**Bitácora Detallada: Implementación de procesamiento de transcripciones con ChatGPT**

**Contexto Inicial**

Partiendo de la bitácora inicial proporcionada, la tarea consistía en implementar un sistema de procesamiento de transcripciones de audio con ChatGPT al sistema DocuBox. Este desarrollo incluía la capacidad de procesar archivos MP3, transcribirlos a texto, y opcionalmente mejorar esa transcripción utilizando la API de OpenAI (ChatGPT).

**Trabajo Realizado Durante la Sesión**

**1. Análisis de la Situación Actual**

* Se evaluó el estado del sistema DocuBox, que ya contaba con capacidad de transcripción de audio MP3 a texto.
* Se identificó que el sistema necesitaba ampliarse para permitir el procesamiento adicional de las transcripciones mediante ChatGPT.
* Se revisaron los siguientes archivos clave:
  + E:\Admin7\master\backend\server.js: Servidor principal de Node.js
  + E:\Admin7\master\backend\.env: Archivo de variables de entorno
  + E:\Admin7\master\frontend\src\components\FileList.js: Componente de interfaz de usuario
  + E:\Admin7\master\frontend\src\services\api.js: Servicios de API para el frontend

**2. Modificaciones Realizadas**

**2.1. Configuración de Variables de Entorno**

* Se agregó la API Key de OpenAI en el archivo .env del backend:
  + OPENAI\_API\_KEY=sk-proj-xsGQDDcP2HfxWZRBT05OOZ0Y-irUAl9shgneah8BoDskpVAd7xBVze8taWQQwEor7YIC1RRkTST3BlbkFJc8QMovU\_yOoWrYy85Z1O-UjUKXYJsbs3uaaWXegQTxo3v8XkjwPmKYlnoACCitb3ms7XJnHVAA

**2.2. Modificación del Servidor Backend**

* Se modificó el endpoint /api/transcribe-audio en server.js para:
  + Aceptar un nuevo parámetro processWithGPT
  + Implementar lógica para invocar un script Python (process\_transcript.py) cuando el parámetro es verdadero
  + Configurar la ruta del archivo procesado y devolverla en la respuesta

**2.3. Actualización de la Interfaz de Usuario**

* Se modificó el componente FileList.js:
  + Se agregó un diálogo de confirmación para preguntar al usuario si desea procesar la transcripción con ChatGPT
  + Se actualizó el mensaje de carga para indicar cuando se está aplicando procesamiento adicional
  + Se modificó la visualización del resultado para incluir información sobre el archivo procesado

**2.4. Actualización del Servicio de API**

* Se modificó la función transcribeAudio en api.js:
  + Se agregó un nuevo parámetro processWithGPT con valor predeterminado false
  + Se actualizó la documentación JSDoc para reflejar el nuevo parámetro
  + Se modificó la solicitud HTTP para incluir el nuevo parámetro en el cuerpo JSON

**3. Pruebas Realizadas**

* Se realizó una prueba del sistema modificado con un archivo MP3 real
* Se confirmó que la transcripción básica funcionaba correctamente
* Se identificó un problema: aunque se muestra un mensaje de éxito, no se está generando el segundo archivo con la transcripción procesada por ChatGPT

**Problemas Identificados y Pendientes**

**Problema Principal: No se genera el archivo procesado por ChatGPT**

A pesar de que el sistema muestra un mensaje indicando que hay una "Versión mejorada con IA disponible", en realidad está mostrando la misma ruta del archivo de transcripción original, lo que indica que el procesamiento con ChatGPT no se está realizando correctamente.

**Posibles causas:**

1. **Script process\_transcript.py faltante o con errores**: Este script, mencionado en la bitácora inicial, debería existir en E:\Admin7\master\backend\scripts\ pero no se ha verificado su existencia ni su contenido.
2. **Falta de prompt personalizado**: El sistema no está enviando ningún prompt específico a ChatGPT para guiar el procesamiento de la transcripción.
3. **Error en la ejecución del script Python**: Puede haber un error al invocar el script desde Node.js que no se está manejando correctamente.

