# main.py

from fastapi import FastAPI

import pandas as pd

from pydantic import BaseModel

import numpy as np

# --- 1. Configuración de la API para Auditoría y Consulta ---

app = FastAPI(

title="API de Auditoría y Consulta para CODEX JARVIS",

description="""

Endpoint robusto diseñado para la \*\*extracción exhaustiva de datos\*\*.

Recibe una consulta clínica, la procesa contra la Matriz de Datos Clínicos y,

utilizando una \*\*vinculación de doble clave (ITEM ID + Código ADN)\*\*, extrae

y consolida la totalidad de los datos de la Matriz de Medidas Funcionales.

Ideal para auditorías, consultas y flujos de trabajo en n8n.

""",

version="4.0 - Auditor"

)

# --- 2. Carga Segura y Consistente de las Bases de Datos ---

# Se definen los tipos de datos para las columnas clave para garantizar una

# coincidencia perfecta y evitar errores de tipo (ej. 123 vs "123").

column\_types = {

'ITEM ID DE LA MATRIZ': str,

'Código ADN Clínico-Funcional': str

}

try:

# Carga la matriz de datos clínicos, asegurando la consistencia de tipos.

df\_clinico = pd.read\_csv(

"CODEX JARVIS - MATRIZ DE DATOS CLÍNICO - FUNCIONALES.xlsx - Hoja1.csv",

dtype=column\_types

)

df\_clinico.columns = df\_clinico.columns.str.strip()

print("✅ Base de Datos Clínica cargada y lista para auditoría.")

# Carga la matriz de medidas funcionales, asegurando la consistencia de tipos.

df\_medidas = pd.read\_csv(

"CODEX JARVIS - MATRIZ DE MEDIDAS FUNCIONALES.xlsx - Hoja1.csv",

dtype=column\_types

)

df\_medidas.columns = df\_medidas.columns.str.strip()

print("✅ Base de Datos de Medidas Funcionales cargada y lista para auditoría.")

except FileNotFoundError as e:

print(f"❌ ERROR CRÍTICO AL INICIAR: Archivo no encontrado: {e.filename}")

df\_clinico = None

df\_medidas = None

except Exception as e:

print(f"❌ ERROR INESPERADO AL CARGAR ARCHIVOS: {e}")

df\_clinico = None

df\_medidas = None

# Define la estructura de la consulta que la API recibirá

class Consulta(BaseModel):

texto\_de\_la\_consulta: str

# --- 3. Lógica de Inferencia y Extracción Exhaustiva ---

def auditar\_consulta(query: str):

"""

Motor principal de la API. Ejecuta el protocolo de auditoría:

1. Busca en la matriz clínica (Columnas K y L) para encontrar el registro único.

2. Extrae las dos llaves maestras: ITEM ID y Código ADN.

3. Usa AMBAS llaves para una búsqueda de alta precisión en la matriz de medidas.

4. Si la vinculación es exitosa, extrae y devuelve \*\*TODOS los campos\*\* de ambas filas.

"""

if df\_clinico is None or df\_medidas is None:

return {"error": "Bases de datos no disponibles. Contacte al administrador."}

# Búsqueda inicial en las columnas primarias del núcleo clínico

col\_k = "CONDICIÓN CLÍNICA MÉDICA"

col\_l = "HALLAZGO ESPERADO"

mask = (df\_clinico[col\_k].str.contains(query, case=False, na=False)) | \

(df\_clinico[col\_l].str.contains(query, case=False, na=False))

resultados\_df = df\_clinico[mask]

# --- Evaluación de Resultados ---

if len(resultados\_df) == 0:

return {"error": f"Auditoría fallida. Cero registros encontrados para la consulta: '{query}'."}

if len(resultados\_df) > 1:

return {"error": f"Ambigüedad detectada. La consulta arrojó {len(resultados\_df)} resultados. Se requiere mayor especificidad para una auditoría precisa."}

# --- Éxito: Registro Único Localizado. Procediendo a la extracción completa ---

if len(resultados\_df) == 1:

print(f"🎯 Registro único localizado para '{query}'. Iniciando extracción de datos completa...")

# Convierte la fila clínica a un diccionario, manejando valores nulos.

resultado\_clinico\_series = resultados\_df.iloc[0].replace({np.nan: None})

datos\_clinicos\_completos = resultado\_clinico\_series.to\_dict()

# Extrae las dos llaves maestras para la vinculación

item\_id = datos\_clinicos\_completos.get('ITEM ID DE LA MATRIZ')

adn\_code = datos\_clinicos\_completos.get('Código ADN Clínico-Funcional')

if not item\_id or not adn\_code:

return {"error": "Fallo de integridad de datos. El registro clínico localizado carece de ITEM ID o Código ADN."}

print(f"🧬 Vinculando con Doble Llave Maestra -> ID: {item\_id}, ADN: {adn\_code}")

# --- Búsqueda de alta precisión en Matriz de Medidas con la doble llave ---

mask\_medidas = (df\_medidas['ITEM ID DE LA MATRIZ'] == item\_id) & \

(df\_medidas['Código ADN Clínico-Funcional'] == adn\_code)

medidas\_result\_df = df\_medidas[mask\_medidas]

if len(medidas\_result\_df) == 0:

return {

"advertencia": "Extracción parcial. Se encontró el registro clínico pero su par ID/ADN no existe en la Matriz de Medidas.",

"datos\_clinicos\_completos": datos\_clinicos\_completos,

"medidas\_funcionales\_completas": None

}

# Convierte la fila de medidas a un diccionario, manejando valores nulos.

resultado\_medidas\_series = medidas\_result\_df.iloc[0].replace({np.nan: None})

medidas\_funcionales\_completas = resultado\_medidas\_series.to\_dict()

print("✅ ¡Extracción completa y vinculación exitosa!")

# --- Ensamblaje del Informe Final de Auditoría ---

return {

"estado\_auditoria": "Exitosa",

"consulta\_realizada": query,

"datos\_clinicos\_completos": datos\_clinicos\_completos,

"medidas\_funcionales\_completas": medidas\_funcionales\_completas

}

return {"error": "Error desconocido durante el proceso de auditoría."}

# --- 4. El Endpoint de la API: La Puerta de Entrada para n8n ---

@app.post("/auditar")

def procesar\_auditoria(consulta: Consulta):

"""

Recibe una consulta textual desde un cliente (n8n, etc.),

la procesa con el motor de auditoría de SAMI y devuelve

el conjunto de datos completo y consolidado.

"""

query\_texto = consulta.texto\_de\_la\_consulta

resultado\_final = auditar\_consulta(query\_texto)

return resultado\_final