**1. Configuración considerada más justa**

La configuración actual de pesos entre el json y la bd es:

* **first\_name**: 5
* **last\_name**: 7
* **email**: 12

Esta configuración otorga el mayor peso al campo **email**, lo cual es adecuado porque el correo electrónico suele ser un identificador único y menos propenso a errores de escritura que los nombres. Los campos de nombre y apellido tienen pesos menores, pero razonables, permitiendo que pequeñas diferencias en estos campos no descarten un posible match si el email coincide.

**Conclusión:**  
La configuración es justa y lógica para la mayoría de los escenarios de matching de usuarios, priorizando la unicidad y confiabilidad del email.

**2. Posibles problemas detectados**

* **Gestión de pesos:**  
  Si los pesos en la base de datos y en el archivo difieren, se toma el más reciente, pero no hay una advertencia clara al usuario sobre el conflicto ni una opción para resolverlo manualmente.
* **Validación de pesos:**  
  El rango de pesos está limitado entre 1 y 50, pero no se valida que la suma total tenga sentido (por ejemplo, que no sea demasiado baja o alta para todos los campos).
* **Dependencia de nombres de columnas:**  
  El sistema depende de que los nombres de columnas en los mapeos y en los pesos coincidan exactamente. Si hay un error de mapeo, el matching puede fallar silenciosamente.
* **Falta de logs o auditoría en la BD:**  
  Aunque hay auditoría en JSON, no se auditan los cambios de pesos hechos directamente en la base de datos desde la aplicación. El admin tiene que entrar a la terminal de maríaDB y hacer uso del SP para que la aplicación detecte cambios y los guarde en los logs.

**3. Recomendaciones de mejora**

**Beneficios:**

* El sistema es flexible y permite ajustar pesos fácilmente.
* El uso de RapidFuzz y pesos ponderados mejora la precisión del matching.
* La auditoría en JSON permite rastrear cambios locales.

**Limitaciones y mejoras sugeridas:**

* **Unificación de auditoría:**  
  Centralizar la auditoría de cambios de pesos, tanto en archivo como en base de datos, para evitar inconsistencias y facilitar el seguimiento.
* **Validación de configuración:**  
  Añadir una validación que advierta si la suma de los pesos es demasiado baja o si algún campo esencial tiene peso cero.
* **Mejor manejo de conflictos:**  
  Informar al usuario cuando hay diferencias entre los pesos en archivo y en base de datos, y permitirle elegir cuál usar.
* **Documentación y mensajes claros:**  
  Mejorar los mensajes al usuario para explicar el impacto de los cambios de pesos y los resultados del matching.

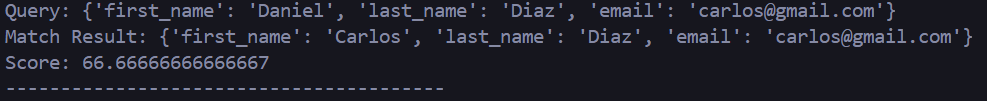
**Resumen:**  
La configuración actual es adecuada, pero el sistema puede beneficiarse de una mejor gestión de conflictos, validaciones adicionales y una auditoría más robusta. Esto aumentará la confiabilidad y facilidad de uso del proceso de matching.

Una captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

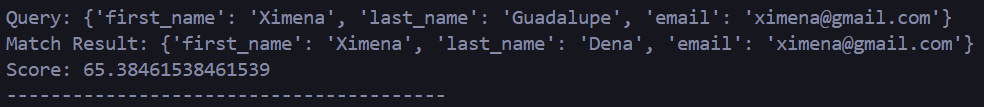
En esta foto podemos ver que pasa cuando el email tiene mas valor, ya que al ser un identificador debe tener mas peso. Todo coincide menos el Email.





En esta en first\_name tiene más peso que las demás columnas y solamente el first\_name no coincide.





Y aquí en esta ultima foto el peso dominante es de last\_name. Donde el score nos da 65.

Texto

El contenido generado por IA puede ser incorrecto.

Aquí esta pruebas de todos los registros con los scores con los pesos justos que mencioné al inicio. Por qué da menos score en last\_name que en email? Porque el algoritmo compara que tanta similitud tiene los campos que está comparando. Se puede apreciar que en el ultimo registro compara Guadalupe con Dena, el cual no tiene NADA de similitud. Y en donde compara los correos, tienen algo de similitud, el test, el @, el .com, el Gmail.