Prueba Técnica – Coordinador de IA y Automatización

Objetivo general: Evaluar tu habilidad para diseñar y construir una solución de automatización que combine código Python (FastAPI) con orquestación low-code (n8n) e incluya un componente de IA. Debes demostrar buena estructuración, documentación y buenas prácticas de desarrollo.

Ejercicio 1 – Microservicio Python + API

1. **Crear un microservicio** en Python utilizando FastAPI.
2. **Fuentes de datos**:
    - Consumir una API pública (por ejemplo, OpenWeatherMap, NewsAPI o similar).
    - Seleccionar los campos relevantes y guardarlos en una estructura adecuada (pandas DataFrame o diccionario).
3. **Componente de IA**:
    - Implementar una función que utilice un modelo de Machine Learning o una API de IA (puedes usar un modelo preentrenado de scikit-learn, TensorFlow o una API como OpenAI). Por ejemplo: hacer un análisis de sentimiento o clasificar las noticias por categoría.
4. **Interfaz de usuario**:
    - Exponer mediante FastAPI un endpoint REST que, al recibir un parámetro (por ejemplo, nombre de ciudad o categoría de noticias), devuelva la información procesada y el resultado del análisis de IA en formato JSON.

Entrega esperada: Código Python bien estructurado, requisitos (requirements.txt), documentación en README y capturas o notas que demuestren la ejecución local.

Ejercicio 2 – Diseño de flujo de automatización low-code

1. **Seleccionar una herramienta low-code**: n8n (preferida) u otra similar (Make, Zapier).
2. **Objetivo del flujo**:

    - Automatizar la ingesta diaria de datos generados por el microservicio del **Ejercicio 1**.
    - Guardar el resultado en un archivo CSV o en Google Sheets y notificar por email o Slack cuando se genere un registro.
3. **Documentación**:
    - Describir cada paso del flujo: nodos utilizados, configuración de conexiones, programaciones.
    - Incluir al menos un control de errores (por ejemplo, reintentar ante fallos de API).

Entrega esperada:

- Capturas de pantalla o export del flujo (si la herramienta permite exportar JSON).
- Un README explicando cómo desplegar o reproducir el flujo paso a paso.

Ejercicio 3 – Diseño de asistente basado en RAG

1. **Concepto**: Proponer un asistente interno de IA para consultas sobre documentación técnica de la empresa, aplicando el patrón *Retrieval-Augmented Generation* (RAG).
2. **Entrega**:
    - Documento (puede ser un Markdown o PDF) en el que expliques la arquitectura: fuentes de datos, pipeline de ingesta, vector store (por ejemplo, Pinecone, Milvus), cómo el usuario interactúa con la IA y cómo se controla la seguridad y la relevancia de las respuestas.
    - Detallar las ventajas y posibles limitaciones del enfoque RAG.
    - Incluir un diagrama (puedes usar PlantUML, draw.io o similar) que ilustre el flujo de datos.

Entrega esperada:

- Documento con texto y diagrama, demostrando comprensión de RAG, arquitectura propuesta y justificaciones técnicas.

Formato y plazos

- **Idioma de entrega**: Español.
- **Plazo sugerido**: 5 días hábiles (adaptable según agenda).
- **Envío**:
    - Repositorio de Git con los códigos, flujos y documentación.
    - El README general debe explicar cómo ejecutar cada ejercicio y las dependencias utilizadas.

Criterios de evaluación

1. **Calidad de código y buenas prácticas**: Limpieza, modularidad, uso de entornos virtuales, claridad en comentarios y documentación.
2. **Habilidad con nuevas herramientas**: Capacidad para aprender y aplicar n8n u otra herramienta low-code de forma efectiva.
3. **Comprensión de arquitectura de IA**: Coherencia en la propuesta RAG, identificación de componentes y justificación de elecciones.
4. **Creatividad y claridad**: Forma de presentar soluciones, diagramas y argumentos.
5. **Cumplimiento de plazos**.

Esta prueba busca evaluar tanto la capacidad técnica como la estructuración y claridad al presentar soluciones de automatización e IA. ¡Éxitos en la resolución y esperamos tu entrega!