Proyecto Final – Introducción a la Inteligencia Artificial

> **Objetivo general:** Aplicar los conceptos básicos aprendidos en el curso para construir un **proyecto práctico de Inteligencia Artificial** que integre los elementos esenciales vistos en clase: manejo de datos, segmentación (chunks), embeddings y similitud, bases de datos vectoriales simples, extracción de datos con OCR o APIs, y una arquitectura de **agentes múltiples construida en LangChain**. El objetivo es que el estudiante **demuestre comprensión y aplicación básica de los componentes fundamentales de un sistema de IA moderno.**

Alcance del proyecto

Este proyecto está diseñado para estudiantes de **Tecnología en Desarrollo de Software.** Por ello, el objetivo no es construir un sistema avanzado, sino **un prototipo funcional, claro y documentado**, que integre los temas principales aprendidos en clase.

Cada equipo (2–3 estudiantes) deberá construir un **asistente o miniaplicación** que realice una tarea de IA con los siguientes componentes:

Componentes mínimos:

1. **Fuente de datos:** puede ser un conjunto de documentos, textos o imágenes (máx. 20 archivos).
2. **Extracción:** leer texto directamente o usar OCR (Tesseract o Google Vision) si hay imágenes o PDFs escaneados.
3. **Segmentación (chunks):** dividir el texto en partes pequeñas.
4. **Embeddings y similitud:** comparar textos usando similitud del coseno o distancia euclidiana.
5. **Base de datos vectorial simple:** guardar y consultar los embeddings (puede usarse FAISS, SQLite + vectores, listas en memoria, o servicios modernos como Pinecone, QDrant, Weaviate o PostgreSQL con PGVector).
6. **Arquitectura multiagente en LangChain:** construir una solución donde varios agentes colaboren (por ejemplo: Agente de extracción, Agente de análisis y Agente de respuesta final). Cada agente debe tener un rol claro dentro del flujo.
7. **Interfaz:** una forma simple de interactuar (puede ser terminal, Gradio, Streamlit o HTML básico desplegado en Replit o Vercel).
8. **Repositorio en GitHub:** con un README claro, paso a paso, y código funcional.

Entregables

1. **Repositorio público en GitHub** con:
2. Código organizado y documentado.
3. `README.md` explicativo (instalación, ejecución, ejemplo de uso).
4. Documento técnico (`docs/Documento_Tecnico.md`).

5. **Demo funcional local o en línea**:
6. Puede ejecutarse localmente (`streamlit run app.py`) o en Replit/Vercel.
7. **Documento Técnico** con:
8. Introducción y problema.
9. Metodología (cómo se construyó el sistema y qué herramientas usaron).
10. Arquitectura (diagrama simple de agentes y flujo de información).
11. Resultados, conclusiones y aprendizajes.

Ejemplos de proyectos posibles

- **Asistente de búsqueda de información:** carga textos o PDFs y permite buscar frases similares con agentes que leen, comparan y responden.
- **Extractor de texto inteligente:** convierte imágenes a texto con un agente de OCR y otro de validación.
- **Mini RAG multiagente:** hace preguntas sobre documentos cargados (agente de recuperación, agente de resumen y agente de generación de respuesta).
- **Comparador de documentos:** agentes que detectan similitudes o diferencias entre textos o contratos.

Sugerencia de estructura del repositorio

```
proyecto-ia-basico/
├─ data/
│  ├─ ejemplos/
│  └─ resultados/
├─ src/
│  ├─ agentes/
│  │  ├─ agente_extraccion.py
│  │  ├─ agente_analisis.py
│  │  └─ agente_respuesta.py
│  ├─ extraccion.py
│  ├─ chunking.py
│  ├─ embeddings.py
│  ├─ similitud.py
│  ├─ app.py  (interfaz Streamlit o CLI)
├─ docs/
│  └─ Documento_Tecnico.md
├─ README.md
└─ requirements.txt
```

Plantilla de Documento Técnico

```
# Documento Técnico – Proyecto Final Introducción a IA

## 1. Introducción
Explicar brevemente el contexto del proyecto y el objetivo general.

## 2. Problema a resolver
Qué se busca solucionar o demostrar.

## 3. Metodología
Describir los pasos principales: extracción, chunking, embeddings,
arquitectura multiagente en LangChain e interfaz.

## 4. Arquitectura de agentes
Explicar cómo se comunican los agentes, su rol y flujo general de
información.

## 5. Resultados y conclusiones
Mostrar ejemplos de uso y aprendizajes del equipo.

## 6. Trabajo futuro
Qué se podría mejorar o ampliar en el futuro.
```

Recomendaciones

- Empezar con datos pequeños y ejemplos sencillos.
- Cada agente debe tener una función clara y documentada.
- Usar **LangChain** para la orquestación entre agentes.
- Documentar cada paso con comentarios en el código.
- Dividir tareas (uno trabaja en extracción, otro en embeddings, otro en agentes, otro en frontend).
- Usar entornos gratuitos: **Replit, Google Colab o Vercel.**

> **Consejo final:** No busques complejidad, busca claridad y funcionalidad. Un sistema multiagente simple pero funcional en LangChain mostrará tu comprensión de la IA moderna aplicada al desarrollo de software.