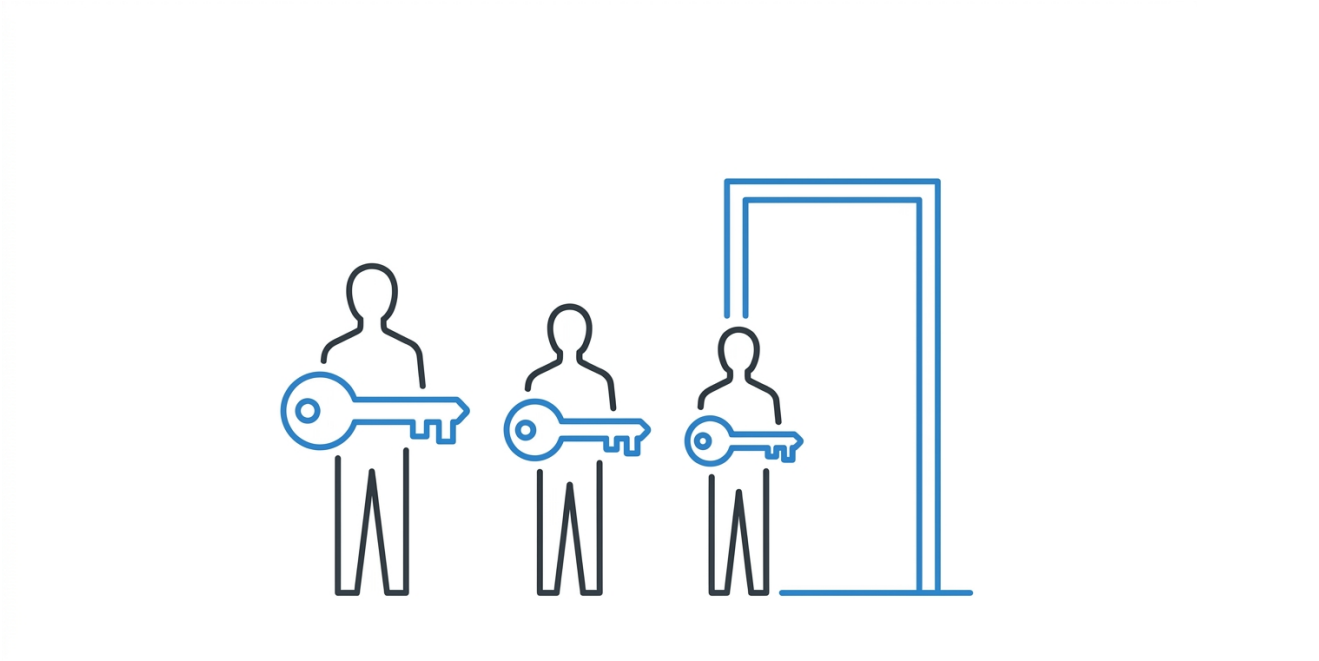
LO ESENCIAL · Módulo 12 · Evaluación y observabilidad

Un set de evals desde cero, métricas por tarea, trazas multipaso y canarios nocturnos que avisan cuando algo se degrada.

12.1 · Un set de evals desde cero



Evaluación y observabilidad

Objetivos

- Escribir 60 casos con respuesta esperada y criterio
- Automatizar la corrida

Un examen que se repite

Un **eval** es un caso con entrada, respuesta esperada y criterio de corrección. Un **set de evals** son sesenta. Se corre antes de cada cambio (modelo, prompt, esquema, embedding) y después. Si la nota baja, el cambio no entra. Es la única defensa contra «mejoramos el prompt y rompimos las facturas».

De dónde salen los 60 casos

Fuente	Cuántos	Tipo
Set de facturas anotadas (7.4)	20	extracción
Consultas del cliente con ficha esperada (6.2)	20	recuperac correcta
Transcripciones con acta corregida por el gerente (8.1)	10	redacción nes correc
Casos de permisos (6.3)	5	seguridad
Casos «trampa» (pregunta sin respuesta en los documentos)	5	honestida

```
{"id":"ev-031","tipo":"consulta","entrada":{"usuario":"contadora","pregunta":"¿Cuál es el plazo
    de garantía del contrato 2026-014?"},
 "esperado":{"fuente":"contrato-2026-014#7","contiene":["12 meses"],"no_contiene":["24 meses"]},
 "criterio":"fuente correcta Y contiene Y no contiene"}
```

PROCEDIMIENTO · **Laboratorio 12.1 · correr el set**

```
python3 evals.py --set evals/ --salida corrida-$(date +%F).json
python3 - <<'PY'
import json; r=json.load(open("corrida-2026-09-05.json"))
for t in ["extraccion","consulta","acta","permisos","trampa"]:
    c=[x for x in r if x["tipo"]==t]; print(f"{t:11s} {sum(x['ok'] for x in c)}/{len(c)}")
PY
```

Lo que debe ver: cinco filas con fracciones. La de `permisos` tiene que ser **5/5 siempre**; si baja, nada más importa. La de `trampa` dice si el sistema inventa.

