Agentes de Inteligencia Artificial

Nombre del taller: **INGENIERIA DE SOLUCIONES CON INTELIGENCIA ARTIFICIAL 002D**

Nombre de la asignatura: **INGENIERÍA DE SOLUCIONES CON INTELIGENCIA ARTIFICIAL**

Nombre del profesor: **GIOCRISRAI GODOY BONILLO**

Integrantes del grupo (nombre y apellido): MATIAS NICOLAS CERDA REYES
JAVIER ALEXIS MUNOZ TORO

Índice

ANÁLISIS DEL CASO ORGANIZACIONAL

Contexto

Las organizaciones modernas manejan un volumen creciente de información interna: políticas, protocolos, instructivos, normativa legal, procesos operacionales y documentación administrativa.
El acceso a esta información suele ser fragmentado — repositorios, carpetas compartidas, PDFs, correo interno — generando:

- Duplicación de preguntas entre colaboradores.
- Baja productividad del personal administrativo.
- Pérdida de trazabilidad en las consultas internas.
- Retrasos operacionales en respuestas críticas.

El proyecto nace para centralizar, automatizar y estandarizar la entrega de conocimiento institucional mediante un agente conversacional potenciado con LLM + RAG.

Es un asistente capaz de interpretar preguntas naturales, buscar información en documentos internos, recuperar memoria, ejecutar herramientas (APIs, DB, operaciones internas) y generar respuestas estructuradas con acompañamiento de trazabilidad y observabilidad.

Stakeholders

- Estudiantes
- Personal administrativo
- Dirección de TI
- Coordinación académica
- Equipo desarrollador del agente

Requerimientos

- Generación de respuestas consistentes basadas en fuentes verificadas.
- Sistema capaz de actualizar su conocimiento sin reentrenar el modelo.
- Persistencia de memoria para mejorar las interacciones.
- Logs detallados de consultas para auditoría.
- Interfaz API simple para integrarse con portales internos o dashboards.
- Bajos costos de operación y escalabilidad progresiva.

Restricciones

- El sistema debe respetar políticas de privacidad corporativa.
- Los documentos internos pueden contener información sensible, por lo tanto:
    - RAG debe ser local u on-premise.
    - No se puede enviar documentos completos al modelo.
- Infraestructura inicial de bajo costo → se priorizan componentes livianos.
- Tiempo limitado de respuesta: SLA de < 2 segundos para preguntas simples.

PROBLEMA A RESOLVER Y MÉTRICAS DE ÉXITO

Las organizaciones requieren un medio eficiente para democratizar el acceso al conocimiento interno y reducir la sobrecarga operacional. El proyecto busca reemplazar consultas repetitivas y dispersas por un asistente centralizado capaz de proporcionar:

- Respuestas inmediatas.
- Información siempre alineada a los documentos oficiales.
- Interacción natural y en lenguaje cotidiano.
- Mecanismos de control, trazabilidad y aprendizaje continuo.

Métricas de éxito ampliadas

- Precisión RAG
  Objetivo ≥ 85%. Se evalúa comparando chunks relevantes vs chunks recuperados.
- Reducción de carga operacional
  Ahorro esperado ≥ 40% en consultas internas repetitivas.
- Latencia promedio
  Meta ≤ 1.5 segundos por respuesta.
- Índice de satisfacción del usuario
  Encuestas internas ≥ 4/5.
- Tasa de resolución autónoma
  Meta ≥ 70% sin intervención humana.

- Tasa de alucinación controlada
  Meta ≤ 5% de respuestas sin evidencia documental.

Diseño de la solución LLM + RAG

Formulación de prompts

1. ***Prompt Inicial:*** *Este prompt se usa al arrancar el agente. Define el rol, el estilo de respuesta y los límites. Su tono es claro y práctico, orientado a que el modelo entienda cómo debe interactuar con usuarios reales*

"Eres Eva, asistente virtual de la organización. Tu función es ayudar a empleados y estudiantes entregando información confiable, basada en los documentos internos que tengas disponibles. Explica las cosas de forma sencilla, pide aclaraciones cuando algo no sea claro y evita inventar datos. Si el usuario solicita acciones específicas, guíalo paso a paso o utiliza las herramientas internas del sistema cuando corresponda."

2. ***Prompt de seguimiento:*** *Este es el prompt que el agente usa cuando ya existe una conversación en curso. Su objetivo es leer el historial, evitar repeticiones y continuar la interacción sin perder coherencia*

"A continuación tienes el historial reciente de la conversación. Úsalo para mantener el contexto y continuar la respuesta del modo más coherente posible. Evita repetir información ya entregada y revisa si el usuario dejó alguna solicitud pendiente. Si detectas inconsistencias o falta de datos, pide una aclaración antes de avanzar."

3. ***Prompt para activar herramientas:*** *Este prompt sirve cuando el agente debe decidir si debe usar una herramienta (por ejemplo, leer notas, escribir una nueva, consultar conocimiento interno, actualizar un registro, etc.). El estilo está orientado a decisiones claras y acciones seguras*

"Evalúa si la petición del usuario requiere usar una herramienta del sistema.

