Flask-Chat

Generated by Doxygen 1.9.4

Chapter 1

Flask Chat API

1.1 Descripción

Aplicación backend en Flask para la generación de la API que consultará el frontend web de Flask Chat

1.2 Notas

- Notas principales del proyecto

Chapter 2

Namespace Index

2.1 Package List

Here are the packages with brief descriptions (if available):

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 chatgpt_flask Namespace Reference

Namespaces

- namespace app

  Elemento principal de la aplicación FlaskChat.
- namespace chatgpt_api

4.2 chatgpt_flask.app Namespace Reference

Elemento principal de la aplicación FlaskChat.

Functions

- def create_app ()

  Inicializar el programa.

Variables

- def app = create_app()
- debug
- host
- port

4.2.1 Detailed Description

Elemento principal de la aplicación FlaskChat.

4.2.2 Function Documentation

4.2.2.1 create_app()

```
def create_app ( )
```

Inicializar el programa.

4.2.3 Variable Documentation

4.2.3.1 app

```
def app = create_app()
```

4.2.3.2 debug

```
debug
```

4.2.3.3 host

```
host
```

4.2.3.4 port

```
port
```

4.3 chatgpt_flask.chatgpt_api Namespace Reference

Namespaces

- namespace api
- namespace db

4.4 chatgpt_flask.chatgpt_api.api Namespace Reference

Namespaces

- namespace contexto
- namespace conversacion
- namespace mensaje
- namespace modelo
- namespace tools

4.5 chatgpt_flask.chatgpt_api.api.contexto Namespace Reference

Functions

- def obtener_contexto (id)

  Recupera el estado del contexto de una conversación específica.
- def toggle_contexto (id)

  Cambia el estado del contexto de una conversación específica.

Variables

- contexto_bp = Blueprint('contexto', __name__)

4.5.1 Function Documentation

4.5.1.1 obtener_contexto()

```
def obtener_contexto (
              id )
```

Recupera el estado del contexto de una conversación específica.

Este endpoint permite obtener el estado del contexto asociado a una conversación. El contexto se guarda como un valor booleano (`True` o `False`).

Parameters

id	Identificador único de la conversación cuya configuración de contexto se quiere consultar.

Returns

- 200 OK: Si la operación es exitosa, devuelve el estado actual del contexto.
- 404 Not Found: Si no se encuentra una conversación con el ID proporcionado.

Note

Si la conversación con el ID proporcionado no existe, se devuelve un error 404.

```
Ejemplo de solicitud:
GET /api/contexto/1
Respuesta esperada:
{
    "contexto":  true
}
```

4.5.1.2 toggle_contexto()

```
def toggle_contexto (
               id )
```

Cambia el estado del contexto de una conversación específica.

Este endpoint permite alternar el estado del contexto de una conversación en la base de datos. Si el contexto está desactivado (`False`), se activa (`True`), y si está activado, se desactiva.

Parameters

id	Identificador único de la conversación cuya configuración de contexto se quiere modificar.

Returns

- 200 OK: Si la operación es exitosa, devuelve el nuevo estado del contexto.
- 404 Not Found: Si no se encuentra una conversación con el ID proporcionado.

Note

Si la conversación con el ID proporcionado no existe, se devuelve un error 404.

```
Ejemplo de solicitud:
POST /api/contexto/1
Respuesta esperada:
{
    "contexto":  true
}
```

4.5.2 Variable Documentation

4.5.2.1 contexto_bp

```
contexto_bp = Blueprint('contexto', __name__)
```

4.6 chatgpt_flask.chatgpt_api.api.conversacion Namespace Reference

Functions

- def historial ()

  Recupera el historial de todas las conversaciones.
- def nueva_conversacion ()

  Crea una nueva conversación en la base de datos.

Variables

- conversacion_bp = Blueprint('conversacion', __name__)

4.6.1 Function Documentation

4.6.1.1 historial()

```
def historial ( )
```

Recupera el historial de todas las conversaciones.

