DuocUC | ESCUELA DE INFORMÁTICA Y TELECOMUNICACIONES

Asistente inteligente para una Universidad

Fernanda Paredes

Giocrisrai Godoy

Ingeniería de soluciones con IA / 003D

Caso organizacional

La organización corresponde a una universidad ficticia, de tamaño medio, dedicada a la educación superior. Atiende a más de 10.000 estudiantes en distintas carreras, tanto diurnas como vespertinas. Su rubro principal es la formación académica profesional y técnica.

Problemática o desafío

Actualmente, los estudiantes enfrentan demoras y dificultades para resolver consultas frecuentes relacionadas con notas, calendarios académicos y becas. La gestión manual mediante secretarías genera sobrecarga administrativa y baja satisfacción estudiantil. El desafío consiste en implementar un asistente inteligente capaz de responder estas consultas de manera rápida, confiable y transparente

Objetivos de la intervención

- Reducir los tiempos de respuesta a consultas estudiantiles.
- Disminuir la carga administrativa de secretarías.
- Mejorar la satisfacción y confianza de los estudiantes.
- Garantizar transparencia en la entrega de información institucional.

Motivación para el uso de IA y agentes inteligentes

En nuestra primera fase del proyecto, el enfoque inicial se basó en el uso de modelos LLM (GPT-4o) y estrategias de prompt engineering para generar respuestas claras y consistentes. En esta nueva versión vamos a evolucionar hacia un agente funcional con memoria y planificación, integrando frameworks de agentes como *LangChain* y *CrewAI*.

Las técnicas empleadas incluyen:

- Zero-Shot y Few-Shot prompting: Para mantener respuestas estandarizadas.
- Chain-of-Thought (CoT): Razonamiento paso a paso ante preguntas complejas.
- Retrieval-Augmented Generation (RAG): búsqueda en documentos académicos (becas, reglamentos) con generación contextual.
- Memoria de corto y largo plazo: Almacenamiento de contexto conversacional y persistencia entre sesiones.

Diseño e implementación del agente

El agente está diseñado bajo una arquitectura modular, compuesta por tres niveles:

1. Cerebro (Core Engine): Impulsado por GPT-4o, gestiona el razonamiento y la toma de decisiones.

2. Memoria (Memory): Implementada mediante almacenamiento temporal de contexto y registro histórico de conversaciones, garantizando coherencia entre consultas prolongadas.

3. Herramientas (Tools): Acceso a APIs, búsqueda documental y orquestación entre subagentes.

Orquestación y componentes

- Agente principal (controller): Este es quien recibe la consulta, activa los subagentes según la intención detectada (consulta, búsqueda, redacción, validación).

- Subagente de búsqueda: Emplea RAG para recuperar información en fuentes institucionales.

- Subagente de redacción: Estructura la respuesta en formato empático y académico.

- Memoria contextual: Usa almacenamiento temporal para recordar consultas previas dentro de la sesión.

Diseño e implementación

Estructura del proyecto:

- assistant_uni.py → flujo principal del agente.

- .env → variables de entorno (OPENAI_API_KEY, BASE_URL).

- memory_module.py → Gestión de contexto conversacional.

- planner_agent.py → Lógica de planificación de tareas.

Dependencias:

openai, langchain, crewai, python-dotenv

El sistema fue probado en entorno virtual (.venv) y ejecutado desde Git Bash, confirmando respuestas adaptativas en tiempo real.

Configuración de memoria y planificación

El agente integra dos niveles de memoria:

- Corto plazo: Conserva el hilo conversacional actual para mantener coherencia inmediata.
- Largo plazo: Registra interacciones anteriores (simuladas mediante logs) que influyen en futuras decisiones.

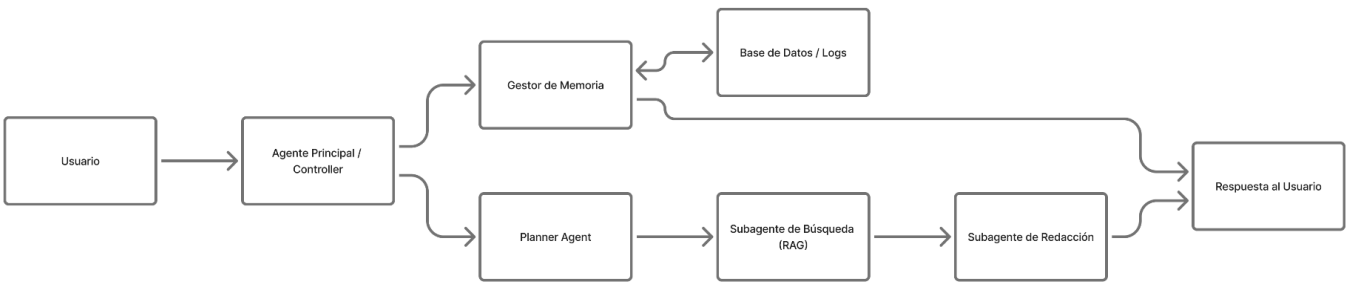
La planificación de tareas se realiza con un enfoque de meta-razonamiento, donde el agente identifica la intención del usuario, selecciona herramientas y ejecuta acciones según prioridad.

Ejemplo de flujo

1. Usuario formula una consulta (“¿Cómo postular a becas?”).
2. El agente planifica la búsqueda en bases internas y genera pasos concretos.
3. El sistema evalúa la respuesta, ajusta el tono y entrega la solución citando fuentes.

Documentación técnica

- Frameworks: OpenAI API, LangChain, CrewAI.
- Memoria: Almacenamiento local simulado y contexto semántico.
- Planificación: Ejecución secuencial mediante tareas definidas (Goal-Oriented Planning).
- Repositorio GitHub: Contiene código fuente, README, y documentación.



Redacción técnica y resultados

Durante la validación, se consultaron preguntas frecuentes como:

“¿Cuándo inician las postulaciones a becas?”

Respuesta generada:

Las postulaciones se realizan entre marzo y abril, con fechas exactas que varían según la institución y el tipo de beneficio. Reúne tu formulario, concentración de notas y antecedentes socioeconómicos, y valida los plazos en la página oficial. (Utiliza esta información desde el archivo [Becas_2026.txt])

Evaluación:
El sistema logró coherencia en la respuesta, mantuvo tono formal y redujo repeticiones gracias al uso de memoria contextual y temperatura baja (0.5). Los resultados evidencian una mejora en consistencia, planificación y trazabilidad.

Conclusión

El Asistente Universitario Inteligente evolucionó de un prototipo basado en prompting a un agente funcional con orquestación, memoria y planificación.
El sistema demuestra autonomía parcial en la toma de decisiones y una estructura modular escalable para entornos educativos. Se proyecta integrar una base de datos persistente y conexión a APIs institucionales para ampliar su capacidad de respuesta.

Referencias

- OpenAI. (2025). *OpenAI API Documentation*. https://platform.openai.com/docs
- LangChain. (2025). *LangChain Framework Documentation*. https://python.langchain.com
- CrewAI. (2025). *Multi-Agent Coordination Library*. https://docs.crewai.io
- Saurabh Kumar. (s. f.). *python-dotenv Documentation*. https://saurabh-kumar.com/python-dotenv
- ChatGPT. (2025). *OpenAI Platform*. https://chat.openai.com