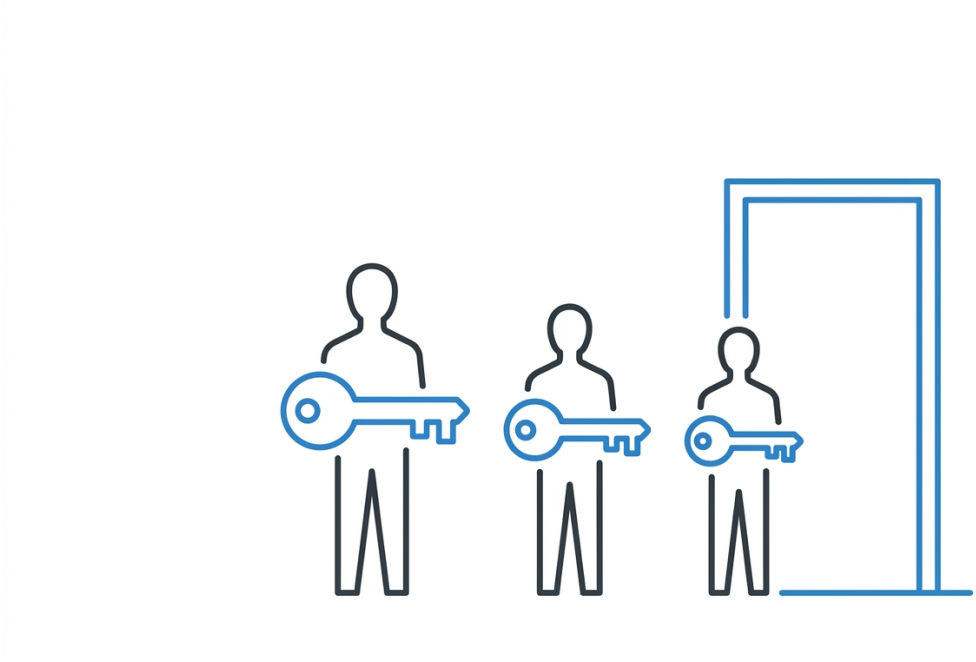
> **LO ESENCIAL · Módulo 11 · Multiusuario y operación**
>
> Tres empleados, tres permisos, un sistema. Trazas atribuidas como producto y Git que separa plantilla de cliente.

11.1 · La pantalla que usa el vendedor (y el harness detrás)



Multiusuario y operación

El vendedor no va a abrir una terminal

Todo lo que hemos montado hasta aquí —el orquestador, el harness, los agentes, las trazas— es invisible. Y tiene que seguir siéndolo. Porque quien lo va a usar de verdad es Wilmer, el vendedor, que tiene 22 años, atiende el mostrador con las manos ocupadas y no va a abrir una ventana negra a escribir comandos. Si la herramienta le estorba, no la usa. Y si no la usa, el proyecto fracasó, por muy bien que esté hecho por dentro.

La pregunta de esta lección es esa: **¿qué ve Wilmer?**

Y la respuesta empieza por una idea que no es de programación: un cajero automático. Por dentro tiene una red bancaria, cifrado, autorizaciones, auditoría. Por fuera tiene cuatro botones grandes. Nadie le enseña a nadie a usar un cajero. Eso es lo que hay que construir.

Tres pantallas, no un chat

El primer impulso de todo el mundo es poner un chat: una cajita donde el usuario escribe lo que quiera. Es lo peor que se puede hacer con alguien no técnico, por tres razones.

La primera: **la página en blanco asusta**. Wilmer no sabe qué se le puede pedir. Va a escribir «hola», va a recibir algo genérico, y no vuelve.

La segunda: **el chat invita a pedir cualquier cosa**, incluidas las que el sistema no hace bien. Wilmer pregunta «¿cuánto vendimos el año pasado?» y el sistema, que solo sabe de documentos, se inventa una cifra.

La tercera: en un mostrador, escribir es lento.

Lo que sí funciona son **botones que corresponden a las tareas reales**. La regla: una pantalla por cada cosa que la persona hace de verdad, y nada más.

Pantalla uno — Buscar. Una caja de búsqueda grande y, debajo, tres o cuatro búsquedas frecuentes ya escritas, para tocar con el dedo: «Precio de un producto», «¿Llegó la remisión de un proveedor?», «Garantía de un contrato». Wilmer toca una, le cambia el nombre, y listo. La respuesta viene **con la fuente**: «Contrato 2026-014, cláusula 7, página 3» —y esa línea se puede tocar para ver el documento original. Sin fuente, no se muestra la respuesta.

Pantalla dos — Subir. Un cuadro grande que dice «Suelte aquí la foto de la factura». Wilmer la toma con el celular y la suelta. El sistema responde una sola cosa: «Recibida. Le avisamos cuando esté lista». Nada más. Todo lo que pasa después —el ruteo, el OCR, la extracción, la validación— es invisible.