Este endpoint devuelve una lista de todas las conversaciones almacenadas en la base de datos, ordenadas por fecha de creación en orden descendente. Cada conversación incluye su ID, nombre y fecha de creación.

Returns

- 200 OK: Devuelve un historial de conversaciones en formato JSON.

```
Ejemplo de solicitud:
GET /api/historial
Respuesta esperada:
[
    {
        "id": 123,
        "nombre": "Mi Conversación",
        "fecha_creacion": "2025-04-14T10:00:00"
    },
    ...
]
```

4.6.1.2 nueva_conversacion()

```
def nueva_conversacion ( )
```

Crea una nueva conversación en la base de datos.

Este endpoint permite crear una nueva conversación. El nombre de la conversación se puede especificar en la solicitud, o se asignará el nombre "Nueva Conversación" por defecto si no se proporciona. La nueva conversación se almacena en la base de datos y se devuelve el ID generado y el nombre asignado.

Parameters

nombre	Nombre de la nueva conversación. Si no se proporciona, se utiliza el valor predeterminado "Nueva Conversación".

Returns

- 201 Created: Si la conversación se crea correctamente, devuelve el ID y el nombre de la nueva conversación.

Note

El nombre de la conversación es opcional. Si no se especifica en la solicitud, se utilizará el valor predeterminado.

```
Ejemplo de solicitud:
POST /api/chat
{
    "nombre": "Mi Conversación"
}
Respuesta esperada:
{
    "id": 123,
    "nombre": "Mi Conversación"
}
```

4.6.2 Variable Documentation

4.6.2.1 conversacion_bp

```
conversacion_bp = Blueprint('conversacion', __name__)
```

4.7 chatgpt_flask.chatgpt_api.api.mensaje Namespace Reference

Functions

- def enviar_mensaje (id)

  Envía un mensaje de usuario y obtiene la respuesta del asistente.
- def obtener_mensajes (id)

  Recupera los mensajes de una conversación específica.

Variables

- mensaje_bp = Blueprint('mensaje', __name__)

4.7.1 Function Documentation

4.7.1.1 enviar_mensaje()

```
def enviar_mensaje (
              id )
```

Envía un mensaje de usuario y obtiene la respuesta del asistente.

Este endpoint permite enviar un mensaje de usuario a la conversación especificada. Después de recibir el mensaje, el sistema genera una respuesta utilizando la API de OpenAI. Si el contexto está activado, se conserva el historial de mensajes, y si el número de tokens excede el límite, el historial se resume. El modelo de conversación también se consulta antes de generar una respuesta.

Parameters

id	Identificador único de la conversación a la que se está enviando el mensaje.

Returns

- 200 OK: Si el mensaje es procesado correctamente, devuelve la respuesta del asistente en formato JSON.
- 400 Bad Request: Si el mensaje proporcionado está vacío.
- 500 Internal Server Error: Si ocurre un error al comunicarse con OpenAI.

Note

Si el contexto está activado en la conversación, el sistema almacenará y procesará el historial de mensajes. Si se excede el límite de tokens, el historial será resumido para ajustarse al límite.

```
Ejemplo de solicitud:
POST /api/chat/123
{
    "mensaje":  "¿Cómo puedo integrar la API de OpenAI?"
}
Respuesta esperada:
{
    "respuesta":  "Para integrar la API de OpenAI, necesitas obtener una clave de API..."
}
```

4.7.1.2 obtener_mensajes()

```
def obtener_mensajes (
              id )
```

Recupera los mensajes de una conversación específica.

Este endpoint permite obtener todos los mensajes asociados a una conversación específica en orden de fecha de creación. Devuelve tanto los mensajes de los usuarios como los del sistema (si existen).

Parameters

id	Identificador único de la conversación cuyos mensajes se desean obtener.

Returns

- 200 OK: Devuelve una lista de mensajes de la conversación en formato JSON.