Entregable: `evals/` con 60 casos, `evals.py`, y la primera corrida. Commit: «Capa 12.1: set de evals».

Nada en el método. En A, correr 60 casos tarda ~1 h: se hace de noche (12.4).

Cuándo NO usar esto **Un set de evals desde cero**

- No use un modelo grande «como juez» sin casos con respuesta esperada: el juez también se equivoca, y se equivoca parecido.
- No escriba casos que el sistema ya pasa: los evals valiosos son los que fallan.

PRÁCTICA · **Práctica**

1. Escriba tres casos «trampa» para la ferretería.
2. ¿Por qué el set de facturas para medir (7.4) y el set de evals deben ser distintos del set

para ajustar el prompt?

Referencias

1. Módulos 6.2, 6.3, 7.4 y 8.1 — de donde salen los casos.

2. OpenAI — `evals` (marco de referencia, adaptable a local): https://github.com/openai/evals `[VOLÁTIL — verificado: 2026-09]`

12.2 · Métricas por tipo de tarea

> Objetivos
>
> - Elegir la métrica según la tarea
> - Leer un tablero sin engañarse

Cada tarea tiene su número

«Precisión del 92 %» no significa nada sin decir de qué. Cada tipo de tarea tiene una métrica que corresponde a **lo que cuesta el error**.

Tarea	Métrica	Por qué esa	Umbral inicial
Extraer campos	exactitud **por campo**	un total equivocado cuesta dinero; un proveedor con una letra distinta, no	total/subtotal/iva $\geq$ 98 %; otros $\geq$ 95 %
Recuperar	recall@5	si la ficha no está entre las 5, el modelo no puede citarla	$\geq$ 0,85
Responder con cita	fuente correcta + contiene/no contiene	una respuesta correcta con cita equivocada no es auditable	$\geq$ 90 %
Redactar acta	secciones presentes + decisiones correctas (lista)	el estilo es opinable; las decisiones no	100 % decisiones
Permisos	fichas indebidas en contexto	cualquier fuga es grave	**0**
Clasificar	matriz de confusión	qué se confunde con qué dice dónde falla	según clase
Latencia	TTFT p95	lo que siente el usuario en el peor caso normal	$\leq$ 5 s en B
Costo	tokens por tarea completada	prefill por paso $\times$ pasos	tendencia, no umbral

Leer un tablero sin engañarse

- **Promedios esconden.** 92 % global puede ser 99 % en totales y 70 % en proveedores.
- **p95, no media**, para latencia: la media de 2 s con un 5 % de 40 s es un sistema que la gente odia.
- **Tendencia sobre valor.** Un 90 % que baja es más grave que un 85 % estable.
- **Un cambio a la vez.** Si cambió el modelo y el prompt, el tablero no dice cuál fue.

> PROCEDIMIENTO · **Laboratorio 12.2 · el tablero**
>
> `tablero.py` que lee las corridas (`corrida-*.json`) y produce una tabla por tarea con valor actual, valor anterior, tendencia y umbral. Salida en Markdown para pegar en el reporte mensual (11.2).
> **Entregable:** `tablero.py` y el tablero con dos corridas. Commit: «Capa 12.2: tablero».
> Los umbrales de latencia cambian por perfil (A: $\leq$ 60 s en lote; B: $\leq$ 5 s; C: $\leq$ 2 s). Los demás no.

Cuándo NO usar esto **Métricas por tipo de tarea**

- No use BLEU/ROUGE ni similitud de embeddings para juzgar respuestas de negocio:

miden parecido, no corrección.

- No ponga umbrales sin haber medido el desacuerdo humano (7.4): no se le exige al sistema más de lo que dos personas acuerdan.

PRÁCTICA · **Práctica**

1. El p95 de TTFT subió de 3 a 9 s tras cambiar el embedding. ¿Tiene sentido? ¿Qué revisa?
2. Proponga la métrica para «resumir un contrato en 10 líneas».

Referencias

1. Módulo 7.4 — desacuerdo humano como techo de la métrica.

2. Liang, P. et al. (2022). *Holistic Evaluation of Language Models (HELM)*. arXiv:2211.09110 — por qué una sola métrica no describe un sistema.