**Tareas Pendientes**

Para completar la implementación del procesamiento con ChatGPT, se deben realizar las siguientes tareas:

1. **Verificar la existencia del script process\_transcript.py**:
   * Comprobar si existe en E:\Admin7\master\backend\scripts\
   * Si no existe, crearlo con la lógica necesaria para:
     + Recibir la ruta del archivo de transcripción original
     + Leer su contenido
     + Enviar el contenido a la API de OpenAI con un prompt personalizado
     + Guardar la respuesta en un nuevo archivo con un nombre que indique que fue procesado
2. **Implementar un prompt personalizado en el script Python**:
   * El prompt debe guiar a ChatGPT para mejorar la transcripción
   * Debe especificar el formato, estilo y tipo de mejoras esperadas
3. **Corregir la ruta de salida del archivo procesado**:
   * En server.js, asegurarse de que se está construyendo correctamente la ruta del archivo procesado
   * Verificar que el nombre del archivo procesado sea diferente al de la transcripción original
4. **Mejorar el manejo de errores**:
   * Implementar un mejor sistema de registro de errores para poder diagnosticar problemas en la ejecución del script Python
   * Asegurarse de que los errores se propagan correctamente al frontend
5. **Realizar pruebas completas**:
   * Probar con diferentes archivos MP3
   * Verificar que se generan dos archivos distintos: la transcripción original y la versión mejorada por ChatGPT
   * Comparar el contenido de ambos archivos para confirmar que el procesamiento se está realizando correctamente

**Conclusión**

La implementación del procesamiento de transcripciones con ChatGPT está parcialmente completada. La infraestructura básica (modificaciones al servidor, frontend y API) está en su lugar, pero falta desarrollar o corregir el componente clave: el script process\_transcript.py que debe realizar el procesamiento efectivo del texto mediante la API de OpenAI, incluyendo un prompt personalizado que guíe la mejora de la transcripción.

Este desarrollo pendiente es crucial para completar la funcionalidad y permitir que los usuarios puedan beneficiarse de transcripciones mejoradas automáticamente con inteligencia artificial.

**Bitácora Ultra Detallada: Implementación de procesamiento de transcripciones con IA**

**Punto de partida: Bitácora inicial**

La tarea consistía en implementar un sistema de procesamiento de transcripciones de audio con ChatGPT al sistema DocuBox. Este desarrollo incluía la capacidad de procesar archivos MP3, transcribirlos a texto, y opcionalmente mejorar esa transcripción utilizando la API de OpenAI (ChatGPT).

La bitácora inicial proporcionada indicaba que el sistema DocuBox ya contaba con capacidad de transcripción de audio MP3 a texto, pero necesitaba ampliarse para permitir el procesamiento adicional de las transcripciones mediante ChatGPT.

Se habían identificado los siguientes archivos clave:

* E:\Admin7\master\backend\server.js: Servidor principal de Node.js
* E:\Admin7\master\backend.env: Archivo de variables de entorno
* E:\Admin7\master\frontend\src\components\FileList.js: Componente de interfaz de usuario
* E:\Admin7\master\frontend\src\services\api.js: Servicios de API para el frontend

También se había identificado un problema principal: a pesar de que el sistema mostraba un mensaje indicando que había una "Versión mejorada con IA disponible", en realidad estaba mostrando la misma ruta del archivo de transcripción original, indicando que el procesamiento con ChatGPT no se estaba realizando correctamente.

**Trabajo realizado en esta sesión**

**1. Análisis del script process\_transcript.py**

Se revisó el archivo E:\Admin7\master\backend\scripts\process\_transcript.py para entender su funcionalidad. Este script estaba diseñado para tomar un archivo de transcripción, procesarlo con la API de OpenAI (ChatGPT) y generar un archivo mejorado.