Note

El historial de mensajes se devuelve ordenado por la fecha de creación de cada mensaje. Incluye información de si el mensaje es del usuario o de OpenAI (es_usuario)

```
Ejemplo de solicitud:
GET /api/chat/123
Respuesta esperada:
[
    {
        "mensaje":  "Hola, ¿cómo estás?",
        "es_usuario":  true,
        "fecha_creacion":  "2025-04-14T10:00:00"
    },
    ...
]
```

4.7.2 Variable Documentation

4.7.2.1 mensaje_bp

```
mensaje_bp = Blueprint('mensaje', __name__)
```

4.8 chatgpt_flask.chatgpt_api.api.modelo Namespace Reference

Functions

- def actualizar_modelo (id)

  Actualiza el modelo asociado a una conversación específica.
- def obtener_modelo (id)

  Recupera el modelo asociado a una conversación específica.

Variables

- modelo_bp = Blueprint('modelo', __name__)

4.8.1 Function Documentation

4.8.1.1 actualizar_modelo()

```
def actualizar_modelo (
              id )
```

Actualiza el modelo asociado a una conversación específica.

Este endpoint permite actualizar el modelo de una conversación almacenada en la base de datos. Se valida que el nuevo modelo sea uno de los permitidos (`gpt-3.5-turbo` o `gpt-4`).

Parameters

id	Identificador único de la conversación.
modelo	El modelo a actualizar, que debe ser uno de los modelos permitidos.

Returns

- 200 OK: Si el modelo se actualiza correctamente, devuelve un mensaje de éxito y el nuevo modelo.
- 400 Bad Request: Si el modelo proporcionado no es válido.

Note

Los modelos permitidos son: `gpt-3.5-turbo` y `gpt-4`. Si se intenta actualizar con un modelo diferente, se devuelve un error 400.

Exceptions

Exception	Lanza una excepción si ocurre un error al actualizar la base de datos.

Precondition

La conversación debe existir en la base de datos con el ID proporcionado.

Postcondition

El modelo de la conversación será actualizado con el nuevo modelo.

```
Ejemplo de solicitud:
PUT /api/modelo/1
{
    "modelo":  "gpt-4"
}
Respuesta esperada:
{
    "mensaje":  "Modelo actualizado correctamente",
    "modelo":  "gpt-4"
}
```

4.8.1.2 obtener_modelo()

```
def obtener_modelo (
               id )
```

Recupera el modelo asociado a una conversación específica.

Este endpoint permite obtener el modelo que está asociado a una conversación en la base de datos. La búsqueda se realiza mediante el ID de la conversación.

Parameters

id	Identificador único de la conversación.

Returns

- 200 OK: Si la conversación se encuentra, devuelve el modelo asociado.
- 404 Not Found: Si no se encuentra una conversación con el ID proporcionado.

Note

Si el ID de la conversación no existe en la base de datos, se retornará un error 404.

```
Ejemplo de solicitud:
GET /api/modelo/1
Respuesta esperada:
{
    "modelo":  "gpt-3.5-turbo"
}
```

4.8.2 Variable Documentation

4.8.2.1 modelo_bp

```
modelo_bp = Blueprint('modelo', __name__)
```

4.9 chatgpt_flask.chatgpt_api.api.tools Namespace Reference

Functions

- def delete_file_from_openai ()

  Elimina un archivo previamente subido a la API de OpenAI.
- def list_openai_files ()

  Recupera la lista de archivos subidos a la API de OpenAI.
- def upload_file_to_openai ()

  Carga un archivo a la API de OpenAI.

Variables

- tools_bp = Blueprint('tools', __name__, url_prefix='/api/tools')

4.9.1 Function Documentation

4.9.1.1 delete_file_from_openai()

```
def delete_file_from_openai ( )
```

Elimina un archivo previamente subido a la API de OpenAI.

Esta ruta permite eliminar un archivo que ha sido previamente cargado a la API de OpenAI. Es útil cuando ya no se necesita el archivo o se desea liberar espacio en el sistema.

