquemas, contratos, máquinas de estado	`.env` y cualquier secreto
`docker-compose`, configuración de proxy y SSO **sin** claves	pesos de modelos
documentación, runbooks, políticas	índice vectorial, bases de datos
set de prueba anotado (en el repo **privado** del cliente)	trazas
exportaciones de flujos (9.1)	documentos del cliente

Secretos fuera del repo

```
# .env.example — versionado, sin valores
POSTGRES_PASSWORD=
SSO_CLIENT_SECRET=
RESTIC_PASSWORD_FILE=/run/secrets/restic
# .gitignore
.env
*.key
config/secretos/
```

Los valores reales viven en el servidor (`.env` con permisos 600) o en un gestor de secretos, y **una copia impresa en sobre cerrado** en la caja fuerte del cliente (14.2). Un secreto que solo existe en el portátil del implementador es un secreto perdido.

Antes de cada commit, un gancho busca secretos:

```
git config core.hooksPath .githooks
# .githooks/pre-commit: rechaza si aparece algo con pinta de clave
git diff --cached | grep -nE '(PASSWORD|SECRET|TOKEN|KEY)=[^ ]{8,}' && { echo "SECRETO EN EL
    COMMIT"; exit 1; } || exit 0
```

> PROCEDIMIENTO · **Laboratorio 11.3**
>
> Cree `ia-local-pymes-plantilla` y derive `el-tornillo`. Haga un cambio en la plantilla (un canario nuevo), propáguelo, y verifique que el `.env` del cliente **no** se tocó ni se subió. Intente commitear un `.env` con clave: el gancho debe rechazarlo.
> **Entregable:** los dos repos, el gancho, y `labs/11-03/propagacion.md`. Commit: «Capa 11: plantilla y derivado».
> Nada.

Cuándo NO usar esto **Git para el servicio: plantilla, derivados y secretos fuera**

- No haga un repo por cliente copiando y pegando: en seis meses son seis productos distintos.
- No versione el set de prueba en el repo de la plantilla: son datos del cliente.

PRÁCTICA · **Práctica**

1. Un cliente pide una regla especial en el extractor. ¿Va en la plantilla o en el derivado?
2. Rotó la clave de la base. Liste dónde hay que cambiarla.

Referencias

1. Git — submódulos y `core.hooksPath`: https://git-scm.com/docs
2. Módulo 14.2 — la entrega incluye el sobre con los secretos.