LANGCHAIN Y LANGGRAPH

Un mensaje, dos caminos

Una aplicación responde cuántas plazas quedan en un taller o pide que se aclare cuál.

> **¿Cuántas plazas quedan para Fotografía nocturna?**

Si identifica el taller	**consulta las plazas**
Si no lo identifica	**pide una aclaración**

EL CASO

Un catálogo y dos mensajes

Datos ficticios en memoria. Cada mensaje debería llevar a un camino distinto.

Plazas disponibles en el catálogo

Fotografía nocturna	3
Cerámica	0

Mensajes de la persona

M1 ¿Cuántas plazas quedan para Fotografía nocturna?

M2 ¿Cuántas plazas quedan?

M2 no dice el taller: la aplicación no elige uno al azar.

LANGCHAIN

El modelo solo elige un nombre

LangChain conecta con el modelo y le pide una respuesta con forma fija: un taller admitido o nada.

El modelo recibe una instrucción y el mensaje

> Elige el taller pedido: Fotografía nocturna o Cerámica. Si falta o hay duda, devuelve una selección vacía (None).

M1	workshop_name = Fotografía nocturna
M2	workshop_name = None

Selecciones ilustrativas, no medidas. Una selección con forma válida aún puede interpretar mal el mensaje.

LANGGRAPH

Estado, nodos y aristas

LangGraph representa el recorrido como un grafo: pasos conectados por transiciones.

Estado los datos compartidos	message, workshop_name y reply
Nodos los pasos	identify, answer y clarify: funciones de Python
Aristas qué paso sigue	La condicional elige camino según el estado

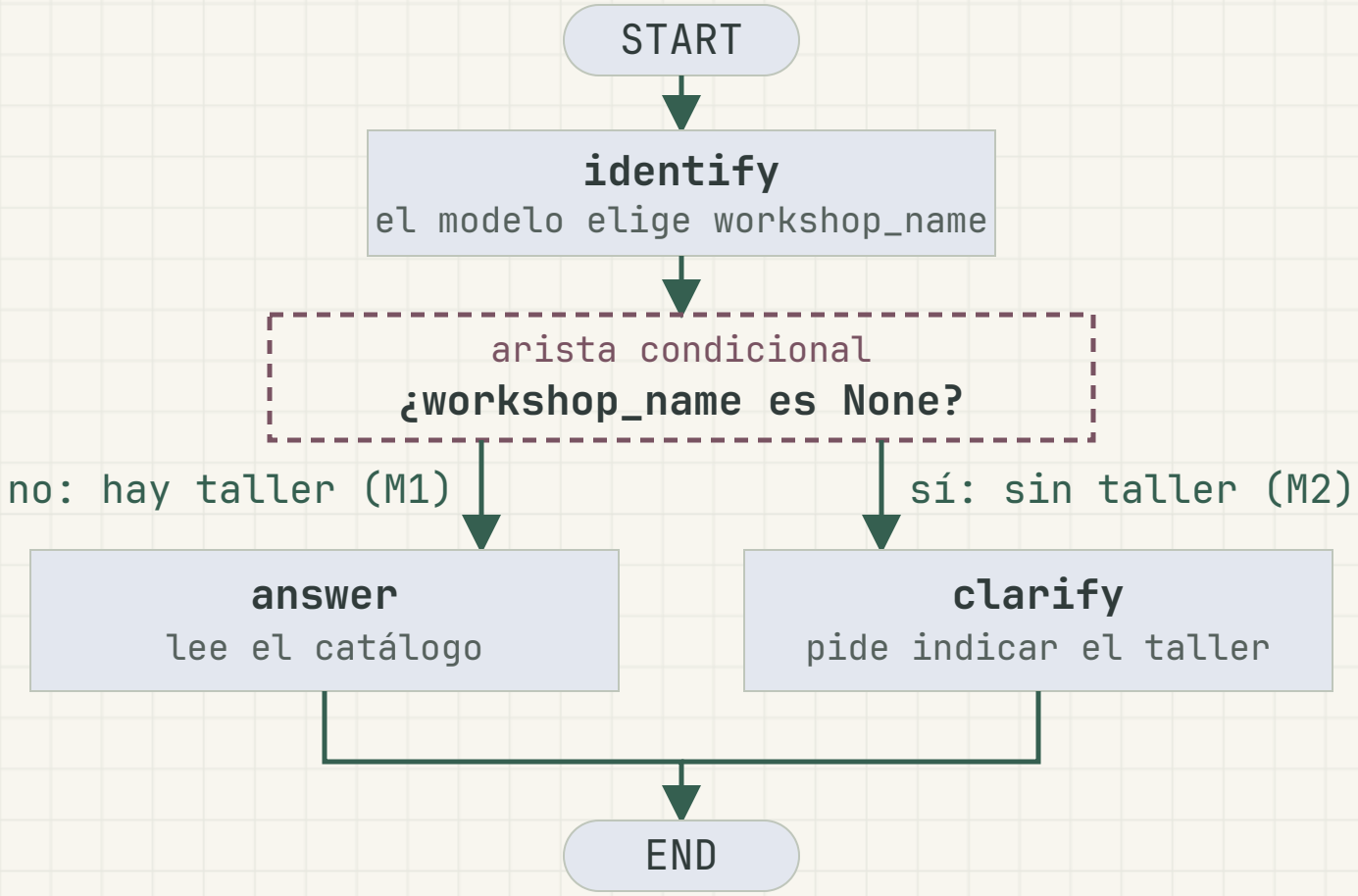
Estado inicial al enviar M1

```
message = "¿Cuántas plazas quedan para Fotografía nocturna?"
workshop_name = None
reply = ""
```

EL GRAFO

Una decisión, dos caminos

Una función de Python mira workshop_name y elige el nodo siguiente. El modelo no decide el recorrido.



CAMINO CON TALLER

M1 llega a la consulta

En cada paso cambia una parte del estado.

- **1. identify**

  El modelo lee M1 y elige Fotografía nocturna.

  `workshop_name` pasa de None a Fotografía nocturna

- **2. Arista condicional**

  `workshop_name` no es None: sigue answer.

- **3. answer**

  Busca Fotografía nocturna en el catálogo: 3 plazas.

  `reply` deja de estar vacía

Plazas disponibles en Fotografía nocturna: 3.

CAMINO SIN TALLER

Sin taller no es cero plazas

M2 no nombra ninguno y se aclara. Cerámica sí es un taller, aunque no queden plazas.

M2 · sin taller

identify deja workshop_name = None.

Sigue clarify:

> **Indica el taller: Fotografía nocturna o Cerámica.**

No se consulta ninguna plaza.

M3 · Cerámica

Mensaje ilustrativo: ¿Cuántas plazas quedan para Cerámica?

workshop_name = Cerámica; sigue answer:

> **Plazas disponibles en Cerámica: 0.**

Hay selección: el 0 es un dato del catálogo.

None (taller sin identificar) y 0 plazas (taller identificado, sin sitio) son cosas distintas.

LA IDEA

El modelo interpreta, el código decide

Un grafo hace explícitos pasos y datos.

1 **El modelo interpreta**
Convierte el mensaje en una selección: un taller o None.

2 **El código decide**
Una condición de Python elige el camino y construye la respuesta.

3 **El grafo muestra el recorrido**
Estado, nodos y aristas: qué ocurre y con qué datos.

4 **Cuándo compensa**
Aquí bastaría un if; el grafo ayuda cuando hay más caminos.