



Proyecto de Grado - Reunión de seguimiento

ARCHIA

ÍNDICE

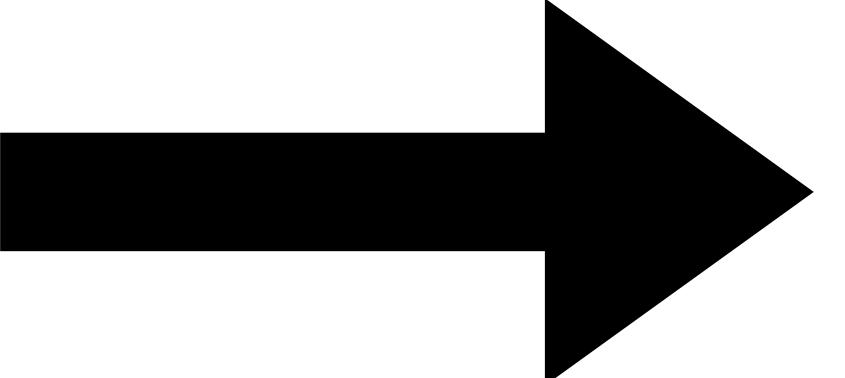
- Refactor
- Mejora de generación de gráficos
- Mejora de respuestas con varios requisitos
- Preservación del lenguaje del usuario
- Demostración



REFACTOR

ANTES

graph.py



DESPUÉS

graph/

- **__init__.py:**
- **workflow.py:** Define la estructura del grafo.
- **state.py:** Define el estado del grafo y las validaciones de salida de cada nodo
- **consts.py:** Contiene prompts del sistema y esquemas para tácticas y diagramas.
- **utils.py:** Funciones auxiliares que gestionan tácticas de arquitectura y diagramas.
- **resources.py:** Recursos compartidos (LLM, RAG y la persistencia del grafo.)
- **nodes/:** Directorio de los nodos.



NODES/

- **supervisor.py:** El orquestador principal que decide cuál es el siguiente paso basado en la intención del usuario.
- **classifier.py:** Determina la intención e idioma del mensaje inicial.
- **investigator.py:** Realiza búsquedas del RAG sobre conceptos arquitectónicos.
- **asr.py:** Genera y propone ASR.
- **style.py:** Sugiere los estilos arquitectónicos más adecuados para cumplir con los ASR.
- **tactics.py:** Propone tácticas de diseño específicas para satisfacer los atributos de calidad.
- **diagram.py:** Compila ASR, estilo, tácticas y contexto adicional para generar, diagrama.
- **evaluator.py:** Analiza y critica las propuestas arquitectónicas buscando riesgos o mejoras.
- **unifier.py:** Consolida todas las respuestas de los nodos previos en un mensaje final coherente para el usuario.
- **tools.py:** Define herramientas adicionales que los nodos pueden invocar durante su ejecución.

