

IA local para PyMEs · Módulo 9

Automatización y orquestación

Flujos deterministas donde se pueda, modelo solo donde haya ambigüedad. Detectar la ruptura silenciosa.

Qué se construye en este módulo

🔗 Herramientas de flujo: fortalezas y límites reales

Decidir entre flujo visual y código

🔗 La regla determinista / probabilístico

Refactorizar un flujo dejando el modelo solo donde hace falta

🔗 Scraping responsable, tareas programadas y ruptura silenciosa

Programar tareas con canario y alerta

9.1 · Herramientas de flujo: fortalezas y límites reales

Herramientas de flujo: fortalezas y límites reales

Las herramientas de flujo (n8n, Node-RED y similares [VOLÁTIL – verificado: 2026-09]) permiten armar automatizaciones arrastrando cajas: «cuando llegue un PDF a esta carpeta → extraer campos → guardar → avisar». Son útiles y tienen límites que conviene decir antes de que el cliente se enamore de la interfaz.

En una tabla

Perfil	Qué pasa
A	El lote nocturno es la forma natural de usar el perfil A: el prefill lento no molesta de madrugada
B / C	Igual, y además en vivo

Cuándo NO usar esto

Herramientas de flujo: fortalezas y límites reales

- No meta la lógica del extractor dentro de cajas del flujo: no se prueba ni se versiona.
- No use un flujo para algo que corre cada minuto con carga: el motor de flujos añade latencia y memoria.
- No confíe en que «si falla, se ve en la interfaz»: se ve si alguien mira. Lección 9.3.

9.2 · La regla determinista / probabilístico

La regla determinista / probabilístico

Un modelo de lenguaje es caro, lento y **no repetible**: la misma entrada puede dar salidas distintas. Una función de Python es barata, rápida y da siempre lo mismo. Cada paso de un flujo se clasifica antes de construirlo:

Cuándo NO usar esto

La regla determinista / probabilístico

- No convierta en reglas algo que tiene 40 casos especiales: llegó el momento del modelo.
- No use el modelo «para que sea flexible» en un paso que un auditor tiene que poder reproducir.

9.3 · Scraping responsable, tareas programadas y ruptura silenciosa

Scraping responsable, tareas programadas y ruptura silenciosa

Un proveedor cambia el formato de su factura. La página de precios cambia una clase CSS. El modelo se actualizó solo. Ninguno de esos casos lanza un error: el sistema **sigue corriendo y produce basura** con toda confianza. Se llama **ruptura silenciosa** y es el modo de fallo número uno de las automatizaciones en producción.

El comando, copiable

```
# /etc/cron.d/el-tornillo
0 2 * * * app timeout 3h /opt/tornillo/ingesta.sh >> /var/log/tornillo/ingesta.
log 2>&1
0 6 * * * app /opt/tornillo/resumen.sh >> /var/log/tornillo/resumen.
log 2>&1
30 3 * * * app /opt/tornillo/canario.sh >> /var/log/tornillo/canario.
log 2>&1
```

Si no corre desde cero con un comando, no entra al curso.

Cuándo NO usar esto

Scraping responsable, tareas programadas y ruptura silenciosa

- No haga scraping de un sitio que ofrece API o descarga de datos: use eso.
- No programe tareas sin timeout: una tarea colgada bloquea la siguiente noche.

Entregables del módulo

- el flujo exportado, y la captura del resumen de las 6 a. m.
- el flujo refactorizado y la tabla antes/después.
- `cron.d`, `canario.sh`, `alertar.sh`, y la captura de la alerta.

**Flujos deterministas donde
se pueda, modelo solo
donde haya ambigüedad.**