Esta función espera recibir una solicitud POST con un JSON en el cuerpo que contiene el ID de archivo (`file_id`) a eliminar. La respuesta será un objeto JSON confirmando si la eliminación fue exitosa, junto con el ID del archivo eliminado.

Parameters

file_id	Identificador del archivo que se desea eliminar.

Returns

- 200 OK: Si el archivo fue eliminado correctamente, devuelve un objeto JSON con la propiedad `deleted` igual a `true` y el `id` del archivo eliminado.
- 400 Bad Request: Si no se proporciona el parámetro `file_id`.
- 500 Internal Server Error: Si ocurre un error al intentar eliminar el archivo en la API de OpenAI.

Note

El parámetro `file_id` debe ser una cadena de texto representando el identificador único del archivo en OpenAI.

Exceptions

Lanzará	una excepción si ocurre un error durante la interacción con la API de OpenAI.

Precondition

El archivo debe haber sido previamente cargado en la API de OpenAI.

Postcondition

El archivo será eliminado de manera permanente si se encuentra y se elimina correctamente.

```
Ejemplo de solicitud:
POST /api/tools/delete
{
    "file_id":  "file-abc123"
}
Respuesta esperada:
{
    "deleted":  true,
    "id":  "file-abc123"
}
```

Note

- 400: El `file_id` es obligatorio.
- 500: Error al comunicarse con OpenAI.

4.9.1.2 list_openai_files()

```
def list_openai_files ( )
```

Recupera la lista de archivos subidos a la API de OpenAI.

Este endpoint recupera la lista de archivos que han sido subidos a la API de OpenAI, filtrando únicamente aquellos cuyo propósito sea "assistants". Esto evita incluir archivos de fine-tuning u otros usos no relevantes para el sistema de asistentes.

Esta función realiza una solicitud GET a la API de OpenAI para obtener la lista completa de archivos subidos. Luego filtra los archivos para incluir solo aquellos cuyo propósito es 'assistants', y retorna la lista filtrada de archivos como un objeto JSON.

Note

Este endpoint no requiere parámetros en la solicitud, y solo devuelve archivos relacionados con el uso de asistentes en la API de OpenAI.

Returns

- 200 OK: Retorna una lista de archivos cuyo propósito es 'assistants'. Cada archivo tiene los siguientes campos:
  - id: Identificador único del archivo.
  - filename: Nombre del archivo.
  - purpose: Propósito del archivo (solo 'assistants' será incluido).
  - status: Estado del archivo (ej. 'processed').
  - created_at: Fecha de creación del archivo en formato timestamp.
- 500 Internal Server Error: Si ocurre un error al intentar obtener la lista de archivos de la API de OpenAI.

```
Ejemplo de respuesta:
[
    {
        "id":  "file-abc123",
        "filename":  "documento.pdf",
        "purpose":  "assistants",
        "status":  "processed",
        "created_at":  1710000000
    },
    ...
]
```

Note

- 500: Error al comunicarse con la API de OpenAI o al procesar la respuesta.

4.9.1.3 upload_file_to_openai()

```
def upload_file_to_openai ( )
```

Carga un archivo a la API de OpenAI.

Este endpoint permite cargar un archivo a la API de OpenAI, lo que hace posible su uso con los modelos de OpenAI, incluyendo GPT-4. El archivo debe ser enviado en el cuerpo de la solicitud bajo el formato adecuado (multipart/form-data).

El archivo cargado se almacena en la infraestructura de OpenAI y puede ser utilizado para tareas asociadas con el modelo GPT-4. El endpoint espera un archivo y un propósito (como "assistants") como parámetros.

Parameters

file	Archivo que se va a cargar a la API de OpenAI.

Returns

- 200 OK: Si el archivo se carga correctamente, se devuelve un objeto JSON con el ID del archivo y otros detalles asociados.
- 400 Bad Request: Si el archivo no se proporciona o no es válido.