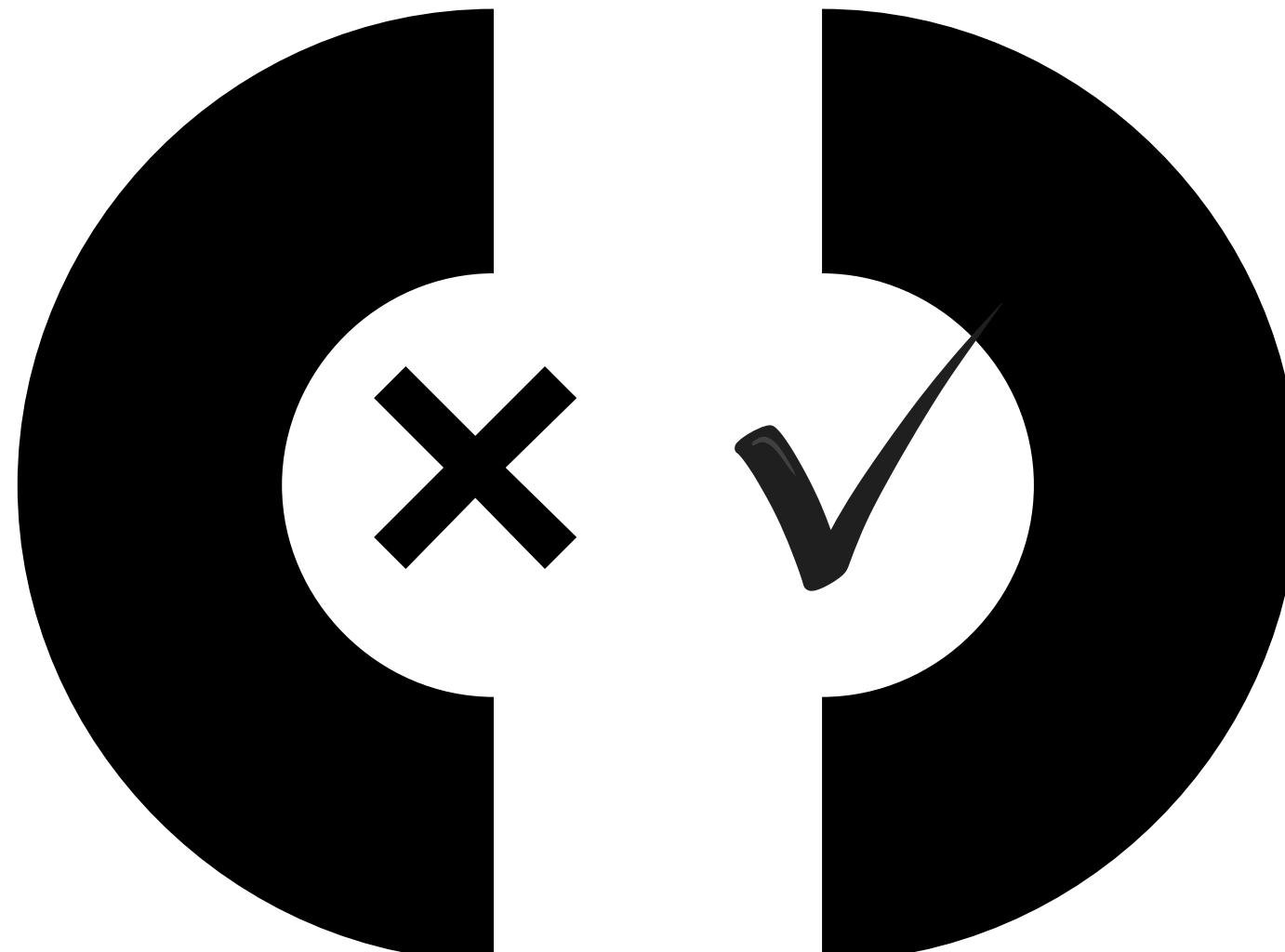
BENEFICIOS

- **Mantenibilidad:** Código distribuido en nodos y módulos pequeños facilita localizar, entender y modificar cada responsabilidad.
- **Escalabilidad funcional:** Agregar o reemplazar nodos es tan simple como añadir un archivo y conectarlo en workflow.py.
- **Pruebas y depuración:** Cada nodo y helper puede probarse de forma aislada.

SQLITESAVER → MEMORYSAVER

PROBLEMA

- SqliteSaver retorna un context manager, no una instancia directa
- El grafo se compilaba fuera de un with generando errores como:
 - AttributeError:
 __enter__
 - conexiones cerradas
 - SQLite (file-based) generaba locks en entorno async (FastAPI/Uvicorn)

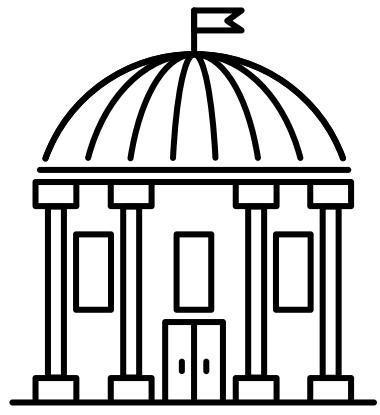


SOLUCION

- Reemplazo por MemorySaver
- Sin context management
- Sin archivos físicos
- Compatible directamente con graph.compile()

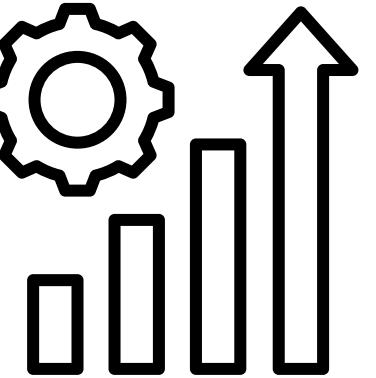


QUE GANAMOS



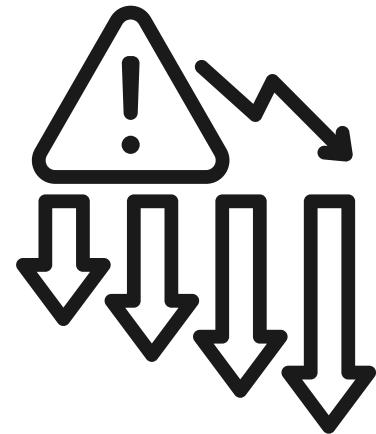
Arquitectura limpia:

- Single Responsibility
- Fuente única de verdad para el estado
- Separación lógica vs wiring



Escalabilidad para ADD 3.0

- Fácil agregar nuevos nodos
- Fácil extender supervisor
- Prompts desacoplados de lógica