– Si corresponde, elige la herramienta adecuada y especifica claramente los argumentos.

– Si la información puede responderse con los documentos internos, usa el contenido recuperado por RAG.

– Si no existe suficiente información, indícalo.

No inventes resultados y prioriza siempre acciones seguras y coherentes."

RAG — Pipeline

Flujo completo

1. • Carga de documentos en `storage/docs/*.md`.
2. • Limpieza y preprocesamiento.
3. • Generación de embeddings.
4. • Almacenamiento en **ChromaDB**.
5. • Consulta semántica.
6. • Reranking por similitud.
7. • Envío de chunks al LLM.

Manejo de versiones de documentos

Se recomienda almacenar:

- *Hash MD5.*
- *Fecha de ingestión.*
- *ID de versión.*

Esto permite trazabilidad de respuestas y auditoría completa.

Arquitectura

1. Ingestion Layer

- *Parser Markdown → texto plano.*
- *Chunker.*
- *Generación de embeddings.*
- *Registro en metadata store.*

2. Vector DB

- *Metadatos de documentos.*

3. Retrieval Layer

- *Similarity search.*
- *Filtros por categoría (políticas/procedimientos).*
- *Re-ranking opcional.*

4. LLM Layer

- *Modelo principal.*
- *Motor de prompts.*
- *Control de formato de salida.*

5. Agent Layer

- *Planificador (planning.py).*
- *Memoria conversacional y episódica.*
- *Selector de herramientas.*

6. Observabilidad

- *Dashboard web (`dashboard.py`).*

7. Seguridad

- *Sanitización de entradas.*

- *Cifrado de claves.*

Justificación técnica

Por qué RAG y no fine-tuning

- *Se evita el riesgo de filtrar datos internos.*
- *Actualizar políticas es inmediato.*
- *Mucho más económico.*
- *Respuestas trazables y explicables.*

¿Por qué embeddings?

Permiten identificar similitud semántica entre preguntas y contenido, incluso si el usuario utiliza lenguaje informal.

DESARROLLO DEL AGENTE FUNCIONAL

Integración de herramientas

Ejemplos prácticos:

1. Web Search (DuckDuckGo)

El agente activa esta herramienta cuando la información no está en documentos internos.

2. RAG mediante ChromaDB

Consulta documentos institucionales indexados.

3. LLM vía Ollama

Se usa para generar la respuesta final en todos los modos.

Memoria y recuperación

Buffer memory

- *Conserva la última ventana de conversación (5–10 interacciones).*
- *Ideal para follow-ups.*

Episodic memory

- *Datos importantes y permanentes del usuario.*
- *Guardada en* `storage/notes.json`.

TTL recomendado

Tipo	*TTL*	*Motivo*
Conversacional	*20 minutos*	*Mantener coherencia*

Episódica	*Permanente*	*preferencia o datos estables*
Herramientas	*Según sesión*	*Evitar contaminación*

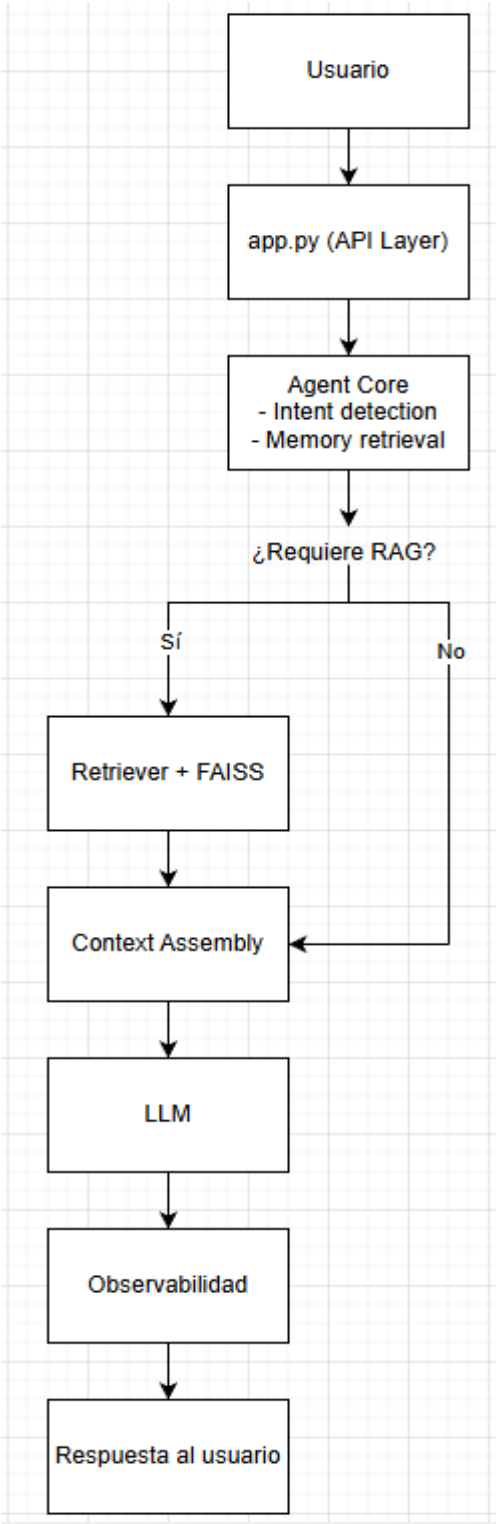
Estrategias de planificación

El agente debe:

1. ***Entender la intención.***
2. ***Clasificar: información → acción → herramienta.***
3. ***Seleccionar si corresponde usar:***
    - *RAG*
    - *Memoria*
    - *Tool*
4. ***Reensamblar contexto.***
5. ***Generar respuesta final con orientación institucional.***

Orquestación

Diagrama textual


Usuario
app.py (API Layer)
Agent Core
- Intent detection
- Memory retrieval
¿Requiere RAG?
Sí
No
Retriever + FAISS
Context Assembly
LLM
Observabilidad
Respuesta al usuario

Métricas

Latencia

Dividida en:

- Preprocesamiento
- Retrieval
- Reranking
- LLM
  Se almacenan timestamps para análisis.

Precisión

Comparación entre:

- Chunks recuperados
- Chunks etiquetados relevantes

Consistencia

Métricas:

- Similitud entre respuestas históricas.
- Detección de variaciones no justificadas.

Logs y análisis

Formato recomendado:

```
timestamp
usuario
consulta_original
embedding_query_id
documentos_recuperados
modelo_utilizado
tiempo_respuesta
resultado
```

Hallazgos:

- *Preguntas ambiguas generan chunks irrelevantes.*
- *La memoria episódica influye en conversaciones largas.*
- *Usuarios intentan pedir información no autorizada — se detectan patrones de seguridad.*

Protocolos de seguridad

- *Sanitizar inputs para evitar ataques como prompt injection.*
- *Clasificar contenido según confidencialidad.*
- *API keys rotadas automáticamente.*
- *Control de longitud del prompt para evitar extracción de información.*

CONCLUSIONES + MEJORAS PROPUESTAS

Conclusiones

IAEva representa una base robusta para construir un asistente institucional, con una arquitectura modular que prioriza:

- *Seguridad*
- *Trazabilidad*
- *Control*
- *Capacidad de expansión*

El sistema demuestra que una organización puede implementar un LLM + RAG sin depender de infraestructuras costosas ni complejas, manteniendo control sobre el conocimiento interno.

Mejoras y propuestas futuras

1. Rendimiento

- *Caching semántico con umbral >0.9.*
- *Precomputación de embeddings.*

2. Memoria

- *Reemplazar JSON con Postgres.*
- *Memoria híbrida basada en embeddings.*

3. Arquitectura

- *Contenedorización con Docker/Kubernetes.*
- *Autoscaling de RAG y LLM.*

4. Seguridad

- *Clasificador de contenido sensible.*

5. Experiencia de usuario

- *Dashboard analítico más completo.*
- *Informe semanal de actividad.*

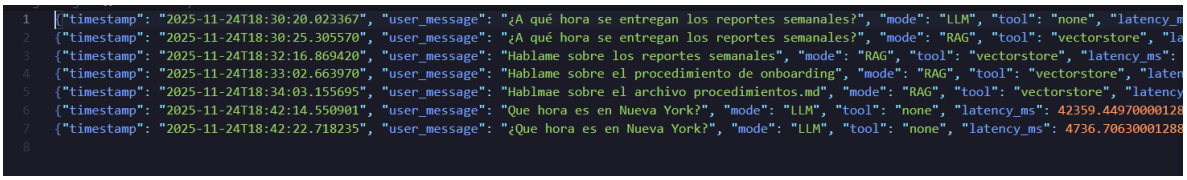
Reflexiones

Matias Cerda R.: *Este proyecto nos mostró que la inteligencia artificial no viene a reemplazar a las personas, sino a potenciar lo que ya hacemos. IAEva no solo responde preguntas; ayuda a liberar tiempo, a reducir la carga operativa y a que cada persona pueda enfocarse en tareas que realmente aportan valor. Entender este equilibrio, donde la tecnología acompaña y no desplaza, es fundamental para adoptarla de manera responsable y consciente.*

Javier Muñoz T.: *Este proyecto no solo nos enseñó sobre RAG, LLMs y observabilidad; también nos obligó a enfrentar problemas reales de ingeniería: tomar decisiones, lidiar con limitaciones, elegir entre distintos caminos y asumir la responsabilidad de lo que construimos. La inteligencia artificial deja de ser solo un concepto teórico cuando la llevamos a un sistema que interactúa con personas reales, y es ahí donde entendemos el verdadero peso de lo que estamos creando.*

ANEXOS

Capturas



Adjuntos

IAEva-main.zip