El script incluía funcionalidades para:

* Leer el archivo de transcripción
* Extraer la sección relevante del texto
* Crear un prompt para ChatGPT
* Realizar la llamada a la API de OpenAI
* Guardar la respuesta en un nuevo archivo

**2. Modificación del script process\_transcript.py**

Se realizaron modificaciones al script para mejorar su funcionalidad:

* Se integró un prompt personalizado específico para formatear actas de asambleas
* Se cambió el formato del nombre del archivo de salida a "\*\_acta\_formatada.txt"
* Se eliminó la parte donde se incluía la transcripción original en el archivo de salida
* Se ajustó el parámetro temperature a 0.5 para obtener resultados más consistentes

El script quedó configurado para recibir los siguientes parámetros:

* Ruta del archivo de transcripción (obligatorio)
* API Key de OpenAI (parámetro --api\_key, obligatorio)
* Prompt personalizado (parámetro --prompt, opcional)
* Ruta de salida (parámetro --output, opcional)

**3. Actualización del servidor (server.js)**

Se identificó que el servidor necesitaba actualizaciones para llamar correctamente al script Python y manejar los resultados. Se modificó el endpoint /api/transcribe-audio para:

* Implementar un mecanismo de reintentos con backoff exponencial para operaciones que pudieran fallar
* Corregir la forma en que se llamaba al script process\_transcript.py, pasando los parámetros correctos
* Mejorar la detección del archivo de salida generado por el script de ChatGPT
* Implementar un mejor manejo de errores

**4. Implementación de una solución para manejo de errores de conexión**

Se detectaron problemas de timeout al conectarse con Supabase, lo que impedía el funcionamiento correcto del sistema. Para solucionar esto, se implementó:

* Una función de utilidad retryOperation que permitía reintentar operaciones asíncronas con un retraso exponencial entre intentos
* Aplicación de esta función a las operaciones de subida, descarga y eliminación en Supabase
* Mejores mensajes de registro para seguir el progreso y depurar posibles problemas

**5. Creación de solución alternativa local**

Debido a los problemas persistentes de conexión con Supabase, se comenzó a implementar una solución alternativa local que permitiera:

* Crear un directorio local\_storage para almacenar archivos
* Crear nuevos endpoints para subir, procesar y descargar archivos localmente
* Implementar un flujo completo que no dependiera de Supabase

Para esto se comenzaron a implementar los siguientes endpoints:

* /api/upload-audio-local: Para subir archivos MP3 directamente al servidor
* /api/transcribe-local: Para transcribir archivos locales y opcionalmente procesarlos con ChatGPT
* /api/download-local: Para descargar los archivos generados

**6. Pruebas de la funcionalidad de procesamiento con OpenAI**

Se creó un script de prueba independiente (test\_gpt\_processing.js) para verificar el funcionamiento del script process\_transcript.py sin depender del servidor completo. Este script:

* Creaba un archivo de prueba con una transcripción simulada
* Llamaba al script Python para procesarlo
* Mostraba el resultado o los errores

Durante estas pruebas se descubrieron problemas con la API de OpenAI:

* Error de "model\_not\_found" al intentar usar "gpt-4-turbo" y "gpt-4"
* Problemas de límite de tasa (rate limiting) al usar "gpt-3.5-turbo"

**7. Exploración de alternativas con Claude**

Se empezó a explorar la posibilidad de usar la API de Claude (Anthropic) como alternativa a OpenAI. Se comenzó a desarrollar:

* Un nuevo script process\_transcript\_claude.py para usar la API de Claude
* Un script de prueba test\_claude\_processing.js para verificar su funcionamiento

**Problemas identificados y pendientes**

**Problemas solucionados:**

1. Implementación del prompt personalizado en el script process\_transcript.py
2. Corrección de la forma de llamar al script desde server.js
3. Implementación de un mecanismo de reintentos para operaciones con Supabase