Reducción de Riesgos

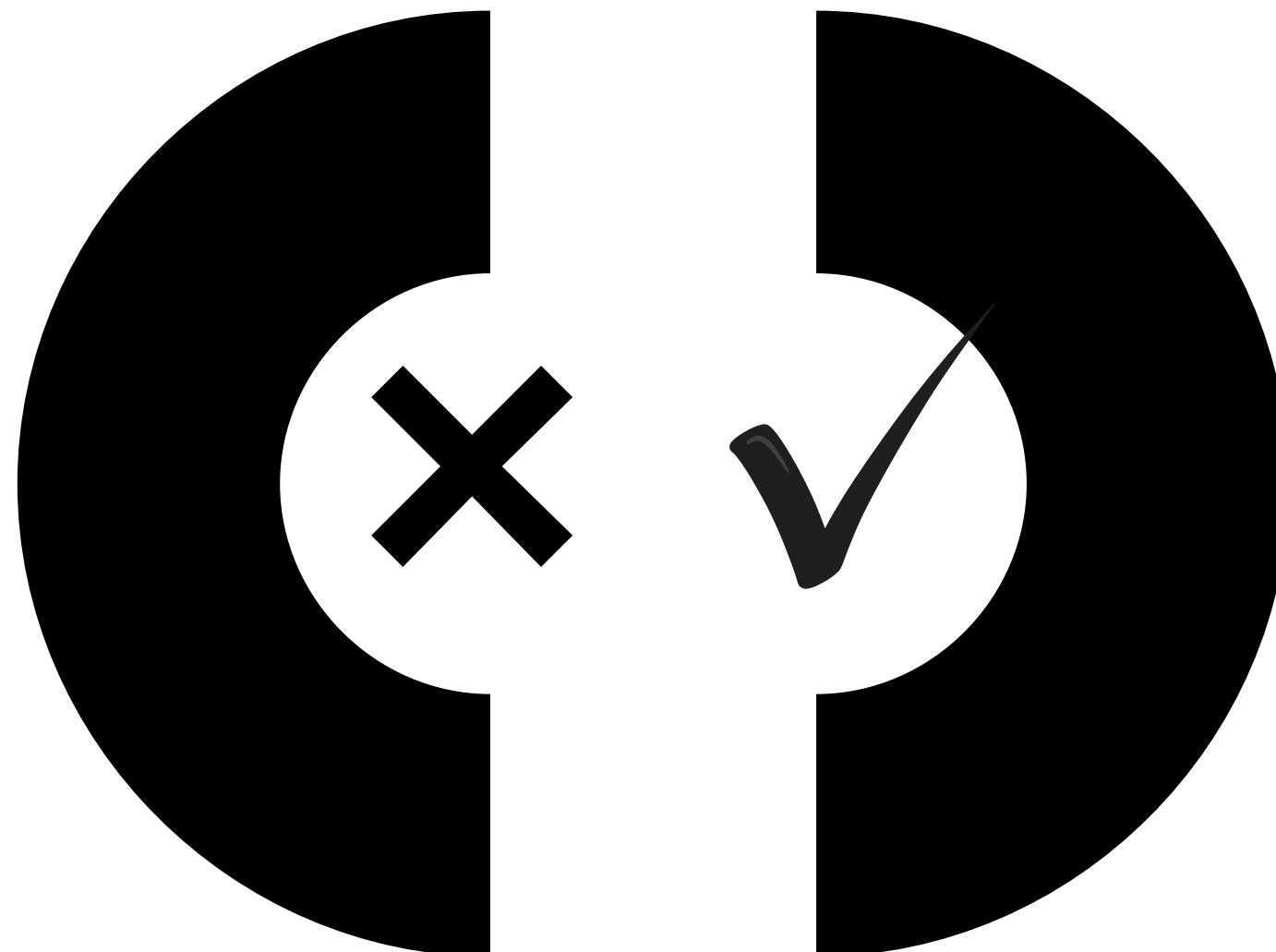
- No más imports circulares
- No más fallos por .env
- Checkpointing estable



MERMAID → GRAPHVIZ

PROBLEMA

- **Renderizado en el frontend:** Si fallaba la sintaxis o la librería JS, el usuario no veía nada
- **Sin validación desde backend:** No había control de errores antes de enviarlo
- **Diagrama generado solo con la pregunta actual:** Ignoraba ASR, estilos, tácticas y contexto acumulado



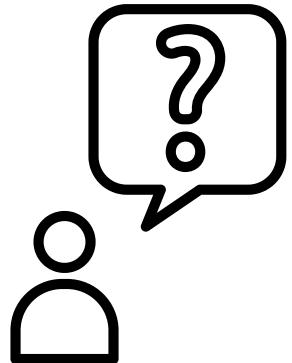
SOLUCION

- **Pipeline completo server-side:** Generación DOT → sanitización → renderizado SVG
- **Diagrama basado en contexto arquitectónico completo:** ASR + estilo + tácticas + memoria + RAG
- **Validación en backend:** Errores capturados.
- **Salida portable:** Salida SVG sin dependencias JS.



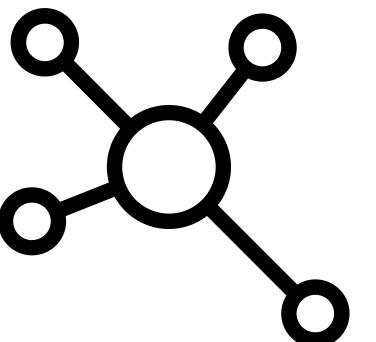
RESPUESTAS CON VARIOS REQUISITOS

RESPUESTAS CON VARIOS REQUISITOS + LENGUAJE



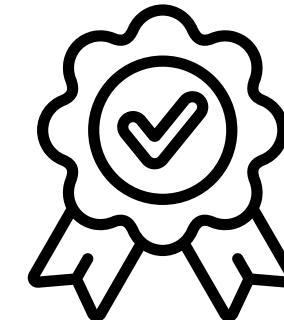
Antes:

- Las respuestas con varios requisitos (ASR, estilos, tácticas o diagramas en un solo prompt) se cortaban o generaban una respuesta incorrecta.
- Se generaban diagramas cuando no se pedían.
- A veces se retornaba el mismo prompt (eco).
- El lenguaje podía cambiar en ejecución.



