Generación de Audio para Conversaciones

Solicitud:

```
Haz al menos 100 rondas de conversación cubriendo todos los detalles sobre este PDF, dame el formato JSON sobr
```

```
[
    {
      "speaker": "A",
      "line": "Oye, he estado escuchando mucho sobre Machine Learning (ML), Deep Learning (DL) y GPT últimamer
    },
    {
      "speaker": "B",
      "line": "¡Claro! Comencemos con lo básico. Machine Learning es un campo de la informática donde los sist
    }
]
```

Código:

```
import os
import json
import random
import subprocess
from google.cloud import texttospeech
import tempfile
import time
import argparse

# Directorio de salida fijo para las conversaciones
OUTPUT_DIRECTORY = "assets/conversations"
INPUT_DIRECTORY = "scripts/conversation"

def text_to_speech(text, output_filename, voice_name=None):
    print(f"Generando audio para: {output_filename}")
    try:
        client = texttospeech.TextToSpeechClient()
        synthesis_input = texttospeech.SynthesisInput(text=text)
        if not voice_name:
            voice_name = random.choice(["en-US-Journey-D", "en-US-Journey-F", "en-US-Journey-O"])
        voice = texttospeech.VoiceSelectionParams(language_code="en-US", name=voice_name)
```

```
        audio_config = texttospeech.AudioConfig(
            audio_encoding=texttospeech.AudioEncoding.MP3,
            effects_profile_id=["small-bluetooth-speaker-class-device"]
        )

        reintentos = 5
        for intento in range(1, reintentos + 1):
            try:
                response = client.synthesize_speech(input=synthesis_input, voice=voice, audio_config=audio_co
                with open(output_filename, 'wb') as out:
                    out.write(response.audio_content)
                print(f"Contenido de audio escrito en {output_filename}")
                return True
            except Exception as e:
                print(f"Error en el intento {intento}: {e}")
                if intento == reintentos:
                    print(f"Fallo al generar audio después de {reintentos} intentos.")
                    return False
                tiempo_espera = 2 ** intento
                print(f"Reintentando en {tiempo_espera} segundos...")
                time.sleep(tiempo_espera)
    except Exception as e:
        print(f"Ocurrió un error al generar audio para {output_filename}: {e}")
        return False

def process_conversation(filename):
    filepath = os.path.join(INPUT_DIRECTORY, filename)
    output_filename = os.path.join(OUTPUT_DIRECTORY, os.path.splitext(filename)[0] + ".mp3")

    if os.path.exists(output_filename):
        print(f"El archivo de audio ya existe: {output_filename}")
        return

    try:
        with open(filepath, 'r', encoding='utf-8') as f:
            conversation = json.load(f)
    except Exception as e:
        print(f"Error al cargar el archivo de conversación {filename}: {e}")
        return
```

```
archivos_temporales = []

voice_name_A = random.choice(["en-US-Wavenet-D", "en-US-Wavenet-E", "en-US-Wavenet-F"])
voice_name_B = random.choice(["en-US-Studio-O", "en-US-Studio-M", "en-US-Studio-Q"])

for idx, line_data in enumerate(conversation):
    speaker = line_data.get("speaker")
    line = line_data.get("line")
    if not line:
        continue
    archivo_temporal = os.path.join(OUTPUT_DIRECTORY, f"temp_{idx}.mp3")
    archivos_temporales.append(archivo_temporal)

    voice_name = None
    if speaker == "A":
        voice_name = voice_name_A
    elif speaker == "B":
        voice_name = voice_name_B

    if not text_to_speech(line, archivo_temporal, voice_name=voice_name):
        print(f"Fallo al generar audio para la línea {idx+1} de {filename}")
        # Limpiar archivos temporales
        for archivo_temporal_a_eliminar in archivos_temporales:
            if os.path.exists(archivo_temporal_a_eliminar):
                os.remove(archivo_temporal_a_eliminar)
        return

if not archivos_temporales:
    print(f"No se generó audio para {filename}")
    return

# Concatenar usando ffmpeg
archivo_concat = os.path.join(OUTPUT_DIRECTORY, "concat.txt")
with open(archivo_concat, 'w') as f:
    for archivo_temporal in archivos_temporales:
        f.write(f"file '{os.path.abspath(archivo_temporal)}'\n")

try:
    subprocess.run(
        ['ffmpeg', '-f', 'concat', '-safe', '0', '-i', archivo_concat, '-c', 'copy', output_filename],
```

```
            check=True,
            capture_output=True
        )
        print(f"Audio concatenado exitosamente en {output_filename}")
    except subprocess.CalledProcessError as e:
        print(f"Error al concatenar audio: {e.stderr.decode()}")
    finally:
        os.remove(archivo_concat)
        for archivo_temporal in archivos_temporales:
            os.remove(archivo_temporal)


if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Procesa archivos JSON de conversación para generar audio.")
    args = parser.parse_args()

    os.makedirs(OUTPUT_DIRECTORY, exist_ok=True)

    for filename in os.listdir(INPUT_DIRECTORY):
        if filename.endswith(".json"):
            process_conversation(filename)
```