**Problemas pendientes:**

1. **Problemas de conectividad con Supabase**: Siguen ocurriendo errores de timeout al intentar conectarse a Supabase, lo que impide el funcionamiento normal del sistema.
   * Posible solución: Completar la implementación de la solución local que no dependa de Supabase.
2. **Limitaciones con la API de OpenAI**: Se encontraron problemas con los modelos y límites de tasa.
   * Posible solución: Completar la implementación de la alternativa con Claude.
3. **Error en los nombres de archivos procesados**: Se detectó que el sistema muestra la misma ruta para la transcripción normal y la mejorada.
   * Pendiente: Verificar que la ruta del archivo procesado sea correctamente actualizada en la respuesta.
4. **Estructura del endpoint local**: Hubo problemas de sintaxis al implementar los nuevos endpoints locales.
   * Pendiente: Corregir la estructura de los endpoints y asegurar que todas las llaves y paréntesis estén balanceados.

**Conclusiones y siguientes pasos**

La implementación del procesamiento de transcripciones con IA ha avanzado significativamente, pero aún enfrenta desafíos importantes relacionados con la conectividad a servicios externos (Supabase y OpenAI).

Los próximos pasos recomendados son:

1. Completar la implementación de la solución local para no depender de Supabase
2. Finalizar la implementación de la alternativa con Claude
3. Realizar pruebas completas con archivos MP3 reales
4. Actualizar la interfaz de usuario para reflejar el nuevo flujo de trabajo

Todos los cambios realizados están orientados a hacer que el sistema sea más robusto ante fallos de red y limitaciones de API, permitiendo que el procesamiento de transcripciones funcione de manera confiable incluso en condiciones subóptimas.

**Bitácora Detallada: Implementación de procesamiento de transcripciones con Perplexity AI**

**Contexto Inicial**

Partiendo de la bitácora inicial proporcionada, la tarea consistía en implementar un sistema de procesamiento de transcripciones de audio con ChatGPT al sistema DocuBox. En la implementación original, este desarrollo incluía la capacidad de procesar archivos MP3, transcribirlos a texto, y opcionalmente mejorar esa transcripción utilizando la API de OpenAI (ChatGPT). Sin embargo, en nuestra sesión decidimos modificarlo para utilizar Perplexity AI en lugar de Claude o ChatGPT.

La bitácora inicial indicaba que el sistema DocuBox ya contaba con capacidad de transcripción de audio MP3 a texto, pero necesitaba ampliarse para permitir el procesamiento adicional de las transcripciones mediante un servicio de IA.

Los archivos clave identificados originalmente eran:

* E:\Admin7\master\backend\server.js: Servidor principal de Node.js
* E:\Admin7\master\backend.env: Archivo de variables de entorno
* E:\Admin7\master\frontend\src\components\FileList.js: Componente de interfaz de usuario
* E:\Admin7\master\frontend\src\services\api.js: Servicios de API para el frontend

**Trabajo Realizado Durante la Sesión**

**1. Obtención y Configuración de la Clave API de Perplexity**

Se obtuvo una clave API de Perplexity: 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.

Se añadió esta clave al archivo de variables de entorno del backend:

* Archivo: E:\Admin7\master\backend.env
* Línea añadida: PERPLEXITY\_API\_KEY=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

**2. Desarrollo de Scripts de Prueba**

**2.1. Script JavaScript para Prueba de Procesamiento**

Se desarrolló un nuevo script de prueba llamado test\_perplexity\_new.js en la ruta E:\Admin7\master\backend. Este script tenía como propósito:

* Crear una carpeta local\_storage si no existía
* Crear una carpeta scripts si no existía
* Generar un archivo de prueba con texto de transcripción simulada
* Llamar a un script Python para procesar esta transcripción con la API de Perplexity
* Mostrar el resultado del procesamiento

**2.2. Script Python para Interactuar con la API de Perplexity**

Se desarrolló un script llamado process\_transcript\_perplexity.py en la ruta E:\Admin7\master\backend\scripts. Este script tenía como propósito:

* Recibir el archivo de transcripción y procesarlo
* Conectarse a la API de Perplexity
* Enviar un prompt personalizado para formatear la transcripción como un acta formal
* Guardar el resultado en un nuevo archivo

**2.3. Script Python Simplificado para Prueba de Conexión**

Debido a problemas con los scripts más complejos, se desarrolló un script Python minimalista llamado test\_perplexity\_simple.py en la ruta E:\Admin7\master\backend\scripts. Este script tenía como propósito:

* Probar la conexión básica con la API de Perplexity
* Enviar una solicitud simple a la API
* Mostrar la respuesta recibida

También se desarrolló un script JavaScript complementario llamado test\_simple.js en la ruta E:\Admin7\master\backend\ para llamar a este script Python simplificado.

**3. Pruebas y Depuración**

Se realizaron múltiples intentos para ejecutar los scripts, encontrando diversos problemas:

**3.1. Problemas de Sintaxis en el Archivo JavaScript**

* Error inicial: Presencia de caracteres # inesperados al principio del archivo
* Solución: Recreación completa del archivo JavaScript

**3.2. Problemas con Caracteres Especiales en el Archivo Python**

* Error: Caracteres especiales como "¡" generaban error de codificación
* Solución: Recreación del archivo Python eliminando todos los acentos y caracteres especiales

**3.3. Problemas de Estructura en el Código Python**

* Error: Problemas de indentación y errores de sintaxis
* Solución: Simplificación del código Python para hacerlo más robusto

**3.4. Identificación del Error con el Modelo de Perplexity**

* Al ejecutar la prueba simple, se descubrió que el modelo especificado "llama-3-sonar-small-32k-online" no es válido
* Mensaje de error: "Invalid model 'llama-3-sonar-small-32k-online'. Permitted models can be found in the documentation at https://docs.perplexity.ai/guides/model-cards."

**Problemas Identificados y Pendientes**

**1. Modelo de Perplexity Incorrecto**

El principal problema identificado es que el modelo especificado para la API de Perplexity (llama-3-sonar-small-32k-online) no es válido. Según el mensaje de error, es necesario consultar la documentación oficial de Perplexity para obtener la lista de modelos permitidos.

**2. Problemas de Codificación de Caracteres**

Durante el desarrollo se encontraron problemas con la codificación de caracteres especiales en los archivos Python, lo que generaba errores al ejecutar los scripts.

**3. Estructura del Proyecto**

Aunque se crearon las carpetas necesarias durante la ejecución, sería recomendable verificar que la estructura del proyecto está correctamente configurada:

* E:\Admin7\master\backend\scripts\ para los scripts Python
* E:\Admin7\master\backend\local\_storage\ para los archivos temporales

**Tareas Pendientes**

Para completar la implementación del procesamiento con Perplexity AI, se deben realizar las siguientes tareas:

1. **Corregir el modelo de Perplexity**:
   * Consultar la documentación oficial en https://docs.perplexity.ai/guides/model-cards
   * Actualizar el código con un modelo válido
2. **Completar y probar el script de procesamiento**:
   * Una vez corregido el modelo, verificar que el procesamiento funciona correctamente
   * Probar con un archivo de transcripción real
3. **Integrar con el servidor principal**:
   * Modificar server.js para incorporar la funcionalidad de procesamiento con Perplexity
   * Actualizar el frontend para mostrar la opción de procesamiento con Perplexity
4. **Mejorar el manejo de errores**:
   * Implementar un sistema robusto de manejo de errores
   * Documentar los posibles errores y sus soluciones
5. **Documentación**:
   * Completar la documentación del código
   * Crear una guía de uso para los usuarios finales

**Conclusión**

La implementación del procesamiento de transcripciones con Perplexity AI está parcialmente avanzada. Se ha establecido la estructura básica y se han desarrollado los scripts necesarios, pero se encontró un problema con el modelo especificado en la API.

El próximo paso crucial es identificar el modelo correcto en la documentación de Perplexity y actualizar los scripts para usar ese modelo. Una vez solucionado este problema, se podrá continuar con la integración en el sistema principal y las pruebas con archivos reales.

