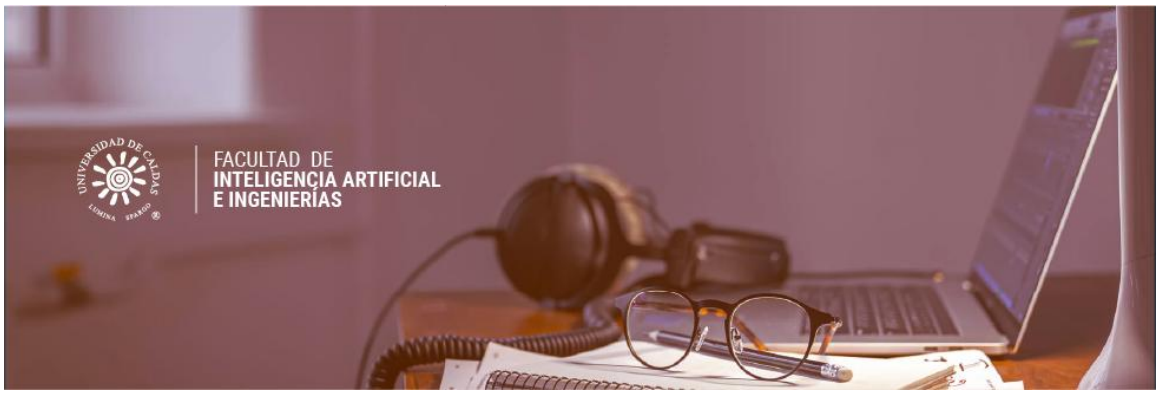
Primer Proyecto– Sistemas Inteligentes I



Programa: Ingeniería de Sistemas y Computación – Universidad de Caldas
Asignatura: Sistemas Inteligentes I
Duración estimada: 4 semanas
Tamaño de grupo: máximo 3 estudiantes
Profesor: Luis Fernando Castillo Ossa
Fecha Entrega 28 Otubre de 2025

Introducción

La comunidad universitaria necesita **un ChatBot confiable sobre Inteligencia Artificial (IA**) que responda preguntas sobre conceptos, historia, aplicaciones, riesgos, marcos regulatorios, ética, información de programas de la Facultad de Inteligencia Artificial e Ingenierías.

El reto del proyecto es diseñar e implementar un prototipo funcional que responda preguntas con **citas verificables**, evitando alucinaciones, y que pueda compararse entre **diferentes LLMs, librerías y arquitecturas de recuperación (RAG)**. Sea transparente: si no sabe, debe indicar que no tiene información suficiente.

El ChatBot debe ofrecer una **interfaz web** (FastAPI, Streamlit, Gradio o similar) y, como **punto adicional**, integrarse con **WhatsApp o Telegram**.

Los equipos deberán además declarar de manera explícita qué **herramientas de IA** usaron para el desarrollo (ChatGPT, Copilot, etc.), en qué etapas, y cómo garantizaron la **validación y confiabilidad de los resultados**.

Pueden usar Herramientas de IA , documentando tiempos , descripciones puntuales y aportes al desarrollo de la aplicación.

Roles del equipo y actividades esperadas

1. Infraestructura y MLOps

Este rol asegura que el proyecto sea **ejecutable, medible y reproducible**.
Actividades esperadas:

- Configurar entornos reproducibles con **Docker** o, si el equipo lo decide, mediante herramientas de orquestación como **n8n** para flujos de integración y pruebas.
- Gestionar dependencias, contenedores y versionado de índices y corpus.
- Implementar **monitoreo y logs anonimizados**: latencia, tokens, consumo de recursos, costo estimado por consulta.
- Diseñar **pipelines de evaluación automática** (script o workflow en n8n) para correr el conjunto de preguntas gold y registrar métricas.
- Asegurar el cumplimiento de prácticas de seguridad básicas: manejo de llaves en .env, uso de HTTPS, evitar almacenamiento de datos sensibles. (En el caso de que no se pueda definir se debe indicar porque si tiene costos asociados)
  Flexibilidad: Los estudiantes podrán elegir entre un stack "código-céntrico" (FastAPI + Docker + LangChain/LlamaIndex/Haystack) o un stack híbrido donde **n8n** se use como orquestador de flujos y monitor de métricas.

