Universidad de los Andes

Proyecto de Grado - Reunión de Apertura

ARCHIA

ÍNDICE

DESCRIPCIÓN DE LA HERRAMIENTA

“Un asistente de arquitectura de software llamado ArchIA que guía al usuario con el método ADD 3.0: primero define un ASR (Architecture Significant Requirement), luego sugiere estilos arquitectónicos, después tácticas y finalmente diagramas.”

NUESTRA VISIÓN

Latinoamérica se consolida como referente en arquitectura de software gracias a ArchIA, una plataforma creada por LA COMUNIDAD ARQUITECTI para arquitectos.

ArchIA acelera el diseño de sistemas complejos, mejora la calidad de las decisiones arquitectónicas y reduce semanas de trabajo a conversaciones guiadas por inteligencia artificial.

DEMOSTRACIÓN

ESTRUCTURA DEL PROYECTO

Backend

Python + FastAPI

Agent / LLM

LangChain + LangGraph (máquina de estados) con checkpointing persistente en SQLite; integración con OpenAI.

RAG

ChromaDB como vector store; ingestión de documentos (PDFs y texto no estructurado) y recuperación integrada en el grafo de agentes.

Diagramas

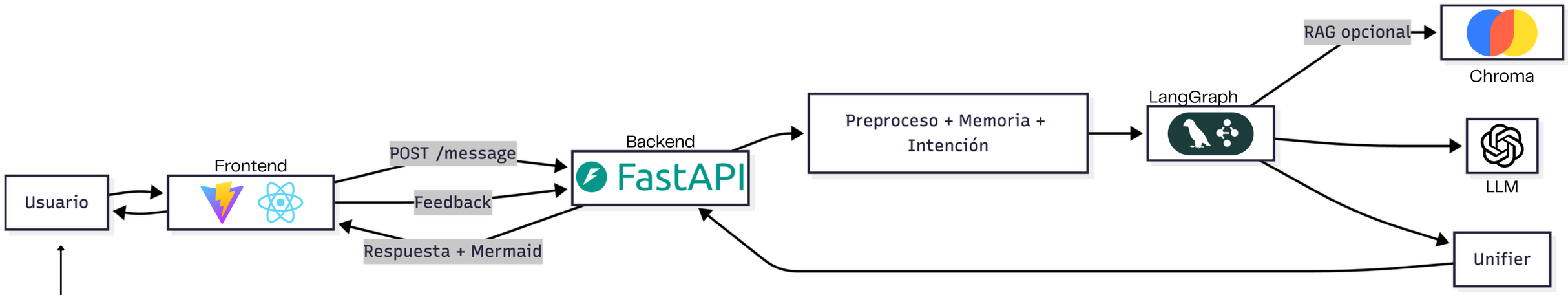
Generación de código Mermaid; posible soporte backend para PlantUML/Kroki