Pantalla tres — Pendientes. Solo la ve doña Marta, porque solo ella aprueba. Una lista de facturas con la confianza baja. Cada una muestra, lado a lado, **el documento a la izquierda y las casillas a la derecha**. Ella mira, corrige un dato si hace falta, y toca «Aprobar». Eso es todo su trabajo en el sistema: mirar y aprobar. Antes transcribía.

Las cinco reglas de la pantalla

Cada respuesta trae su fuente, y se puede abrir. Es lo que separa una herramienta de un oráculo. Si el sistema no puede citar de dónde lo sacó, dice «no lo encontré en los documentos», y ya. Nunca inventa.

El sistema dice que no sabe. Con todas las letras: «No encontré eso en los documentos que tengo». Un sistema que nunca dice que no sabe, miente.

Cada quien ve lo suyo. Wilmer entra y ve inventario y precios. Marta entra y ve además contabilidad y contratos, y la lista de pendientes. El gerente lo ve todo. No es que las opciones estén escondidas: es que **no existen** para quien no tiene permiso. Y eso no se decide en la pantalla —se decide antes de buscar, en el archivador (módulo 6).

Lo que no se puede deshacer, pregunta. Registrar un pago, mandarle un correo al proveedor: eso muestra un cartel con lo que va a pasar, y espera a que una persona con permiso diga que sí.

Se entra sin contraseña nueva. Wilmer ya tiene una clave de la empresa. Toca el ícono, pone la huella, entra. Una contraseña más es una contraseña anotada en un papel debajo del teclado.

Lo que hay detrás de cada botón

Wilmer toca «Buscar precio». Lo que pasa, y que él nunca ve:

La pantalla le manda la pregunta al harness, junto con **quién es** —eso lo pone la portería, no la pantalla; si lo pusiera la pantalla, alguien podría mentir—. El reglamento de la oficina —el harness— mira qué grupos tiene Wilmer y con eso va al archivador: **solo busca en lo que él puede ver**. Trae veinte carpetas, las afina a cinco, se las pone al Modelo sobre la mesa junto con la pregunta, y le exige que responda con casillas: respuesta, fuente, confianza. Si la confianza es baja, la pantalla no le muestra a Wilmer una respuesta insegura: le muestra el documento y le dice «revíselo usted». Y todo eso —quién preguntó qué, qué documentos se usaron, cuánto tardó, cuánto costó— queda en el cuaderno.

Wilmer vio: un botón, tres segundos, una respuesta con un enlace. Ese es el trabajo bien hecho.

Paso a paso: diseñar la pantalla

1. **Siéntese al lado de la persona un día entero.** Anote las cinco cosas que hace de verdad, con sus palabras. No con las suyas.
2. **Descarte tres.** Se queda con las dos más frecuentes. Esas son sus dos primeras pantallas.
3. **Dibújelas en papel** y muéstreselas. Si tiene que explicar algo, está mal: redibuje.
4. **Escriba las búsquedas sugeridas** con las palabras de ella, no las suyas. «¿Ya llegó lo de Tornillería?» le gana a «Consultar estado de remisión».
5. **Haga que la fuente se pueda tocar** desde el primer día. Es la función que genera confianza.
6. **Póngale el cartel de confirmación** a todo lo irreversible antes de dejar entrar a nadie.
7. **Pruebe con la persona real, sin ayudarla.** Cronometre cuántos segundos tarda en hacer su tarea. Anote cada vez que pregunta algo: cada pregunta es un botón mal puesto.
8. **Mire el cuaderno a la semana.** Lo que nadie usó, se quita. Una pantalla con seis botones de los que se usan dos es una pantalla con cuatro estorbos.

La trastienda

- La identidad **no la manda la pantalla**: el proxy inverso valida contra el SSO y pasa cabeceras (`usuario`, `grupos`) que el harness lee. Un servicio alcanzable sin pasar por el

proxy permitiría falsificarlas: por eso solo escucha al proxy (módulo 3.2).

- Filtro de permisos **en la consulta al índice** (`permisos && grupos`), nunca como post-proceso ni como instrucción en el prompt (módulo 6.3, OWASP LLM01).
- La respuesta llega con esquema: `{respuesta, fuentes[], confianza}`. Umbral 0,8: por debajo, la interfaz muestra el documento en vez de la respuesta.
- Acciones irreversibles → cola de aprobación (módulo 10.4), con el original al lado.
- Cada petición escribe una traza atribuida (módulo 11.2): usuario, fuentes por id, tokens, ms.
- Stack sugerido: la interfaz puede ser una página sencilla servida por el mismo nodo de servicios; si ya existe Open WebUI, sirve para el gerente, **no** para el mostrador —ahí van las tres pantallas propias.

Ver módulos 3.2, 6.3, 10.4, 11.1 y 11.2.

11.2 · 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.

11.3 · 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]`

11.4 · 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.