Solución Multi-intent

- Se añaden campos al state (requested, completed y pending)
- Se añadió un Scheduler multi-intent al supervisor con helpers nuevos (infer_requested, augment_completed, append_unique.)
- Los nodos van primero al supervisor antes que al unifier
- Se modificó el unifier para mostrar las respuestas completas



Solución Lenguaje:

- El lenguaje de respuesta usado en todo agente espacializado se toma del state[language]

Soluciones al diagrama

- Se cambió la detección del intent, forzandolo solo a usar palabras clave que denotaran la necesidad del diagrama ("diagrama", "graphviz", "visualización", etc.)

Solución Eco:

- Se cambió el scheduler para que si un ASR era pedido y aún no se había generado, se priorizara este.



ANTES

Tú

Tengo una aplicación en Flutter que se comunica con un back en Spring. Ahora mismo, cuando se tiene más de 10000 usuarios, se está callendo. Quiero que en esas horas pico el backend pueda soportar una carga de 15000 usuarios. El back está desplegado en una maquina de Google Cloud de la categoria micro. Tiene ubuntu. Quiero que generes los ASRs correspondientes, así como estilos y tacticas que pueda implementar para cumplir con ese ASR.

Asistente

...

Aquí tienes el diagrama generado a partir del ASR, el estilo y las tácticas seleccionadas.

Here is the Mermaid script for this diagram. You can copy & paste it into the Mermaid live editor (<https://mermaid.live>), a VS Code Mermaid plugin, or any compatible renderer:

```
graph LR
client["Flutter App"]
gcp_lb["GCP Load Balancer"]
api_gateway["API Gateway"]
rate_limiter["Rate Limiter"]
cb_proxy["Circuit Breaker"]
autoscaler["Autoscaler"]
appA["Spring App Instance A"]
appB["Spring App Instance B"]
appC["Spring App Instance C"]
cache_redis["Redis Response Cache"]
session_redis["Redis Session Store"]
db_primary["Primary Database"]
db_read1["Read Replica 1"]
db_read2["Read Replica 2"]
queue["Message Queue"]
workerA["Background Worker"]
monitoring["Monitoring"]
alerting["Alerting"]
logging["Central Logging"]
health["Health Checks"]
infra_vm["Current Micro VM"]
app_legacy["Existing Spring App"]
gke_cluster["GKE Cluster"]
storage["Object Storage"]

tactic_autoscale["Tactic Autoscale Horizontal"]
tactic_vresize["Tactic Vertical Resize VM"]
```



DESPUÉS

Tú

Tengo una aplicación en Flutter que se comunica con un back en Spring. Ahora mismo, cuando se tiene más de 10000 usuarios, se está callendo. Quiero que en esas horas pico el backend pueda soportar una carga de 15000 usuarios. El back está desplegado en una maquina de Google Cloud de la categoria micro. Tiene ubuntu. Quiero que generes los ASRs correspondientes, así como estilos y tacticas que pueda implementar para cumplir con ese ASR.

Asistente

...

ASR: ASR complete: Cuando 15.000 usuarios concurrentes externos intenten participar en la flash sale y generen un pico sostenido de 3.000 RPS (con ráfagas de hasta 5.000 RPS por hasta 2 minutos) hacia el backend en producción, el servicio Spring que atiende las APIs críticas debe seguir respondiendo sin caer, manteniendo latencia end-to-end $p95 < 500$ ms ($p99 < 2$ s), tasa de errores < 1% y disponibilidad $\geq 99.9\%$ durante la ventana de la venta (30–120 minutos).

Scenario: Source: Usuarios externos usando la aplicación Flutter (clientes finales). Stimulus: Inicio de la flash sale con 15.000 usuarios concurrentes que generan hasta 3.000 solicitudes por segundo sostenidas y ráfagas puntuales de hasta 5.000 RPS. Environment: Entorno de producción durante la ventana pico de la flash sale (30–120 minutos); backend desplegado actualmente en una VM micro de Google Cloud sobre Ubuntu, tráfico por Internet público. Artifact: Backend Spring que expone las APIs críticas (listado y detalle de productos, carrito, add-to-cart, checkout, autenticación). Response: El backend debe atender las peticiones sin caer, priorizar y asegurar la completitud de operaciones críticas (p. ej. checkout), y degradar o poner en cola funciones no críticas si es necesario para mantener los SLOs. Response Measure: Sostener 3.000 RPS (capacidad de absorber ráfagas hasta 5.000 RPS por ≤ 2 min) durante la ventana; latencia end-to-end $p95 < 500$ ms y $p99 < 2$ s; tasa de errores < 1% medido en APIs críticas; disponibilidad durante la ventana $\geq 99.9\%$.

Estilos arquitectónicos: He identificado dos estilos arquitectónicos candidatos para tu ASR:

1. Contenedores orquestados (Kubernetes / GKE)

- Impact: Permite escalado horizontal controlado (HPA/VPA, Cluster Autoscaler) y configuraciones finas de recursos, networking y resiliencia; facilita aplicar tácticas: bulkheads, circuit breakers, connection pooling, service mesh para observabilidad, rate limiting y retries; soporta despliegues canary/rolling y control de recursos por pod para mantener latency y uptime. Contras: mayor complejidad operativa y coste de gestión; requiere empaquetado en contenedores y conocimiento de la plataforma. Trade-offs: mejor control y predictibilidad para mantener Latencia P95 <500ms y autoscaling <2min, pero exige inversión en infra y operaciones.