---

2. LLM y Evaluación

Este rol es responsable de la **calidad de los modelos y la rigurosidad de las métricas**.
Actividades esperadas:

- Seleccionar y configurar al menos **dos LLMs** (ejemplo: uno open-source como LLaMA 3, Mistral, Falcon; y otro vía API como GPT, Claude o Gemini) Se pueden usar los dos LLM Opensource
- Diseñar el **conjunto gold de preguntas (≥60 Preguntas)**, balanceadas entre: conceptos básicos, historia, ML clásico, deep learning/LLMs, ética/regulación y aplicaciones.
- Implementar métricas de calidad:
  - Exactitud y cobertura (semántica y factual).
  - Claridad (escalas humanas 0–5).
  - Citas válidas y porcentaje de alucinaciones.
  - Latencia y costo por consulta.

- Automatizar la ejecución de benchmarks (scripts en Python o nodos de evaluación en n8n).
- Comparar resultados entre distintos **retrievers y librerías de RAG** (ej. LangChain vs. LlamaIndex vs. Haystack).
  Flexibilidad: Los estudiantes pueden sugerir otras métricas relevantes (ej. robustez, seguridad, diversidad de fuentes) o nuevas librerías de evaluación siempre que lo documenten.

---

3. Contexto y Aplicación

Este rol conecta el proyecto con la **experiencia de usuario y la pertinencia del contenido**.
Actividades esperadas:

- Curar y justificar un corpus mínimo de **10 documentos confiables** (ej. UNESCO, OCDE, AI Act, manuales técnicos, libros introductorios, artículos de IEEE/ACM). Documentos propios de CONPES Colombia, Facultad de IA e ING UdCALDAS.
- Implementar el pipeline de recuperación (RAG): chunking, embeddings, búsqueda en **vector DB** (FAISS, Chroma, Milvus).
- Diseñar la **interfaz de usuario**:
  - Web (FastAPI + HTML/JS, Streamlit o Gradio).
  - Opcional: integración con **WhatsApp o Telegram** (con mensaje de bienvenida, comandos básicos y opt-in/opt-out).
- Establecer **modos de respuesta**: breve (2–3 frases) y extendida (explicación con citas).
- Definir políticas claras de respuesta:
  - Avisar cuando no haya información suficiente.
  - Incluir siempre referencias en el formato requerido.
    Flexibilidad: Los estudiantes pueden experimentar con distintos frameworks y frontends siempre que cumplan con los requisitos de **transparencia y trazabilidad**.

Punto adicional (10 pts)

Los grupos podrán obtener un punto adicional si integran el ChatBot en Telegram o WhatsApp.

Requisitos para el punto adicional (WhatsApp/Telegram)

1. **Mensaje de bienvenida con *opt-in* y política de uso**
    - Cuando un usuario inicia la conversación por primera vez, el bot debe **enviar un saludo inicial** explicando:
        - Qué es el bot (ejemplo: "Soy un asistente sobre Inteligencia Artificial de la Universidad de Caldas").
        - Cómo se recopilarán y usarán los datos (ejemplo: "No almacenamos información personal, solo registramos métricas anónimas para mejorar el servicio").
    - El usuario debe confirmar que acepta interactuar (**opt-in**). Esto puede hacerse con un mensaje como:
        - *"Responde 'ACEPTO' para continuar, o 'SALIR' si no deseas usar el servicio"*.
    - Debe incluir una forma de **opt-out** en cualquier momento, por ejemplo, que si el usuario escribe *"SALIR"* el bot deje de responder.

---

2. **Comandos básicos obligatorios**
   El bot debe reconocer, como mínimo, tres comandos/palabras clave:
    - **Política** → muestra un resumen de la política de uso (explicación breve de opt-in, privacidad, y cómo salir).
    - **Fuente(s)** → devuelve la lista de fuentes confiables que el bot está usando (ejemplo: UNESCO 2023, AI Act 2024, libros introductorios de IA, etc.).
    - **Modo breve/extendido** → el usuario debe poder alternar entre respuestas cortas (2–3 frases) y respuestas extendidas (explicación detallada con citas).