Frontend

React 19 + Vite

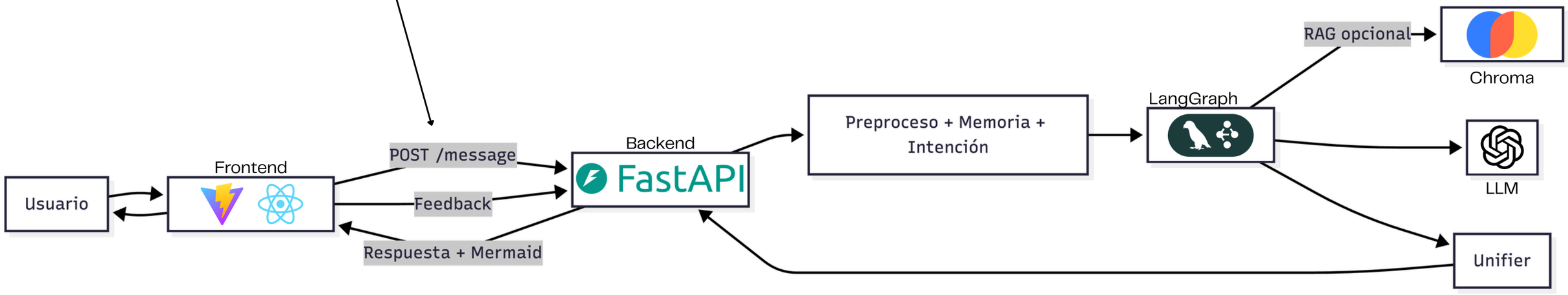
FLUJO ACTUAL

Comportamiento, integraciones, backend y frontend



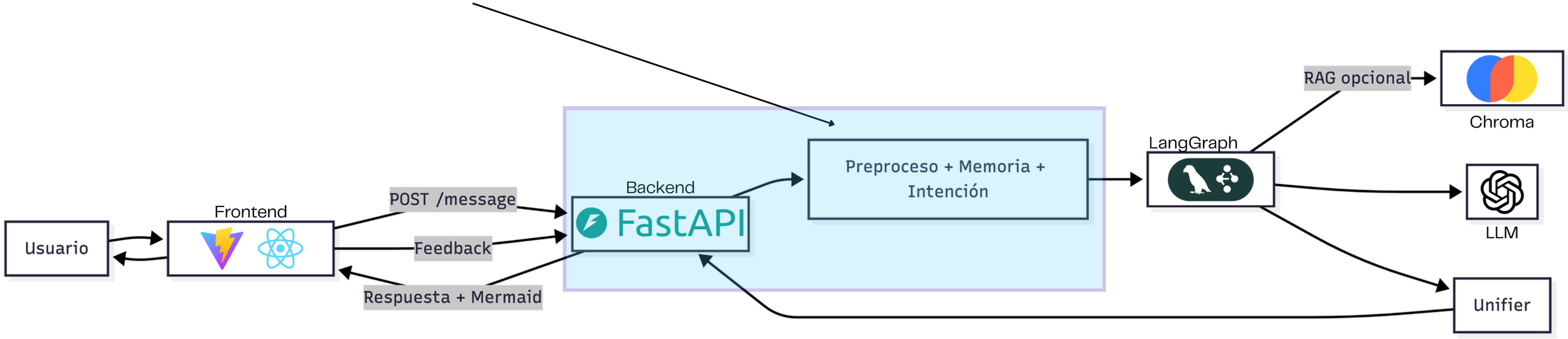
El usuario escribe en el chat y opcionalmente adjunta imágenes o PDFs.

El frontend envía POST /message al backend con el texto, session_id y archivos.



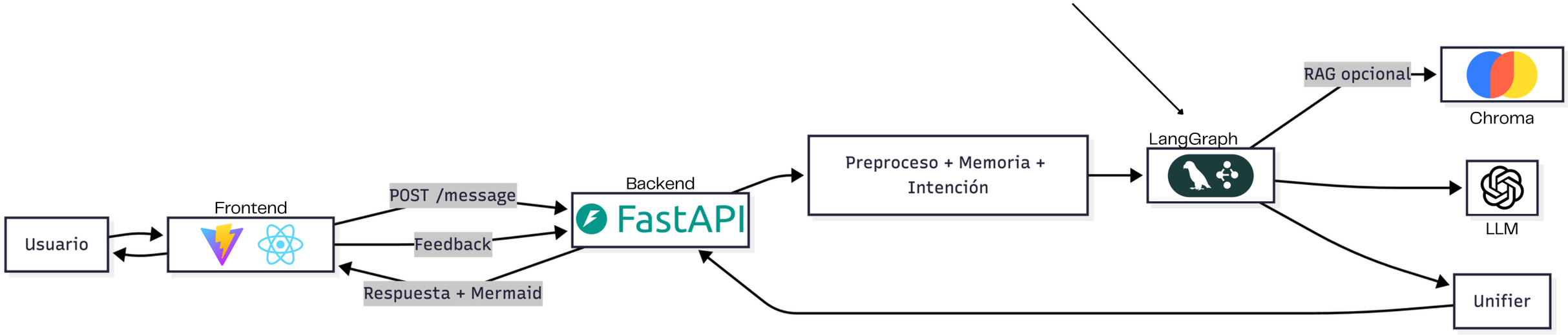
El backend:

- Guarda adjuntos (imágenes o PDFs).
- Si hay PDF, extrae texto y lo usa como contexto.
- Recupera memoria previa (ASR, estilo, tácticas).
- Detecta idioma/intención y configura el flujo.

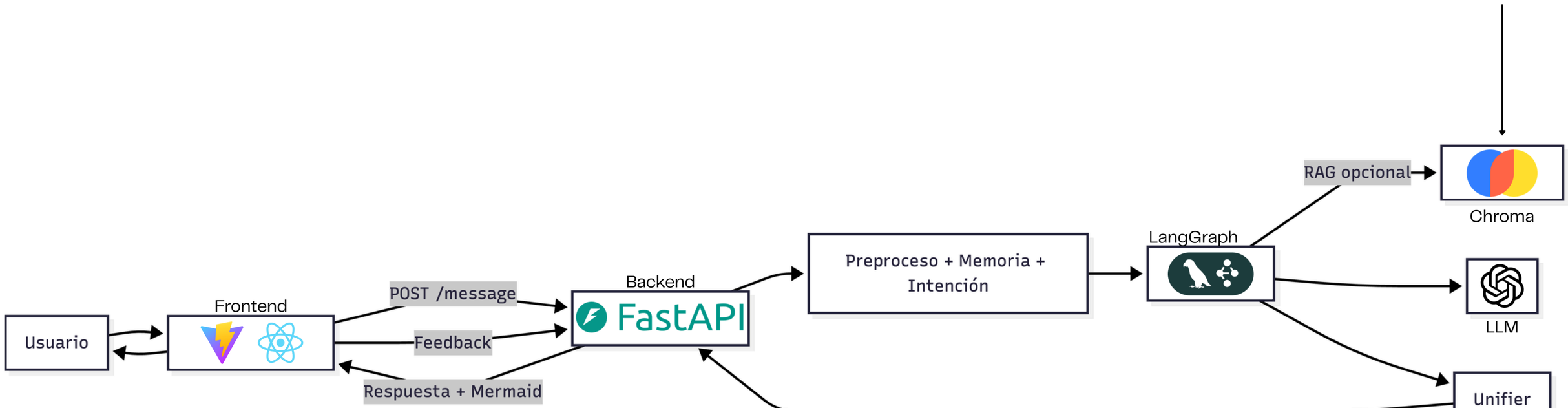


LangGraph decide qué agente ejecutar:

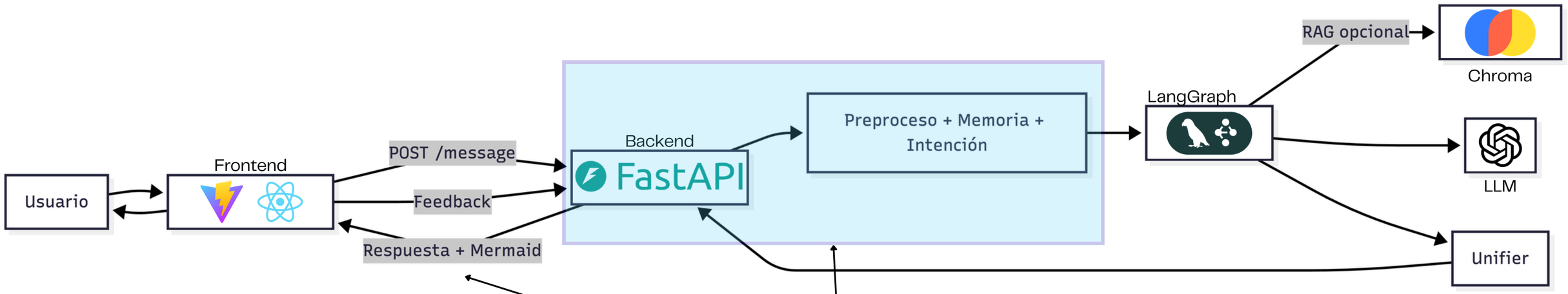
- ASR (si aún no existe),
- Style (si pide estilos),
- Tactics (si pide tácticas),
- Diagram (si pide diagrama),
- Investigator (si hace preguntas técnicas y necesita RAG).



