Llamada de Función Mistral

```python
import os
import requests
from dotenv import load_dotenv
import argparse
import json

load_dotenv()

def llamar_api_mistral(prompt, model="mistral-small-2501", use_function_calling=False):
    api_key = os.environ.get("MISTRAL_API_KEY")
    if not api_key:
        print("Error: La variable de entorno MISTRAL_API_KEY no está configurada.")
        return None

    url = "https://api.mistral.ai/v1/chat/completions"
    headers = {
        "Content-Type": "application/json",
        "Accept": "application/json",
        "Authorization": f"Bearer {api_key}"
    }

    if use_function_calling:
        data = {
            "model": model,
            "messages": [
                {
                    "role": "user",
                    "content": prompt
                }
            ],
            "tools": [
                {
                    "type": "function",
                    "function": {
                        "name": "get_current_weather",
                        "description": "Obtener el clima actual en una ubicación dada",
                        "parameters": {
                            "type": "object",
```

```
                        "properties": {
                            "location": {
                                "type": "string",
                                "description": "La ciudad y el estado, por ejemplo, San Francisco, CA",
                            },
                            "unit": {"type": "string", "enum": ["celsius", "fahrenheit"]},
                        },
                        "required": ["location"],
                    },
                }
            }
        ],
        "tool_choice": "auto",
    }
else:
    data = {
        "model": model,
        "messages": [
            {
                "role": "user",
                "content": prompt
            }
        ]
    }

print(f"URL de la API Mistral: {url}")
print(f"Encabezados de la API Mistral: {headers}")
print(f"Datos de la API Mistral: {data}")
try:
    response = requests.post(url, headers=headers, json=data)
    response.raise_for_status()
    response_json = response.json()
    print(response_json)
    if response_json and response_json['choices']:
        choice = response_json['choices'][0]
        if 'message' in choice and 'content' in choice['message']:
            content = choice['message']['content']
            return content
        elif 'message' in choice and 'tool_calls' in choice['message']:
            tool_calls = choice['message']['tool_calls']
```

```
                print(f"Llamadas a herramientas: {tool_calls}")
                return tool_calls # Devolver las llamadas a herramientas para su procesamiento
            else:
                print(f"Error de la API Mistral: Formato de respuesta no válido: {response_json}")
                return None
        else:
            print(f"Error de la API Mistral: Formato de respuesta no válido: {response_json}")
            return None
    except requests.exceptions.RequestException as e:
        print(f"Error de la API Mistral: {e}")
        if e.response:
            print(f"Código de estado de la respuesta: {e.response.status_code}")
            print(f"Contenido de la respuesta: {e.response.text}")
        return None

def llamar_api_codestral(prompt, model="codestral-latest"):
    api_key = os.environ.get("MISTRAL_API_KEY")
    if not api_key:
        print("Error: La variable de entorno MISTRAL_API_KEY no está configurada.")
        return None

    url = "https://api.mistral.ai/v1/fim/completions"
    headers = {
        "Content-Type": "application/json",
        "Accept": "application/json",
        "Authorization": f"Bearer {api_key}"
    }
    data = {
        "model": model,
        "prompt": prompt,
        "suffix": "return a + b",
        "max_tokens": 64,
        "temperature": 0
    }
    print(f"URL de la API Codestral: {url}")
    print(f"Encabezados de la API Codestral: {headers}")
    print(f"Datos de la API Codestral: {json.dumps(data)}")
    try:
        response = requests.post(url, headers=headers, json=data)
        response.raise_for_status()
```

```
        response_json = response.json()
        print(response_json)
        if response_json and response_json['choices']:
            content = response_json['choices'][0]['message']['content']
            return content
        else:
            print(f"Error de la API Codestral: Formato de respuesta no válido: {response_json}")
            return None
    except requests.exceptions.RequestException as e:
        print(f"Error de la API Codestral: {e}")
        if e.response:
            print(f"Código de estado de la respuesta: {e.response.status_code}")
            print(f"Contenido de la respuesta: {e.response.text}")
        return None

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Probar las APIs de Mistral y Codestral.")
    parser.add_argument("--type", type=str, default="mistral", choices=["mistral", "codestral"], help="Tipo de
    parser.add_argument("--function_calling", action="store_true", help="Habilitar la llamada de funciones par
    args = parser.parse_args()

    if args.type == "mistral":
        prompt = "¿Cómo está el clima en Londres?"
        response = llamar_api_mistral(prompt, use_function_calling=args.function_calling)
        if response:
            print(f"Respuesta: {response}")
    elif args.type == "codestral":
        prompt = "def f("
        response = llamar_api_codestral(prompt, model="codestral-latest")
        if response:
            print(f"Respuesta: {response}")
```