Este desarrollo permitirá ampliar las capacidades del sistema DocuBox, ofreciendo a los usuarios la posibilidad de mejorar automáticamente las transcripciones de audio utilizando la inteligencia artificial de Perplexity.

**Bitácora Ultra Detallada: Implementación de procesamiento de transcripciones con Perplexity AI**

**Punto de partida: Bitácora inicial**

Partiendo de la bitácora inicial proporcionada, la tarea consistía en implementar un sistema de procesamiento de transcripciones de audio con ChatGPT al sistema DocuBox. En la implementación original, este desarrollo incluía la capacidad de procesar archivos MP3, transcribirlos a texto, y opcionalmente mejorar esa transcripción utilizando la API de OpenAI (ChatGPT). Sin embargo, en nuestra sesión decidimos modificarlo para utilizar Perplexity AI en lugar de Claude o ChatGPT.

La bitácora inicial indicaba que el sistema DocuBox ya contaba con capacidad de transcripción de audio MP3 a texto, pero necesitaba ampliarse para permitir el procesamiento adicional de las transcripciones mediante un servicio de IA.

Los archivos clave identificados originalmente eran:

* E:\Admin7\master\backend\server.js: Servidor principal de Node.js
* E:\Admin7\master\backend.env: Archivo de variables de entorno
* E:\Admin7\master\frontend\src\components\FileList.js: Componente de interfaz de usuario
* E:\Admin7\master\frontend\src\services\api.js: Servicios de API para el frontend

**Trabajo realizado durante la sesión actual**

**1. Análisis de los scripts de prueba existentes**

Se revisaron dos scripts de JavaScript de prueba:

* E:\Admin7\master\backend\test\_simple.js - Script para probar la conexión básica con la API de Perplexity
* E:\Admin7\master\backend\test\_perplexity\_new.js - Script para probar el procesamiento completo de transcripciones

Al ejecutar estos scripts, se identificaron los siguientes problemas:

* El script test\_simple.js se ejecutaba sin errores de sintaxis pero devolvía un error 400 de la API de Perplexity por modelo inválido
* El script test\_perplexity\_new.js fallaba debido a un error de sintaxis en el script Python process\_transcript\_perplexity.py (había un "return" fuera de una función)

**2. Análisis de los scripts Python existentes**

Se analizaron dos scripts Python:

* E:\Admin7\master\backend\scripts\test\_perplexity\_simple.py - Script para probar la conexión básica
* E:\Admin7\master\backend\scripts\process\_transcript\_perplexity.py - Script para procesar transcripciones

Problemas identificados:

* En test\_perplexity\_simple.py: Uso de un modelo no válido "llama-3-sonar-small-32k-online"
* En process\_transcript\_perplexity.py: Problemas de indentación en una sección del código que provocaba el error "return outside function"

**3. Corrección del problema de indentación en process\_transcript\_perplexity.py**

Se identificó que la sección después de definir la URL de la API no estaba correctamente indentada dentro de la función process\_transcript\_with\_perplexity.

Ubicación exacta del problema:

* Archivo: E:\Admin7\master\backend\scripts\process\_transcript\_perplexity.py
* Código con indentación incorrecta:
* # URL de la API de Perplexityapi\_url = "https://api.perplexity.ai/chat/completions" # Realizar la llamada a la API

print("Llamando a la API de Perplexity...") print(f"Usando clave API: {api\_key[:5]}...{api\_key[-5:]}")

Corrección:

- Se ajustó la indentación correctamente para que todo el bloque de código estuviera dentro de la función.