Si corresponde, el agente RAG consulta Chroma con PDFs locales y agrega fuentes.

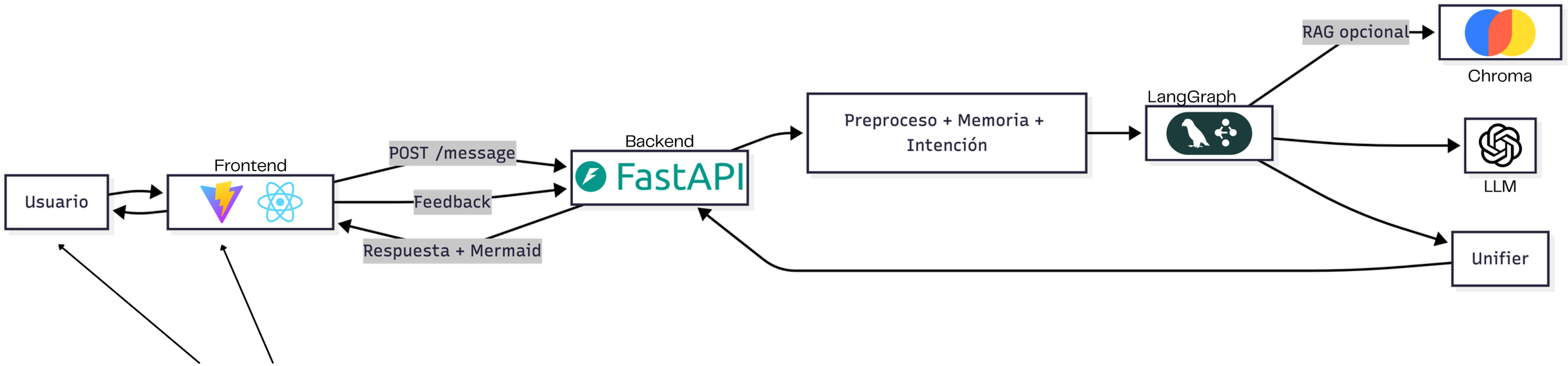


El Unifier compone la respuesta final y sugerencias.



El backend devuelve:

- endMessage (respuesta en texto/Markdown),
- mermaidCode (si hay diagrama),
- suggestions y metadatos.



El frontend renderiza la respuesta y permite dar feedback (/feedback).

