

Appel de Fonction Mistral

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

load_dotenv()

def appeler_api_mistral(prompt, model="mistral-small-2501", use_function_calling=False):
    api_key = os.environ.get("MISTRAL_API_KEY")
    if not api_key:
        print("Erreur : la variable d'environnement MISTRAL_API_KEY n'est pas définie.")
        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": "Obtenir la météo actuelle à un emplacement donné",
                        "parameters": {
                            "type": "object",
```

```

        "properties": {
            "location": {
                "type": "string",
                "description": "La ville et l'état, par exemple 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 l'API Mistral : {url}")
print(f"En-têtes de l'API Mistral : {headers}")
print(f"Données de l'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"Appels d'outils : {tool_calls}")
        return tool_calls # Retourner les appels d'outils pour le traitement
    else:
        print(f"Erreur de l'API Mistral : Format de réponse invalide : {response_json}")
        return None
    else:
        print(f"Erreur de l'API Mistral : Format de réponse invalide : {response_json}")
        return None
except requests.exceptions.RequestException as e:
    print(f"Erreur de l'API Mistral : {e}")
    if e.response:
        print(f"Code de statut de la réponse : {e.response.status_code}")
        print(f"Contenu de la réponse : {e.response.text}")
    return None

def appeler_api_codestral(prompt, model="codestral-latest"):
    api_key = os.environ.get("MISTRAL_API_KEY")
    if not api_key:
        print("Erreur : la variable d'environnement MISTRAL_API_KEY n'est pas définie.")
        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": "retourner a + b",
        "max_tokens": 64,
        "temperature": 0
    }
    print(f"URL de l'API Codestral : {url}")
    print(f"En-têtes de l'API Codestral : {headers}")
    print(f"Données de l'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"Erreur de l'API Codestral : Format de réponse invalide : {response_json}")
    return None

except requests.exceptions.RequestException as e:
    print(f"Erreur de l'API Codestral : {e}")
    if e.response:
        print(f"Code de statut de la réponse : {e.response.status_code}")
        print(f"Contenu de la réponse : {e.response.text}")
    return None

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Tester les API Mistral et Codestral.")
    parser.add_argument("--type", type=str, default="mistral", choices=["mistral", "codestral"], help="Type d'API")
    parser.add_argument("--function_calling", action="store_true", help="Activer l'appel de fonction pour l'API")
    args = parser.parse_args()

    if args.type == "mistral":
        prompt = "Quel est le temps à Londres ?"
        response = appeler_api_mistral(prompt, use_function_calling=args.function_calling)
        if response:
            print(f"Réponse : {response}")
    elif args.type == "codestral":
        prompt = "def f("
        response = appeler_api_codestral(prompt, model="codestral-latest")
        if response:
            print(f"Réponse : {response}")

```