Recruitment Challenge

SSR BACKEND DEV

Querido Dev,

Felicitaciones por haber llegado a esta instancia del proceso de selección. Creemos que tu perfil es compatible con nuestras necesidades y nos encantaría poder trabajar juntos.

En este documento se detallan los requerimientos para para concluir con la última etapa del proceso. Un pequeño code challenge para tener una referencia real de tus habilidades.

El Desafío

Deberás desarrollar una API REST que reciba la URL de un video de YouTube (o un archivo de video) y realice un proceso de extracción, análisis y estructuración de información mediante un flujo de agentes.

Requisitos Funcionales

El sistema debe contar con un endpoint principal que dispare un grafo de **LangGraph** con los siguientes nodos:

1. **Nodo de Extracción:** Obtener la transcripción del video (podés utilizar librerías como youtube-transcript-api, Whisper o servicios similares).
2. **Nodo de Análisis de Sentimiento y Tono:** Procesar el texto para determinar el sentimiento general (positivo, negativo, neutral) y el tono del orador (formal, informal, técnico, sarcástico, etc.).
3. **Nodo de Estructuración (JSON Output):** Generar un resumen de los 3 puntos clave del video y formatear todo el análisis en un esquema de datos específico.

```
{
  "video_metadata": {
    "title": "string",
    "duration_seconds": "integer",
    "language_code": "string (ISO 639-1)"
  },
```

```
  "analysis": {
    "sentiment": "string (enum: 'positivo' | 'negativo' | 'neutral')",
    "sentiment_score": "float (rango 0.0 a 1.0)",
    "tone": "string",
    "key_points": "list[string]"
  }
}
```

4. **Persistencia:** Los resultados finales (URL, transcripción, sentimiento, tono y resumen) deben almacenarse en una base de datos **PostgreSQL** utilizando el ORM de Django.

Requisitos Técnicos

- **Framework:** Django + Django REST Framework (DRF).
- **Orquestación AI:** LangGraph (excluyente).
- **Base de Datos:** PostgreSQL.

Entregables

1. **Repositorio en GitHub:** Con un historial de commits que refleje el proceso de desarrollo.
2. **Archivo README.md:** Con instrucciones claras para correr el proyecto localmente.
3. **Dockerization (opcional):** Incluir un docker-compose.yml que levante la aplicación, la base de datos y cualquier dependencia necesaria.

Criterios de Evaluación

- **Arquitectura:** Cómo organizás el estado del grafo en LangGraph.
- **Manejo de Errores:** Cómo reacciona el sistema si el video no tiene audio o si falla la API del LLM.
- **Clean Code:** Legibilidad, modularidad y seguimiento de buenas prácticas (PEP 8).
- **SQL:** Diseño del modelo de datos en Django.

Nota: *Podés utilizar el modelo de lenguaje de tu preferencia (OpenAI, Anthropic, Groq o modelos locales vía Ollama).*

Plazo de entrega

El desafío está diseñado para realizarse en un lapso de máximo **48 horas**.

¡Muchos éxitos! Estamos ansiosos por ver tu solución.