2. Serverless containers / Platform managed (Cloud Run / App Engine)

- Impact: Ofrece escalado automático transparente y reducción de operación (ideal para cargas variables y despliegue rápido); reduce esfuerzo operativo y puede escalar a muchas instancias sin gestionar nodos, ayudando a cumplir requisitos de carga y coste variable. Contras: menor control sobre afinidad de recursos, límites de concurrencia y posibilidad de cold starts que pueden afectar latencia P95/P99; manejo de conexiones DB y pooling más complejo (posible saturación de DB si muchas instancias se escalan rápido). Trade-offs: menor operational burden y buen costo en cargas variables, pero menos control para tácticas finas (bulkheads, service mesh) y ajuste de rendimiento extremo.

Recomendación: **Contenedores orquestados (Kubernetes / GKE)** porque: Kubernetes/GKE es la mejor opción para este ASR porque proporciona el control necesario sobre el escalado, la asignación de recursos y las tácticas de resiliencia requeridas para mantener latency media <500ms para el 95% y tolerancia a fallo (99.9% uptime). Con GKE puedes ajustar HPA/VPA y Cluster Autoscaler para responder dentro del umbral de <2 minutos, aplicar bulkheads y pools de conexión por servicio para evitar colapsos al alcanzar 15.000 usuarios, introducir colas y throttling, y usar service mesh y herramientas de observabilidad para detectar y diagnosticar en los plazos exigidos. Aunque tiene mayor coste operativo, ese control reduce el riesgo de degradación o caídas que se experimentan hoy en una VM micro y permite optimizar el rendimiento y la disponibilidad de forma predecible; Cloud Run es una alternativa válida si se prioriza rapidez de operación y simplicidad, pero sus limitaciones (cold starts, gestión de conexiones a BD y menos control de recursos) lo hacen menos adecuado para garantizar los SLAs en picos altos sin cambios arquitectónicos adicionales.



LANGSMITH

Runs Threads Evaluators Automations Insights

Retention 14d Dashboard Alerts + New

Default View Last 2 days Traces Runs + Add filter

Reset Save View Display Format

<input type="checkbox"/>	<input checked="" type="checkbox"/> Name	Input	Output	Error	Start Time	Latency	Dataset	Annotation Queue	Tokens	Cost	First Token	Tags	<input type="button"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/> LangGraph	human: Tengo un star...	ai: SOURCES: - S...		2/19/2026, 8:...	⌚ 237.73s	<input type="checkbox"/>	<input type="checkbox"/>	27,597	\$0.03369175			
<input type="checkbox"/>	<input checked="" type="checkbox"/> LangGraph	human: Tengo una apl...	ai: SOURCES: - (n...		2/18/2026, 7:...	⌚ 303.81s	<input type="checkbox"/>	<input type="checkbox"/>	25,523	\$0.0332485			
<input type="checkbox"/>	<input checked="" type="checkbox"/> LangGraph	human: Genera un dia...	human: Genera u...		2/18/2026, 3:...	⌚ 62.75s	<input type="checkbox"/>	<input type="checkbox"/>	3,948	\$0.0063385			
<input type="checkbox"/>	<input checked="" type="checkbox"/> LangGraph	human: Genera un dia...	human: Genera u...		2/18/2026, 2:...	⌚ 57.34s	<input type="checkbox"/>	<input type="checkbox"/>	3,694	\$0.0058305			
<input type="checkbox"/>	<input checked="" type="checkbox"/> LangGraph	human: Genera un dia...	human: Genera u...		2/18/2026, 2:...	⌚ 35.24s	<input type="checkbox"/>	<input type="checkbox"/>	3,576	\$0.0055945			
<input type="checkbox"/>	<input checked="" type="checkbox"/> LangGraph	human: Genera un dia...	human: Genera u...		2/18/2026, 2:...	⌚ 35.13s	<input type="checkbox"/>	<input type="checkbox"/>	3,412	\$0.0052665			
<input type="checkbox"/>	<input checked="" type="checkbox"/> LangGraph	human: Genera un dia...	human: Genera u...		2/18/2026, 2:...	⌚ 48.86s	<input type="checkbox"/>	<input type="checkbox"/>	3,593	\$0.0056285			
<input type="checkbox"/>	<input checked="" type="checkbox"/> LangGraph	human: Tengo una apl...	ai: SOURCES: - (n...		2/17/2026, 7:...	⌚ 207.28s	<input type="checkbox"/>	<input type="checkbox"/>	23,651	\$0.0307645			
<input type="checkbox"/>	<input checked="" type="checkbox"/> LangGraph	human: Tengo una sta...	ai: SOURCES: - (n...		2/17/2026, 7:...	⌚ 356.76s	<input type="checkbox"/>	<input type="checkbox"/>	37,667	\$0.04716425			
<input type="checkbox"/>	<input checked="" type="checkbox"/> LangGraph	human: ¿Qué tácticas...	ai: SOURCES: - (n...		2/17/2026, 7:...	⌚ 170.52s	<input type="checkbox"/>	<input type="checkbox"/>	15,351	\$0.02061325			
<input type="checkbox"/>	<input checked="" type="checkbox"/> LangGraph	human: Genera un dia...	human: Genera u...		2/17/2026, 6:...	⌚ 65.16s	<input type="checkbox"/>	<input type="checkbox"/>	4,690	\$0.0078225			
<input type="checkbox"/>	<input checked="" type="checkbox"/> LangGraph	human: Tengo una sta...	ai: SOURCES: - (n...		2/17/2026, 6:...	⌚ 163.36s	<input type="checkbox"/>	<input type="checkbox"/>	19,579	\$0.02548875			
<input type="checkbox"/>	<input checked="" type="checkbox"/> LangGraph	human: Genera un dia...	human: Genera u...		2/17/2026, 6:...	⌚ 35.41s	<input type="checkbox"/>	<input type="checkbox"/>	3,077	\$0.0045965			
<input type="checkbox"/>	<input checked="" type="checkbox"/> LangGraph	human: Genera un dia...	human: Genera u...		2/17/2026, 6:...	⌚ 40.49s	<input type="checkbox"/>	<input type="checkbox"/>	3,748	\$0.00604525			
<input type="checkbox"/>	<input checked="" type="checkbox"/> LangGraph	human: Genera un dia...	human: Genera u...		2/17/2026, 6:...	⌚ 41.95s	<input type="checkbox"/>	<input type="checkbox"/>	3,813	\$0.00617525			



