README - Talent Gap Analyzer

1. Descripción

El Talent Gap Analyzer es una aplicación web interactiva que transforma el análisis de la brecha de talento de días a un diagnóstico instantáneo. Compara la plantilla actual (`talento_actual.csv`) con la visión futura de la empresa (`vision_futura.json`) para identificar riesgos y oportunidades. Utiliza un algoritmo de ponderación e IA Generativa (LangChain) para generar planes de desarrollo personalizados.

2. Características Principales

- **Análisis Táctico de Roles:** Permite a los mánagers seleccionar un rol futuro y ver rankings de candidatos (internos y externos) clasificados por compatibilidad (`READY`, `NEAR`, `FAR`, etc.).
- **Planes de IA:** Genera planes de desarrollo personalizados para cada candidato y resúmenes estratégicos para la dirección.
- **Explorador de Red:** Muestra un grafo interactivo (Pyvis) que conecta empleados con sus habilidades (el grosor de la línea representa el nivel de experiencia 1-10).
- **Captura de Talento:** Un formulario dinámico en la barra lateral para añadir nuevos perfiles (internos o externos) con un selector de nivel de habilidad (1-10).

3. Arquitectura de Archivos y Stack

El proyecto está organizado en módulos, archivos de datos y documentación:

Archivos de Código (en `modules/`):

- `modules/data_loader.py`: Se encarga de leer, limpiar y unir todos los archivos de datos (los dos CSVs y los JSONs).
- `modules/compatibility.py`: Contiene el motor de cálculo (algoritmo de ponderación del Nivel 2).
- `modules/recommendations.py`: Gestiona la conexión con la API de IA (LangChain + Groq).

Archivos de Datos (en `data/`):

- `talento_actual.csv`: La base de datos de empleados internos (trabajadores actuales de la empresa).
- `candidatos_externos.csv`: La base de datos del "Talent Pool" externo.
- `org_config.json`: Define la lista de Roles y Habilidades que hay en la organización.

- `vision_futura.json`: Define los roles futuros que la empresa necesita.

Archivo de Informe (en `informe/`):

- `Informe: Talent Gap Analyzer`: El informe explicativo completo del proyecto, sus funcionalidades y conclusiones.

Archivos Raíz:

- `app.py`: Es el archivo principal que se ejecuta. Dibuja toda la interfaz de usuario (UI) con Streamlit, organiza las secciones y llama a los otros módulos (en los que se almacenan las diferentes funciones para el proyecto).
- `requirements.txt`: La lista de todas las dependencias de Python necesarias para ejecutar el proyecto.
- `.env`: Archivo de configuración local para guardar la clave secreta de la API de IA usada para generar los informes.

Stack Tecnológico Principal (Nombrados en `requirements.txt` también):

- Python, Streamlit, Pandas, Scikit-learn, Plotly, Pyvis, NetworkX, LangChain.

4. Guía de Ejecución

Siga estos 3 pasos para lanzar la aplicación web:

Paso 1: Configurar el Entorno e Instalar Dependencias Desde la terminal, en la carpeta raíz del proyecto (`Hackaton`), ejecute:

 # 1. Activar el entorno virtual

.\.venv\Scripts\activate

 # 2. Instalar las librerías base del reto

pip install -r requirements.txt

 # 3. Instalar las librerías adicionales para IA y Grafos

pip install langchain-groq networkx pyvis

Paso 2: Configurar la Clave de API Para que las funciones de IA (Nivel 3) funcionen, cree un archivo llamado `.env` en la carpeta raíz. Para esta, hace falta añadir una clave de API de Groq dentro de este archivo (Desde la página oficial de Groq, crear una API Key de forma gratuita):

GROQ_API_KEY="su_clave_secreta_aqui"

Paso 3: Ejecutar la Aplicación Una vez configurado el entorno, lance la aplicación:

streamlit run app.py

La aplicación se abrirá automáticamente en su navegador web.