- 500 Internal Server Error: Si ocurre un error durante el proceso de carga.

Note

- 400: Si no se envía un archivo válido en la solicitud.
- 500: Error al comunicarse con la API de OpenAI.

4.9.2 Variable Documentation

4.9.2.1 tools_bp

```
tools_bp = Blueprint('tools', __name__, url_prefix='/api/tools')
```

4.10 chatgpt_flask.chatgpt_api.db Namespace Reference

Functions

- def close_db (error=None)

  Cierra la conexión a la base de datos.
- def get_db ()

  Obtiene la conexión a la base de datos.
- def init_app (app)

  Inicializa la aplicación Flask para gestionar la base de datos.

Variables

- DATABASE = os.path.join(os.path.dirname(__file__), 'chatgpt.db')

4.10.1 Function Documentation

4.10.1.1 close_db()

```
def close_db (
              error = None )
```

Cierra la conexión a la base de datos.

Esta función cierra la conexión a la base de datos que se ha abierto durante la solicitud. La conexión se gestiona utilizando la variable global `g`. Si no hay una conexión abierta, no se realiza ninguna acción.

Parameters

error	Opción de capturar un error (no se utiliza en este contexto).

Note

La función es llamada automáticamente al finalizar la solicitud, gracias a su integración en el ciclo de vida de Flask.

4.10.1.2 get_db()

```
def get_db ( )
```

Obtiene la conexión a la base de datos.

Esta función establece una conexión con la base de datos SQLite si no se ha creado una conexión previamente en el contexto de la aplicación. Si ya existe una conexión, se reutiliza para evitar conexiones duplicadas. Utiliza la variable global `g` para almacenar la conexión durante el ciclo de vida de la solicitud.

Returns

- Conexión a la base de datos SQLite.

Note

La base de datos se encuentra en el archivo `chatgpt.db` en el directorio del proyecto.

4.10.1.3 init_app()

```
def init_app (
              app )
```

Inicializa la aplicación Flask para gestionar la base de datos.

Esta función se utiliza para integrar las funciones de la base de datos dentro del ciclo de vida de Flask. Registra la función `close_db` para que se ejecute automáticamente cuando la solicitud termine, asegurando que la conexión a la base de datos se cierre correctamente.

Parameters

app	Instancia de la aplicación Flask.

Note

La función `close_db` se ejecutará automáticamente después de cada solicitud gracias a `teardown_appcontext`.

4.10.2 Variable Documentation

4.10.2.1 DATABASE

```
DATABASE = os.path.join(os.path.dirname(__file__), 'chatgpt.db')
```

4.11 openai_service Namespace Reference

Functions

- def contar_tokens (mensajes, modelo="gpt-3.5-turbo")

  Cuenta el número de tokens utilizados por un conjunto de mensajes.
- def obtener_respuesta_openai (mensajes_historial, modelo)

  Obtiene la respuesta de OpenAI para un conjunto de mensajes.
- def obtener_resumen_historial (mensajes_historial)

  Obtiene un resumen del historial de mensajes.

Variables

- api_key
- logger = logging.getLogger(__name__)

4.11.1 Function Documentation

4.11.1.1 contar_tokens()

```
def contar_tokens (
              mensajes,
              modelo = "gpt-3.5-turbo" )
```

Cuenta el número de tokens utilizados por un conjunto de mensajes.

Esta función calcula el número de tokens necesarios para enviar un conjunto de mensajes a la API de OpenAI. El número de tokens es importante para determinar si se excede el límite de tokens del modelo. La función toma en cuenta el modelo utilizado, dado que la codificación y el número de tokens varían entre diferentes modelos.

Parameters

mensajes	Lista de diccionarios que contienen el historial de mensajes. Cada mensaje debe tener un campo `content` con el contenido del mensaje.
modelo	Nombre del modelo de OpenAI que se utilizará para calcular los tokens (e.g., "gpt-3.5-turbo").

Returns

- Integer que representa el número total de tokens utilizados por los mensajes.

