Universidad Santo Tomas de Aquino

Miguel Angel Jimenez Morales
Dikersson Alexis Cañon Vanegas

Introducción

En este informe se describe el desarrollo de un chatbot personalizado desde la terminal de Ubuntu 24.04, utilizando el editor nano, la librería requests de Python y la API de DeepSeek. El objetivo principal es demostrar el flujo completo de creación: desde la escritura del script hasta su despliegue en un repositorio remoto en GitHub.

Marco Teórico

Los chatbots son programas diseñados para simular una conversación humana mediante el procesamiento de lenguaje natural (PLN).

- Modelos basados en API: Plataformas tipo DeepSeek u OpenAI ofrecen endpoints REST que reciben un “prompt” y devuelven una respuesta generada por modelos de lenguaje profundo.

- Terminal en Linux: El uso de nano, git facilita un entorno ligero y reproducible para desarrolladores, sin interfaz gráfica.

- Control de versiones: Git permite rastrear cambios en el código, colaborar en equipo y desplegar versiones estables de forma organizada.

Proceso

Preparación del entorno

Instalación de Python 3, pip, nano y git con apt:

```
sudo apt update
sudo apt install python3 python3-pip nano git
mkdir TAREA2
cd TAREA2
nano chatbot.py
```

Creación del script

En nano chatbot.py se implementó un bucle de lectura/escritura que envía el texto del usuario al endpoint de DeepSeek y muestra la respuesta.

Instalación de dependencias

Ejecución de pip install requests para habilitar las llamadas HTTP al API.

Ejecución y pruebas

Lanzar python3 chatbot.py y validar interacción conversacional; manejo de la palabra clave salir para finalizar.

```
"Eres un asistente experto en telecomunicaciones que responde con entusiasmo y cálculos sencillos"
```

Versionado y despliegue

Inicialización de repositorio local con git init, commits semánticos, configuración del remoto en GitHub y git push.

```
git config --global user.name "Kersson12"
git config --global user.email "alexiscv686@gmail.com"
```

git clone "https://github.com/Kersson12/TAREA2.git"

cd TAREA2
“copiar y pegar los archivos README.md y chatbot.py a la carpeta”
git add README.md
git commit -m "Descripcion del proceso para hacer el chat bot"
git add chatbot.py
git commit -m "Codigo chatbot”
git push
“Usar usuario y el token”

Análisis

- **Robustez:** La implementación básica responde correctamente en la mayoría de entradas, aunque carece de manejo de errores avanzado (tiempos de espera, desconexiones).

- **Extensibilidad:** Fácil de extender añadiendo un system prompt inicial o almacenando historial de conversación.

- **Colaboración:** El uso de ramas (feature/...) y mensajes de commit claros permite integrar nuevas funcionalidades sin afectar la rama principal.

- **Reproducibilidad:** Documentar el flujo en README.md y seguir buenas prácticas de commits asegura que cualquier colaborador pueda clonar y poner en marcha el bot rápidamente.

Resultados

- Script funcional chatbot.py disponible en GitHub dentro de la carpeta TAREA2.

- Flujo de trabajo establecido: creación de ramas, commits descriptivos y pushes.

- Documentación: README.md con pasos de instalación, uso y personalización.

Conclusiones

Simplicidad y rapidez de implementación:
El uso de la terminal de Linux y el editor nano demostró ser una forma muy directa de crear y editar el script del chatbot sin dependencias de entornos gráficos. Esto permite un desarrollo ágil y apto para sistemas con pocos recursos.

Flexibilidad gracias a la API:
Al delegar la generación de lenguaje al servicio de DeepSeek mediante llamadas HTTP, el chatbot hereda toda la potencia de un modelo de lenguaje profundo sin necesidad de instalar librerías pesadas localmente. Solo fue necesario requests para comunicarse con la API.

Buenas prácticas de control de versiones:
Integrar Git desde el inicio —con commits claros, ramas de funcionalidad y configuración del remoto— asegura trazabilidad de cambios y facilita la colaboración. El establecer origin/main como rama upstream reduce la complejidad al publicar nuevas versiones.

Extensibilidad futura:
La estructura básica permite añadir con facilidad:

- Manejadores de errores y timeouts en las peticiones HTTP.

- Instrucciones de sistema (system prompt) para definir mejor la “personalidad” del bot.

- Almacenamiento de historial o integración con bases de datos para conversaciones contextuales.

Reproducibilidad y documentación:
La creación de un README.md y el anexo de los comandos utilizados garantizan que cualquier compañero pueda clonar el repositorio, instalar dependencias y ejecutar el chatbot sin ambigüedades.