LANGSMITH

The screenshot displays two application interfaces side-by-side: ArchIA and LangGraph.

ArchIA (Left):

- Header:** ArchIA, ID, Runs, Threads, Evaluators, A
- Toolbar:** Default View, Last 2 days, Traces, Runs
- Table:** Shows a list of runs for "LangGraph". Each row contains a checkbox, a green checkmark icon, the name "LangGraph", and the input text "human: Tengo un".

LangGraph (Right):

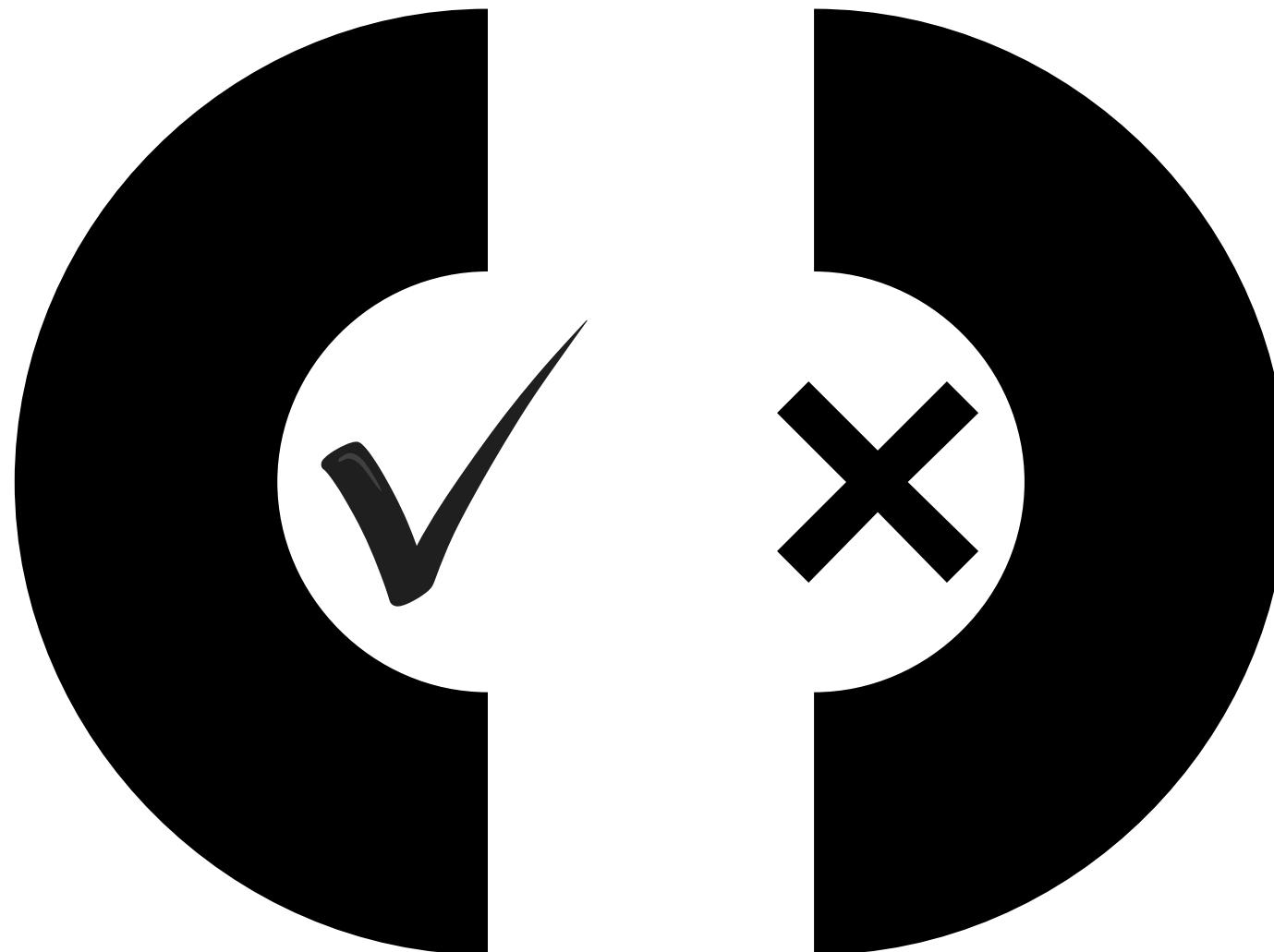
- Header:** LangGraph, ID
- Toolbar:** Run in Studio, Thread, Add to, Up, Down
- Section:** TRACE
- Logs:** A detailed trace log showing interactions between components:
 - agent: 65.94s, 5,540 tokens
 - call_model: 65.94s, 5,540 tokens
 - Prompt: 0.00s
 - ChatOpenAI (gpt-5-mini): 65.93s, 5,540 tokens
 - should_continue: 0.00s
 - supervisor: 0.00s
 - router: 0.00s
 - asr: 40.68s, 4,027 tokens
 - VectorStoreRetriever: 0.26s
 - ChatOpenAI (gpt-5-mini): 40.41s, 4,027 tokens
 - supervisor: 0.00s
 - router: 0.00s
 - style: 17.06s, 3,299 tokens
 - ChatOpenAI (gpt-5-mini): 17.05s, 3,299 tokens
 - supervisor: 0.00s
- Run Details:** START TIME: 02/19/2026, 08:17:28 AM GMT-5; END TIME: 02/19/2026, 08:21:25 AM GMT-5; STATUS: Success; TOTAL TOKENS: 27,597 tokens / \$0.03369175; LATENCY: 237.73s; TYPE: Chain
- Input:** Tengo un startup fintech que está desarrollando una plataforma web para otorgar microcréditos en Latinoamérica. Actualmente el proceso de evaluación crediticia tarda entre 48 y 72 horas porque depende de revisión manual. Quiero que el nuevo sistema reduzca el tiempo de aprobación a menos de 10 minutos utilizando modelos de scoring automatizados. Sin embargo, la regulación financiera del país exige que toda decisión crediticia pueda ser auditada y explicada. El sistema deberá integrarse con bases de datos gubernamentales para validar identidad y con al menos dos burós de crédito. La plataforma será utilizada inicialmente en México y Colombia. El presupuesto es limitado, por lo que no se pueden usar soluciones enterprise de alto costo. Además, el equipo técnico está compuesto por 4 desarrolladores y 1 científico de datos. Los usuarios objetivo son personas no bancarizadas que acceden principalmente desde dispositivos móviles con conexiones inestables. Actualmente, se tiene una implementación inicial con una capacidad de 500.000 usuarios. La empresa espera escalar a 1 millón de usuarios en los próximos 3 años. La empresa también ha estado trabajando con AWS, por lo que se quiere desplegar el nuevo sistema en este servicio, usando una solución económica. Dame 1 ASR de escalabilidad, y para ese mismo ASR genera dos estilos y dos tácticas de arquitectura que ayuden a cumplir ese ASR. Por último, genera un diagrama de componentes con esos estilos y tácticas y teniendo en cuenta el contexto dado.
- Bottom:** A note stating "Some runs have been hidden. Show 4 hidden runs" and a "Polly" button.