GRAFO DE AGENTES

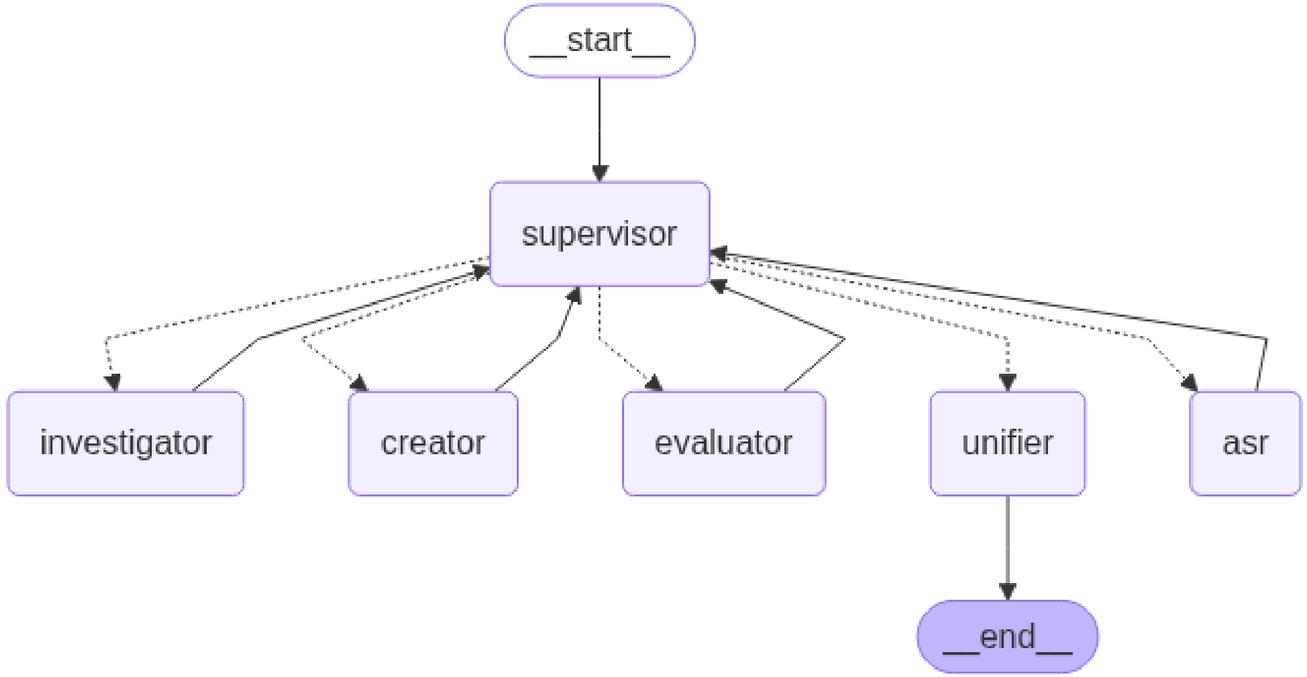
- Arquitectura: LangGraph define una máquina de estados con classifier + supervisor/router que decide qué agente ejecutar.
- Flujo base: START → boot → classifier → supervisor → router → agente especializado → supervisor / unifier → END
- Agentes especializados: investigator, creator, evaluator, diagram_agent, asr, style, tactics, unifier.
- Control de flujo:

-El router aplica reglas de “visitar una vez”.

-Algunos intents pasan primero por investigator (ej. ASR + RAG).

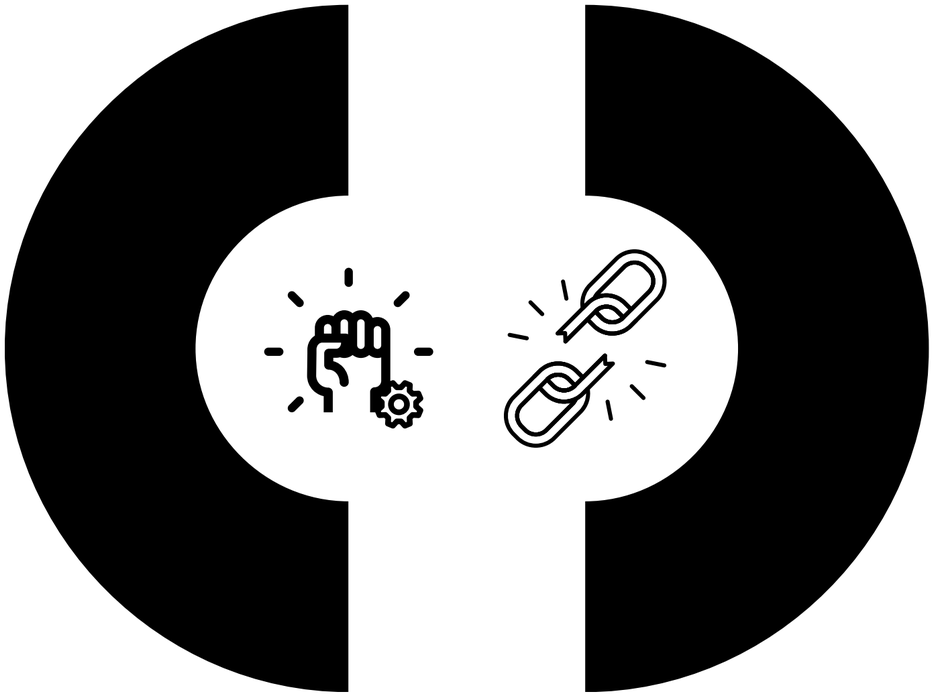
-asr/style/tactics van directo a unifier.