Note

La función utiliza el paquete `tiktoken` para calcular los tokens de manera precisa.

```
Ejemplo de uso:
tokens = contar_tokens(mensajes_historial, "gpt-3.5-turbo")
```

4.11.1.2 obtener_respuesta_openai()

```
def obtener_respuesta_openai (
              mensajes_historial,
              modelo )
```

Obtiene la respuesta de OpenAI para un conjunto de mensajes.

Esta función envía el historial de mensajes a la API de OpenAI para obtener una respuesta del modelo especificado. Se utiliza el modelo indicado para generar una respuesta basada en los mensajes previos. El parámetro `modelo` puede ser un modelo como `gpt-3.5-turbo` o `gpt-4`.

Parameters

mensajes_historial	Lista de diccionarios que contienen el historial de mensajes. Cada mensaje debe tener un campo `role` ('user' o 'assistant') y un campo `content` con el contenido del mensaje.
modelo	Nombre del modelo de OpenAI que se utilizará para generar la respuesta (e.g., "gpt-3.5-turbo").

Returns

- String con la respuesta generada por el modelo de OpenAI.

Exceptions

Exception	Si ocurre un error al interactuar con la API de OpenAI, se registrará el error y se devolverá un mensaje indicando el fallo.

Note

La función realiza una solicitud a OpenAI usando el método `openai.chat.completions.create`, y tiene configurado un límite de tokens de 550.

```
Ejemplo de uso:
respuesta = obtener_respuesta_openai(mensajes_historial, "gpt-3.5-turbo")
```

4.11.1.3 obtener_resumen_historial()

```
def obtener_resumen_historial (
              mensajes_historial )
```

Obtiene un resumen del historial de mensajes.

Esta función envía el historial de mensajes a la API de OpenAI para obtener un resumen de los mensajes de manera concisa y clara. El resumen se realiza utilizando el modelo `gpt-3.5-turbo`.

Parameters

mensajes_historial	Lista de diccionarios que contienen el historial de mensajes. Cada mensaje debe tener un campo `role` ('user' o 'assistant') y un campo `content` con el contenido del mensaje.

Returns

- String con el resumen generado por el modelo de OpenAI.

Exceptions

Exception	Si ocurre un error al interactuar con la API de OpenAI, se registrará el error y se devolverá un mensaje indicando el fallo.

Note

El modelo utilizado para resumir es `gpt-3.5-turbo` y tiene configurado un límite de tokens de 500.

```
Ejemplo de uso:
resumen = obtener_resumen_historial(mensajes_historial)
```

4.11.2 Variable Documentation

4.11.2.1 api_key

```
api_key
```

4.11.2.2 logger

```
logger = logging.getLogger(__name__)
```

Chapter 5

File Documentation

5.1 app.py File Reference

Fichero principal de la aplicación.

Namespaces

- namespace chatgpt_flask
- namespace chatgpt_flask.app

  Elemento principal de la aplicación FlaskChat.

Functions

- def create_app ()

  Inicializar el programa.

Variables

- def app = create_app()
- debug
- host
- port

5.1.1 Detailed Description

Fichero principal de la aplicación.

Punto de entrada y carga de los blueprint existentes

5.1.2 Descripción

Un Backend conectado a la API de OpenAI para dar servicio a la parte de Frontend de la aplicación Flask Chat

5.1.3 Libraries/Modules

- time standard library (https://docs.python.org/3/library/time.html)
  - Access to sleep function.
- sensors module (local)
  - Access to Sensor and TempSensor classes.

5.1.4 Notas

- Todos los comentarios deben ser compatibles con Doxygen

5.1.5 TODO

- Nada específico en el horizonte

5.1.6 Autor(es)

- Created by Rafael Sanchez on 14/04/2025.