---

3. **Logs anonimizados de interacción (sin PII)**
    - Los estudiantes deben guardar métricas de uso para evaluar al bot, **pero sin almacenar información personal**.
    - En la base de datos o archivo de logs **no debe guardarse el número de teléfono o usuario real**. En su lugar:
        - Generar un **ID aleatorio o hash** que identifique la sesión.

        - Guardar solo lo necesario: texto de la pregunta, tiempo de respuesta, modelo usado, latencia, si hubo cita válida, etc.
    - Esto asegura que los resultados del proyecto se puedan analizar sin comprometer la privacidad de los usuarios.

---

4. **Video de ≤2 minutos mostrando la interacción**
    - Cada grupo debe grabar un video corto (pantalla del celular o del emulador) donde se evidencie:
        - El **mensaje de bienvenida** con opt-in.
        - Una **pregunta real** y la respuesta del bot en **modo breve**.
        - La misma pregunta en **modo extendido**.
        - Uso del comando **fuente(s)** mostrando citas o lista de documentos.
        - Uso del comando **política** mostrando las condiciones.
    - El video servirá como evidencia práctica de que el bot funciona y cumple con los requisitos mínimos.

Transparencia y uso de herramientas de IA

Cada grupo debe declarar explícitamente:
- Qué herramientas de IA usaron en el desarrollo del proyecto (ej. ChatGPT, Copilot, etc.).
- En qué etapas fueron usadas (código, documentación, prompts, depuración, etc.).
- Cómo garantizaron la originalidad y validaron que no hubiera alucinaciones ni errores.

Herramientas sugeridas (no obligatorias)

Cada grupo podrá decidir qué herramientas usar, siempre que documenten su elección.
Entre las opciones:

- **Infraestructura y orquestación:** Docker, n8n, Airflow (básico).
- **Frameworks de RAG:** LangChain, LlamaIndex, Haystack.
- **Bases vectoriales:** FAISS, Chroma, Milvus, Weaviate.etc
- **Interfaces:** FastAPI, Gradio, Streamlit, Telegram Bot API, WhatsApp Cloud API.

- **Monitoreo y métricas:** MLflow, callbacks de LangChain, dashboards simples en n8n o CSV.

Entregables

- - Demo funcional: interfaz web y video de uso.
- - Repositorio reproducible: con README, docker compose up y .env.example.
- - Reporte técnico (máx. 12 páginas) con corpus, arquitectura, protocolo de evaluación, resultados, discusión y recomendaciones.
- - Anexo: Declaración de uso de herramientas de IA.
- - Póster de una página para feria de proyectos.
- - Cuaderno de métricas (CSV o JSON con resultados).

Evaluación del ChatBot

Conjunto gold (≥60 preguntas) en 6 ejes: básicos, historia, ML tradicional, deep learning/LLMs, ética/regulación, aplicaciones.

Fórmula de puntuación por respuesta:
Score_r = 0.35*Exactitud + 0.20*Cobertura + 0.15*Claridad + 0.20*Citas – 0.10*Alucinación – 0.05*Seguridad
Promediado sobre todas las preguntas.

Rúbrica de calificación (100 puntos + 10 extra)

- - Ingeniería y reproducibilidad (20)
- - Diseño de RAG y curación (20)
- - Evaluación rigurosa (20)
- - Comparativa de LLMs/librerías (20)
- - Producto y UX (10)
- - Reporte y presentación (10)
- - Contribución por rol y co-evaluación (10)
- - Integración WhatsApp/Telegram (10 extra)

Criterios de trabajo adecuado

Un proyecto será considerado adecuado si:
- Score global ≥ 0.70.
- Tasa de alucinación ≤ 10%.
- Respuestas con citas ≥ 85%.
- Índice Final de Adecuación (IFA) ≥ 0.75.