12.3 · Trazado multipaso y el bucle traza → eval → corrección

> Objetivos
>
> - Encontrar el paso que falló en una traza
> - Cerrar el bucle con una corrección medida

Del síntoma al paso que falló

Con evals que fallan y trazas de cada paso (11.2), el diagnóstico deja de ser adivinanza:

```
eval ev-031 falló → traza …tr_01J → paso 1: buscar_documentos devolvió [contrato-2026-014#3, #9]→
 la ficha #7 no estaba entre las 5 → recall (6.2) → ¿por qué? la consulta "plazo de garantía"
no se parece a la cláusula, que dice "término de la garantía"
```

El bucle:

PROCEDIMIENTO · **Laboratorio 12.3 · tres fallas guiadas por trazas**

Se entregan tres evals que fallan (uno de recuperación, uno de extracción, uno de permisos). Para cada uno: la traza, el paso, la hipótesis, el cambio, la corrida completa antes/después, y la decisión. El de permisos es el importante: la corrección **no** es cambiar el prompt.
Entregable: `labs/12-03/bucle.md` con los tres ciclos completos. Commit: «Capa 12.3: bucle cerrado».
Nada.

Cuándo NO usar esto **Trazado multipaso y el bucle traza → eval → corrección**

- No corrija sin la traza «porque ya sé qué fue»: casi nunca es lo que uno cree (6.4).
- No acumule tres cambios antes de correr el set.

PRÁCTICA · **Práctica**

1. Un eval de acta falla: la decisión 3 no aparece. La traza muestra que la transcripción tenía la decisión. ¿Qué paso revisa?
2. ¿Cuándo un eval que falla se **descarta** en vez de corregir el sistema?

Referencias

1. Módulos 6.4 y 11.2 de este curso.

12.4 · Evals canario nocturnos y degradación

Objetivos

- Programar un canario que corra cada noche
- Detectar cambio de modelo o de fuente

Que el sistema se examine solo cada noche

El canario de 9.3 tenía tres pruebas. El **canario de evaluación** corre un subconjunto del set de evals (15 casos, los más sensibles: permisos, trampa, 5 facturas, 5 consultas) todas las noches, y compara con la noche anterior.

```
# /etc/cron.d/el-tornillo (añadir)
15 4 * * *  app  timeout 1h /opt/tornillo/canario-evals.sh >> /var/log/tornillo/canario-evals.log
     2>&1
```

```
#!/usr/bin/env bash
# canario-evals.sh — corre el subconjunto, compara con ayer, alerta si baja
set -u
python3 /opt/tornillo/evals.py --set /opt/tornillo/evals/canario --salida /var/lib/tornillo/
    canario-$(date +%F).json
python3 - <<'PY' || /opt/tornillo/alertar.sh "canario-evals: degradación (ver log)"
import json, glob, sys
f = sorted(glob.glob("/var/lib/tornillo/canario-*.json"))
hoy, ayer = json.load(open(f[-1])), json.load(open(f[-2])) if len(f) > 1 else None
n_hoy = sum(x["ok"] for x in hoy)
if any(x["tipo"] == "permisos" and not x["ok"] for x in hoy): print("PERMISOS FALLAN"); sys.exit
    (1)
if ayer and n_hoy < sum(x["ok"] for x in ayer): print(f"bajó: {n_hoy} < {sum(x['ok'] for x in
    ayer)}"); sys.exit(1)
print("canario-evals OK", n_hoy, "/", len(hoy))
PY
```

Qué detecta

Cambio silencioso	Cómo se ve
Se actualizó el runtime o el driver	latencia cambia; a veces la calidad
Alguien cambió el modelo «para probar»	hash distinto (9.3) + evals bajan
El cliente añadió un proveedor con otro formato	extracción baja en ese proveedor
Se reindexó con otro embedding	recall cae
Cambió el proveedor de identidad	permisos fallan → alerta inmediata

> PROCEDIMIENTO · **Laboratorio 12.4**
>
> Monte el canario, y **provoque** una degradación (cambie la cuantización del modelo a Q3). Lo que debe ver: la alerta a las 4:15 con el eval que bajó. Luego revierta y verifique que vuelve a verde.
> **Entregable:** `canario-evals.sh`, `evals/canario/` (15 casos), y las dos alertas (degradación y vuelta a verde). Commit: «Capa 12: el sistema se examina solo».

Qué se degrada en cada perfil

A — 15 casos tardan ~15 min de madrugada: bien

B — minutos

Cuándo NO usar esto **Evals canario nocturnos y degradación**

- No ponga los 60 casos en el canario nocturno: 15 sensibles bastan; los 60 se corren antes de cada cambio.
- No silencie una alerta «porque ya sé qué es»: se corrige o se ajusta el eval, nunca se apaga.

PRÁCTICA · **Práctica**

1. ¿Qué 15 casos elige para el canario de la ferretería? Justifique dos.
2. La alerta salta tres noches seguidas con el mismo eval. ¿Qué hace?

Referencias

1. Módulos 9.3 y 12.1 de este curso.
2. Google SRE Book — capítulo *Monitoring Distributed Systems*: https://sre.google/sre-book/monitoring-distributed-systems/