5.2 __init__.py File Reference

5.3 chatgpt_api/__init__.py File Reference

5.4 chatgpt_api/api/__init__.py File Reference

5.5 chatgpt_api/api/contexto.py File Reference

Namespaces

- namespace chatgpt_flask
- namespace chatgpt_flask.chatgpt_api
- namespace chatgpt_flask.chatgpt_api.api
- namespace chatgpt_flask.chatgpt_api.api.contexto

Functions

- def obtener_contexto (id)

  Recupera el estado del contexto de una conversación específica.
- def toggle_contexto (id)

  Cambia el estado del contexto de una conversación específica.

Variables

- contexto_bp = Blueprint('contexto', __name__)

5.6 chatgpt_api/api/conversacion.py File Reference

Namespaces

- namespace chatgpt_flask
- namespace chatgpt_flask.chatgpt_api
- namespace chatgpt_flask.chatgpt_api.api
- namespace chatgpt_flask.chatgpt_api.api.conversacion

Functions

- def historial ()

  Recupera el historial de todas las conversaciones.
- def nueva_conversacion ()

  Crea una nueva conversación en la base de datos.

Variables

- conversacion_bp = Blueprint('conversacion', __name__)

5.7 chatgpt_api/api/mensaje.py File Reference

Namespaces

- namespace chatgpt_flask
- namespace chatgpt_flask.chatgpt_api
- namespace chatgpt_flask.chatgpt_api.api
- namespace chatgpt_flask.chatgpt_api.api.mensaje

Functions

- def enviar_mensaje (id)

  Envía un mensaje de usuario y obtiene la respuesta del asistente.
- def obtener_mensajes (id)

  Recupera los mensajes de una conversación específica.

Variables

- mensaje_bp = Blueprint('mensaje', __name__)

5.8 chatgpt_api/api/modelo.py File Reference

Namespaces

- namespace chatgpt_flask
- namespace chatgpt_flask.chatgpt_api
- namespace chatgpt_flask.chatgpt_api.api
- namespace chatgpt_flask.chatgpt_api.api.modelo

Functions

- def actualizar_modelo (id)

  Actualiza el modelo asociado a una conversación específica.
- def obtener_modelo (id)

  Recupera el modelo asociado a una conversación específica.

Variables

- modelo_bp = Blueprint('modelo', __name__)

5.9 chatgpt_api/api/tools.py File Reference

Namespaces

- namespace chatgpt_flask
- namespace chatgpt_flask.chatgpt_api
- namespace chatgpt_flask.chatgpt_api.api
- namespace chatgpt_flask.chatgpt_api.api.tools

Functions

- def delete_file_from_openai ()

  Elimina un archivo previamente subido a la API de OpenAI.
- def list_openai_files ()

  Recupera la lista de archivos subidos a la API de OpenAI.
- def upload_file_to_openai ()

  Carga un archivo a la API de OpenAI.

Variables

- tools_bp = Blueprint('tools', __name__, url_prefix='/api/tools')

5.10 chatgpt_api/db.py File Reference

Namespaces

- namespace chatgpt_flask
- namespace chatgpt_flask.chatgpt_api
- namespace chatgpt_flask.chatgpt_api.db

Functions

- def close_db (error=None)

  Cierra la conexión a la base de datos.
- def get_db ()

  Obtiene la conexión a la base de datos.
- def init_app (app)

  Inicializa la aplicación Flask para gestionar la base de datos.

Variables

- DATABASE = os.path.join(os.path.dirname(__file__), 'chatgpt.db')

5.11 chatgpt_api/services/openai_service.py File Reference

Namespaces

- namespace openai_service

Functions

- def contar_tokens (mensajes, modelo="gpt-3.5-turbo")

  Cuenta el número de tokens utilizados por un conjunto de mensajes.
- def obtener_respuesta_openai (mensajes_historial, modelo)

  Obtiene la respuesta de OpenAI para un conjunto de mensajes.
- def obtener_resumen_historial (mensajes_historial)

  Obtiene un resumen del historial de mensajes.

Variables

- api_key
- logger = logging.getLogger(__name__)

Index