

Mensajes de Commit en Git Potenciados por IA

Este script de Python debe colocarse en un directorio incluido en el PATH de tu sistema, como ~/bin.

```
import subprocess
import os
from openai import OpenAI
from dotenv import load_dotenv
import argparse

load_dotenv()

def gitmessageai(push=True, only_message=False):
    # Etapa todos los cambios
    subprocess.run(["git", "add", "-A"], check=True)

    # Obtén el diff de los cambios en etapa
    diff_process = subprocess.run(["git", "diff", "--staged"], capture_output=True, text=True, check=True)
    diff = diff_process.stdout

    if not diff:
        print("No hay cambios para hacer commit.")
        return

    # Prepara el prompt para la IA
    prompt = f"""
    Genera un mensaje de commit conciso en formato Conventional Commits para los siguientes cambios de código.
    Usa uno de los siguientes tipos: feat, fix, docs, style, refactor, test, chore, perf, ci, build o revert.
    Si es aplicable, incluye un alcance entre paréntesis para describir la parte del código afectada.
    El mensaje de commit no debe exceder los 70 caracteres.

    Cambios de código:
```

```
{diff}
```

Mensaje de commit:

```
"""
```

```
# Envía el prompt a la API de DeepSeek
```

```
api_key = os.environ.get("DEEPSEEK_API_KEY")
```

```
if not api_key:
```

```
    print("Error: La variable de entorno DEEPSEEK_API_KEY no está configurada.")
```

```
    return
```

```
client = OpenAI(api_key=api_key, base_url="https://api.deepseek.com")
```

```
try:
```

```
    response = client.chat.completions.create(
```

```
        model="deepseek-chat",
```

```
        messages=[
```

```
            {"role": "user", "content": prompt}
```

```
        ],
```

```
        max_tokens=100
```

```
    )
```

```
    if response and response.choices:
```

```
        commit_message = response.choices[0].message.content.strip()
```

```
        commit_message = commit_message.replace('\n', '')
```

```
    else:
```

```
        print("Error: No hubo respuesta de la API.")
```

```
        return
```

```
except Exception as e:
```

```
    print(f"Error durante la llamada a la API: {e}")
```

```
    return
```

```
# Verifica si el mensaje de commit está vacío
```

```
if not commit_message:
```

```
    print("Error: Se generó un mensaje de commit vacío. Abortando el commit.")
```

```
    return
```

```

if only_message:
    print(f"Mensaje de commit sugerido: {commit_message}")
    return

# Haz commit con el mensaje generado
subprocess.run(["git", "commit", "-m", commit_message], check=True)

# Empuja los cambios
if push:
    subprocess.run(["git", "push"], check=True)
else:
    print("Cambios confirmados localmente, pero no empujados.")

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Genera un mensaje de commit con IA y confirma los cambios")
    parser.add_argument('--no-push', dest='push', action='store_false', help='Confirma los cambios localmente')
    parser.add_argument('--only-message', dest='only_message', action='store_true', help='Solo imprime el mensaje')
    args = parser.parse_args()
    gitmessageai(push=args.push, only_message=args.only_message)

```

Luego, en tu archivo ~/.zprofile, agrega lo siguiente:

```

alias gpa='python ~/bin/gitmessageai.py'
alias gca='python ~/bin/gitmessageai.py --no-push'
alias gm='python ~/bin/gitmessageai.py --only-message'

```