- Esta corrección era necesaria para solucionar el error de sintaxis "return outside function"

### 4. Experimentos con diferentes nombres de modelos de Perplexity

Se realizaron múltiples intentos para encontrar un nombre de modelo válido:

#### 4.1. Primera actualización:

- Se actualizó el modelo de "llama-3-sonar-small-32k-online" a "llama-3-sonar-small" en ambos scripts Python

- Al probar, siguió devolviendo error 400 con mensaje de modelo inválido

#### 4.2. Segunda actualización:

- Se actualizó el modelo a "sonar-small-online" en ambos scripts Python

- Al probar, siguió devolviendo error 400 con mensaje de modelo inválido

#### 4.3. Tercera actualización:

- Se actualizó el modelo a "pplx/7b-online" en ambos scripts Python

- Al probar, siguió devolviendo error 400 con mensaje de modelo inválido

#### 4.4. Cuarta actualización:

- Se actualizó el modelo a "mistral-7b-instruct" en ambos scripts Python

- Al probar, siguió devolviendo error 400 con mensaje de modelo inválido

#### 4.5. Quinta actualización:

- Se actualizó el modelo a "pplx/llama-3-8b" en ambos scripts Python

- Al probar, siguió devolviendo error 400 con mensaje de modelo inválido

#### 4.6. Sexta actualización:

- Se actualizó el modelo a "codellama-70b-instruct" en ambos scripts Python

- Al probar, siguió devolviendo error 400 con mensaje de modelo inválido

#### 4.7. Séptima actualización:

- Se actualizó el modelo a "mistral-7b" en ambos scripts Python

- Al probar, siguió devolviendo error 400 con mensaje de modelo inválido

#### 4.8. Octava actualización:

- Se actualizó el modelo a "Perplex 7B" en ambos scripts Python

- Al probar, siguió devolviendo error 400 con mensaje de modelo inválido

### 5. Desarrollo de scripts de diagnóstico adicionales

#### 5.1. Creación del script list\_perplexity\_models.py:

- Ruta: E:\Admin7\master\backend\scripts\list\_perplexity\_models.py

- Propósito: Intentar listar los modelos disponibles en la cuenta de Perplexity

- Resultado: Error 404, indicando que Perplexity no tiene un endpoint /models para listar modelos disponibles

#### 5.2. Creación del script perplexity\_raw\_test.py:

- Ruta: E:\Admin7\master\backend\scripts\perplexity\_raw\_test.py

- Propósito: Usar un modelo deliberadamente incorrecto para obtener un mensaje de error más detallado

- Resultado: Se obtuvo el mismo mensaje de error pero no proporcionó información adicional sobre modelos válidos

#### 5.3. Creación del script list\_models.js:

- Ruta: E:\Admin7\master\backend\list\_models.js

- Propósito: Script Node.js para llamar a list\_perplexity\_models.py

- Resultado: Confirmó que no existe un endpoint para listar modelos

#### 5.4. Creación del script raw\_test.js:

- Ruta: E:\Admin7\master\backend\raw\_test.js

- Propósito: Script Node.js para llamar a perplexity\_raw\_test.py

- Resultado: No proporcionó información adicional sobre modelos válidos

### 6. Consulta externa sobre modelos válidos de Perplexity

Se realizó una consulta directa a Perplexity sobre sus modelos disponibles, obteniendo la siguiente información:

- Modelos mencionados: llama-3-8b-instruct, llama-3.1-8b, llama-3.1-70b

- Otros modelos potencialmente disponibles: CodLlama, Llama 2 70B, Mistral 7B, Mistral 8x7B, Perplex 7B, Perplex 70B

A pesar de intentar con varios de estos nombres de modelos, todos siguieron devolviendo el error 400 con el mensaje de modelo inválido.

### 7. Preparación para contacto con soporte técnico

Se elaboró un mensaje detallado para el soporte técnico de Perplexity, incluyendo:

- Descripción del problema (error consistente de "modelo inválido")

- Detalles técnicos de los intentos realizados

- Ejemplo del código utilizado

- Solicitud específica de nombres de modelos válidos para la API

## Problemas identificados y pendientes

### 1. Problema principal: Imposibilidad de conectarse a la API de Perplexity con los modelos intentados

- Todos los nombres de modelos probados (llama-3-sonar-small-32k-online, llama-3-sonar-small, sonar-small-online, pplx/7b-online, mistral-7b-instruct, mistral-7b, pplx/llama-3-8b, codellama-70b-instruct, Perplex 7B) generaron el mismo error 400 con mensaje de modelo inválido.