SOLUCIONADO VS POR MEJORAR

SOLUCIONADO

- **Soporte multi-intent:** Los usuarios ya pueden pedir varios requisitos en un solo prompt
- **Diagramas generados cuando es necesario:** Ya no se generan diagramas cuando no se piden explicitamente.
- **Preservación del Lenguaje:** Se responde siempre con el lenguaje usado en el ultimo mensaje del usuario

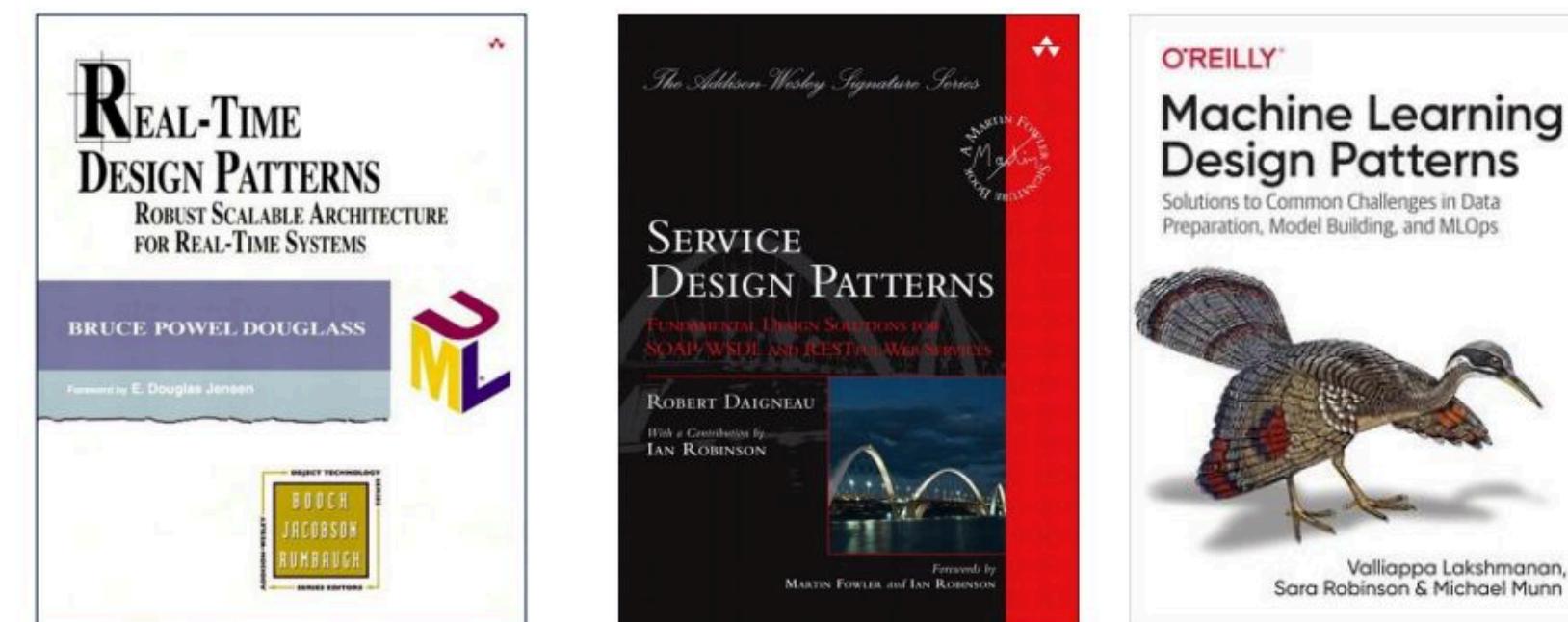
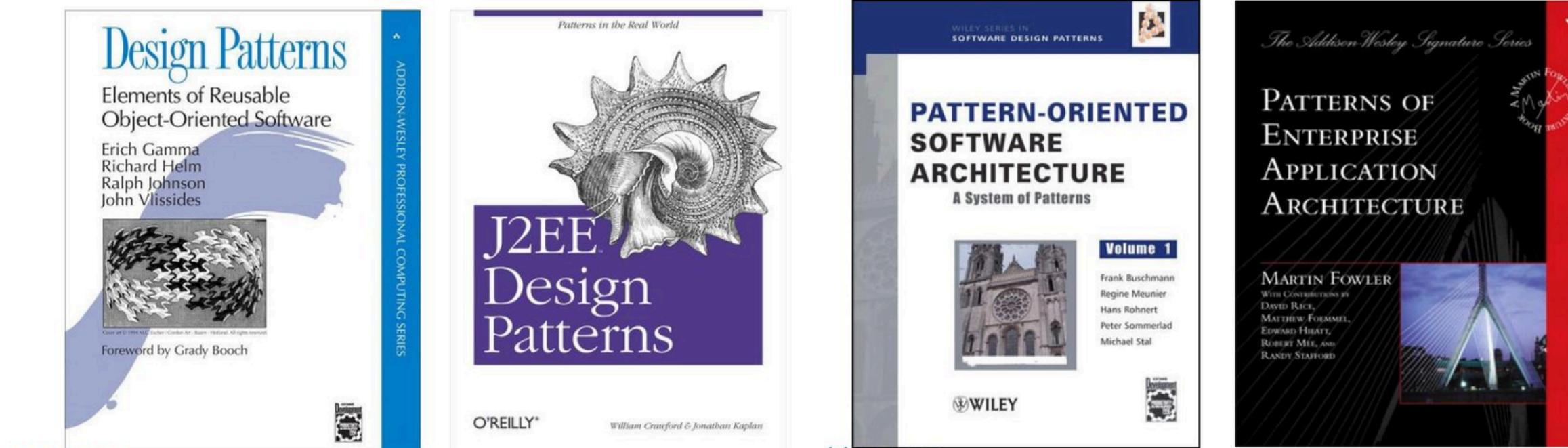


POR MEJORAR

- **Calidad de respuesta:** Las respuestas aún no están siendo tan específicas al contexto del usuario.
- **Tiempo de respuesta:** Hay prompts que toman hasta 350 segundos en responder.
- **Formato de Respuesta:** El formato de respuesta actual es difícil de leer y seguir.



¿CÓMO PODEMOS SEGUIR MEJORANDO LA CALIDAD DE LAS RESPUESTAS?



DEMOSTRACIÓN

REFERENCIAS

- Tencent Cloud (artículo de Techpedia)

Tencent Cloud. (2025, 26 de septiembre). How do conversational robots handle multi-intent input? Tencent Cloud Techpedia. Recuperado de <https://www.tencentcloud.com/techpedia/127525> <https://medium.com/data-science/building-a-multilingual-multi-agent-chat-application-using-langgraph-i-262d40df6b4f>

- Medium – Data Science (parte I)

Santhosh, R. (2024, 6 de septiembre). Building a multilingual multi-agent chat application using LangGraph — Part I. Medium. Recuperado de <https://medium.com/data-science/building-a-multilingual-multi-agent-chat-application-using-langgraph-i-262d40df6b4f>

- LangChain Docs (LangGraph Overview)

LangChain. (s. f.). LangGraph overview. Docs by LangChain. Recuperado de <https://docs.langchain.com/oss/python/langgraph/overview>

GRACIAS