Prueba de la API de Mistral

Esta publicación es una prueba rápida de la API de Mistral. Estoy utilizando el modelo `mistral-small-2501` para esta prueba. El código a continuación muestra cómo llamar a la API y obtener una respuesta.

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

load_dotenv()

def call_mistral_api(prompt, model="mistral-small-2501"):
    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}"
    }
    data = {
        "model": model,
        "messages": [
            {
                "role": "user",
                "content": prompt
            }
        ]
    }
    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 de Mistral: Formato de respuesta no válido: {response_json}")
            return None
    except requests.exceptions.RequestException as e:
        print(f"Error de la API de 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 call_codestral_api(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
    }
    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 de Codestral: Formato de respuesta no válido: {response_json}")
```

```
            return None
    except requests.exceptions.RequestException as e:
        print(f"Error de la API de 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="Prueba de las APIs de Mistral y Codestral.")
    parser.add_argument("--type", type=str, default="mistral", choices=["mistral", "codestral"], help="Tipo de
    args = parser.parse_args()

    if args.type == "mistral":
        prompt = "what's your cutting point of knowledge"
        response = call_mistral_api(prompt)
        if response:
            print(f"Respuesta: {response}")
    elif args.type == "codestral":
        prompt = "def f("
        response = call_codestral_api(prompt, model="codestral-latest")
        if response:
            print(f"Respuesta: {response}")
```