- Persistencia: estado del grafo y conversaciones guardadas con SQLite checkpointing.
- Diagram_agent: solo genera código Mermaid usando el contexto; no renderiza imágenes.



FORTALEZAS

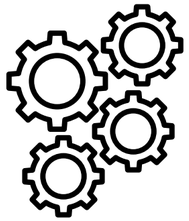
- Bases Solidas
- Agentes Especializados
- Fundación del RAG
- Arquitectura de Langraph fuerte
- Despliegue fácil
- Stack tecnologico coherente



DEBILIDADES

- Alto acoplamiento → Baja Mantenibilidad
- Documentación Deficiente
- Respuestas no optimas y no especificas
- Falta de pruebas
- Lentitud de respuesta

PROPUESTAS DE SOLUCIONES



Separar responsabilidades:

- Separar repositorio de front y de back

Arquitectura Modular para graph.py

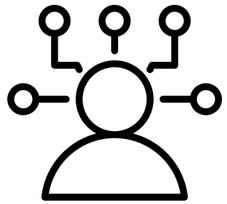
- Estado centralizado: Única fuente de verdad
- Grafo limpio: Solo define conexiones
- Agentes modulares: Un archivo por rol
- Utilidades extraídas: Menos ruido técnico
- Prompts aislados: Configuración, no código

Testing

- Test unitarios: Robustez de los componentes
- Tests E2E: Simulación de usuario real

Calidad del código:

- Hooks: Higiene automática



Calidad de la respuesta

- Implementar agente de reflexión: Auto-corrección
- Hybrid search para el RAG: Precisión semántica y exacta
- Aumentar/mejorar fuente de información del RAG
- Mejorar especificidad al proyecto mediante Prompt-Engineering
- Mejorar manejo del contexto mediante un nuevo nodo, Flow Engineering y tweaks generales

PLAN DE IMPLEMENTACIÓN

Semana 1

- Modularizar nodos
- Extraer utilidades
- Limpiar graph.py

Semana 2

- Aislar prompts
- Generar infraestructura de pruebas
- Primeras pruebas

Semana 3

- Pre commit hooks
- Tests E2E
- Mejora de Prompts
- Extensión del RAG

Semana 4

- Implementación de agente de reflexión
- Implementar hybrid search
- Validación de calidad
- Documentar lo actualizado

REFERENCIAS

LangChain. (n.d.). LangGraph. https://langchain.com/LangGraph

Tiangolo, S. (n.d.). FastAPI. https://fastapi.tiangolo.com/

OpenAI. (n.d.). OpenAI brand guidelines. https://openai.com/brand/

Chroma. (n.d.). Chroma: The AI-native embedding database. https://www.trychroma.com/

LobeHub. (n.d.). LobeHub icons. https://lobehub.com/icons/

Vite. (n.d.). Vite: Next generation frontend tooling. https://vite.dev/

GRACIAS