

IA local para PyMEs · Módulo 5

# Ingeniería de contexto

---

La ventana es un recurso escaso. Salidas con esquema, herramientas pequeñas y poda en agentes largos.

# Qué se construye en este módulo

📦 **De prompts a contexto: la ventana como recurso escaso**

Presupuestar la ventana por secciones

📦 **Salidas estructuradas y validación por esquema**

Definir un JSON Schema y validar cada salida

📦 **Diseño de herramientas y poda de contexto**

Escribir herramientas pequeñas con contratos exactos

---

## **5.1 · De prompts a contexto: la ventana como recurso escaso**

# De prompts a contexto: la ventana como recurso escaso

Durante dos años el oficio fue redactar bien la instrucción. Sigue importando, pero dejó de ser el problema principal. El problema principal es **qué entra en la ventana, en qué orden, y qué se saca**: instrucciones, documentos recuperados, historial, resultados de herramientas, esquemas. Eso es **ingeniería de contexto**: tratar la ventana como un recurso escaso y presupuestado.

# En una tabla

| Partida                      | Qué es                                 | Tope típico (ventana 8k) | Prioridad |
|------------------------------|----------------------------------------|--------------------------|-----------|
| Instrucciones del sistema    | quién es, qué puede, formato de salida | 600–1 200                | fija      |
| Esquema de salida            | el JSON Schema, si aplica              | 200–500                  | fija      |
| Contexto recuperado (RAG)    | chunks relevantes con su cita          | 3 000–4 500              | variable  |
| Historial de la conversación | turnos anteriores, potenciales         | 500–1 500                | variable  |
| Resultados de herramientas   | último resultado, resumen              | 300–800                  | variable  |
| Reserva para la respuesta    | lo que el modelo va a escribir         | 800–1 500                | fija      |

# El comando, copiable

|                       |                                                      |
|-----------------------|------------------------------------------------------|
| [instrucciones fijas] | ← idéntico en todas las llamadas: se cachea          |
| [esquema fijo]        | ← ídem                                               |
| [documentos]          | ← cambia por consulta                                |
| [historial podado]    |                                                      |
| [pregunta actual]     | ← lo más importante, al final, cerca de la respuesta |

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

Cuándo NO usar esto

## De prompts a contexto: la ventana como recurso escaso

- No «arregle» un mal RAG metiendo más chunks: el módulo 6 dice cómo arreglar la recuperación.
- No use la ventana de 128k como excusa para no presupuestar: el prefill de 100k tokens en el perfil B tarda minutos.

# Qué se degrada en cada perfil

## A · sin GPU

RAG con 2–3 chunks;  
sin historial largo

## B · 8–16 GB VRAM

RAG con 5–8 chunks;  
historial podado

## C · 24 GB+

holgura; aun así se  
presupuesta, porque  
el prefill se paga



---

## **5.2 · Salidas estructuradas y validación por esquema**

# Salidas estructuradas y validación por esquema

Si la respuesta tiene estructura —campos de una factura, una decisión sí/no con motivo, una lista de acciones—, se pide **con esquema** y se **valida**. Lo que no cumple el esquema se rechaza y se reintenta o se encola para un humano. Nunca se «parsea» texto libre con expresiones regulares esperando lo mejor.

# En una tabla

| Perfil | Qué pasa                               |
|--------|----------------------------------------|
| A      | Igual, más lento. La validación es CPU |
| B / C  | Igual                                  |

# El comando, copiable

```
cd ~/labs/el-tornillo && mkdir -p labs/05-02 && cd labs/05-02 && pip install -q  
jsonschema
```

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

Cuándo NO usar esto

## Salidas estructuradas y validación por esquema

- No ponga esquema a una tarea de redacción (un acta, un resumen): ahí la salida es texto, y se valida por longitud y por presencia de secciones, no por JSON.
- No confíe en la restricción del runtime para reglas de negocio (`subtotal + iva = total`): eso se valida en código.

---

## 5.3 · Diseño de herramientas y poda de contexto

# Diseño de herramientas y poda de contexto

Un modelo elige herramientas leyendo sus **descripciones**. Una descripción vaga produce llamadas equivocadas; una herramienta que hace tres cosas produce argumentos mezclados. Reglas:

# En una tabla

| Regla                                   | Mal                                         | Bien                                                                                     |
|-----------------------------------------|---------------------------------------------|------------------------------------------------------------------------------------------|
| Una herramienta, una acción             | <code>gestionar_factura(accion, ...)</code> | <code>buscar_factura,</code><br><code>listar_pendiente</code>                            |
| Entradas mínimas y tipadas              | <code>consulta: string</code>               | <code>numero: string (</code><br><code>date, hasta: date</code>                          |
| Descripción con cuándo <b>no</b> usarla | «Busca facturas»                            | «Busca UNA factu<br>ro exacto. Para ra<br>use <code>listar_factu</code>                  |
| Salida en JSON, corta                   | vuelca 200 filas                            | <code>devuelve</code><br><code>total_disponible</code><br><code>sor</code>               |
| Errores como datos                      | lanza excepción                             | <code>{"error": "no_en</code><br><code>"sugerencia": "v</code><br><code>prefijo"}</code> |
| Solo lectura por defecto                | cualquier herramienta escribe               | las que escriben<br>san por aprobaci                                                     |



# El comando, copiable

```
HERRAMIENTAS = [  
  {"name": "buscar_documentos",  
    "description": "Recupera fragmentos de documentos del cliente relevantes para una  
    pregunta, con su cita. Respeta los permisos del usuario. No sirve para buscar un  
    número exacto: use buscar_factura.",  
    "parameters": {"type": "object", "properties": {"pregunta": {"type": "string", "maxLength":  
      300}, "coleccion": {"type": "string", "enum": ["facturas", "contratos", "actas", "todas  
      "]}, "k": {"type": "integer", "minimum": 1, "maximum": 8}}, "required": ["pregunta"]}},  
  {"name": "buscar_factura",  
    "description": "Devuelve UNA factura por su número exacto (FV-AAAA-NNNNNN). Si no  
    existe, devuelve error no_encontrada.",  
    "parameters": {"type": "object", "properties": {"numero": {"type": "string", "pattern": "^  
    FV-\\d{4}-\\d{6}$"}}, "required": ["numero"]}},  
  {"name": "crear_borrador_acta",  
    "description": "ESCRIBE: crea un borrador de acta a partir de una transcripción.  
    Requiere aprobación humana antes de guardarse. No la use para resumir: use  
    responder.",  
    "parameters": {"type": "object", "properties": {"transcripcion_id": {"type": "string"}, "  
    plantilla": {"type": "string", "enum": ["comite", "proveedor"]}} "required": ["
```

Cuándo NO usar esto

## Diseño de herramientas y poda de contexto

- No dé al modelo veinte herramientas: elige peor y gasta contexto en leerlas. Cinco a ocho por agente; si hacen falta más, es otro agente (módulo 10.2).
- No podes el historial de una conversación de un turno: no hay nada que podar.

# Qué se degrada en cada perfil



## A · sin GPU

La poda es más agresiva: ventana de 4k–8k



## B · 8–16 GB VRAM

Ventana de 8k–16k; resumen de resultados basta



## C · 24 GB+

Holgura; se sigue podando por costo de prefill

# Entregables del módulo

- el presupuesto de las tres tareas.
- `factura.schema.json`, `extraer.py`, y `labs/05-02/rechazos.md` con los motivos.
- `herramientas.json`, `podar.py` con prueba.

**La ventana es un  
recurso escaso.**