- Se corrigió el error de indentación en process\_transcript\_perplexity.py, pero el problema principal de conexión con la API persiste.

### 2. Posibles causas identificadas:

- \*\*Problema de créditos o suscripción\*\*: Es posible que la cuenta de Perplexity necesite créditos o una suscripción activa para acceder a ciertos modelos.

- \*\*Modelos restringidos\*\*: La cuenta podría tener acceso limitado a ciertos modelos específicos.

- \*\*Cambio en la API\*\*: Podría haber habido un cambio reciente en la API de Perplexity y la documentación no está actualizada.

- \*\*Formato incorrecto\*\*: Es posible que el nombre del modelo requiera un formato específico no documentado.

### 3. Tareas pendientes:

- \*\*Contactar al soporte técnico de Perplexity\*\*: Enviar el mensaje elaborado para obtener información específica sobre:

- Nombres exactos de modelos que pueden utilizarse con la cuenta

- Verificación del estado de créditos/suscripción

- Ejemplo de código funcional para una solicitud básica

- \*\*Explorar opciones alternativas\*\*: Considerar retomar las implementaciones anteriores con OpenAI (ChatGPT) o Claude que ya se habían empezado a desarrollar.

- \*\*Prueba con respuesta del soporte\*\*: Una vez recibida la respuesta del soporte de Perplexity, probar con los nombres de modelos exactos proporcionados.

## Conclusión

La implementación del procesamiento de transcripciones con Perplexity AI ha avanzado en términos de la estructura del código y la corrección de errores de sintaxis, pero enfrenta un obstáculo fundamental con la conexión a la API y la identificación de modelos válidos.

Se han corregido problemas de indentación en el script principal, se ha explorado exhaustivamente la documentación disponible, y se han realizado múltiples pruebas con diferentes nombres de modelos, pero el problema persiste.

El siguiente paso crítico es obtener información definitiva del soporte técnico de Perplexity sobre los modelos válidos para la cuenta específica, o considerar retomar las alternativas con OpenAI o Claude para avanzar con el proyecto.

Todos los cambios realizados están documentados en esta bitácora, permitiendo continuar el desarrollo una vez que se obtenga la información necesaria sobre los modelos válidos o se tome la decisión de cambiar a otra API de IA.

Respuesta de perplexity:

Hola,

He revisado tu consulta sobre el error "Invalid model" y puedo ayudarte. El problema se debe a que los nombres de modelos que estás intentando usar no son los correctos. Los modelos válidos disponibles en la API de Perplexity son:

* sonar
* sonar-pro
* sonar-reasoning
* sonar-reasoning-pro
* sonar-deep-research
* r1-1776

Aquí tienes un ejemplo de código Python funcional:

import requests headers = { "accept": "application/json", "content-type": "application/json", "authorization": f"Bearer {api\_key}" } data = { "model": "sonar", # O cualquier otro modelo de la lista anterior "messages": [ {"role": "system", "content": "..."}, {"role": "user", "content": "..."} ] } response = requests.post("https://api.perplexity.ai/chat/completions", headers=headers, json=data)

Respecto a tus créditos y permisos de API, esto depende de tu nivel de uso actual. Te sugiero que revises tu página de "Usage & Billing" en el dashboard de la API para verificar tu nivel de acceso y créditos disponibles. Si tienes dudas sobre tu nivel de acceso, puedes compartir una captura de pantalla de tu dashboard de la API.

Intenta modificar tu código usando uno de los nombres de modelo válidos mencionados arriba y házme saber si continúas teniendo problemas.

¿Hay algo más en lo que pueda ayudarte?

Regards,  
Sam  
AI Support agent for Perplexity