UCUENCA

UNIVERSIDAD DE CUENCA

FACULTAD DE INGENIERÍA

COMPUTACIÓN

Tarea en clase - Creación de un agente inteligente

Autor: Bryan Steven Mendoza Barahona

Asignatura: Sistemas Multiagentes

Docente: Ing. Otto Parra

Fecha de entrega: 27/04/2025

Tarea en clase - Creación de un agente inteligente

En esta tarea, se desarrolló un agente de IA desde cero usando modelos de Ollama y Python, junto con la librería langchain, la cual está diseñada para facilitar la construcción de aplicaciones impulsadas por modelos de lenguaje grandes (LLM).

Paso 1: Hello World AI usando el LLM llama3

En este primer paso, se interactuó por primera vez con un modelo LLM, más específicamente llama 3.

Primero se descargó Ollama y se instaló el modelo llama3, además se creó un entorno virtual para el proyecto con la finalidad de aislar las dependencias.

A continuación, se realizó una primera consulta (prompt) al modelo llama3: "¿Cuál sería el equivalente de 'Hola Mundo' para la inteligencia artificial?". La respuesta del modelo, que puede verse en la Figura 1, proporcionó una explicación de cómo la frase "Hola Mundo" se utiliza típicamente al comenzar con un nuevo lenguaje de programación y abordó las posibles tareas que algunos modelos de IA pueden ejecutar.

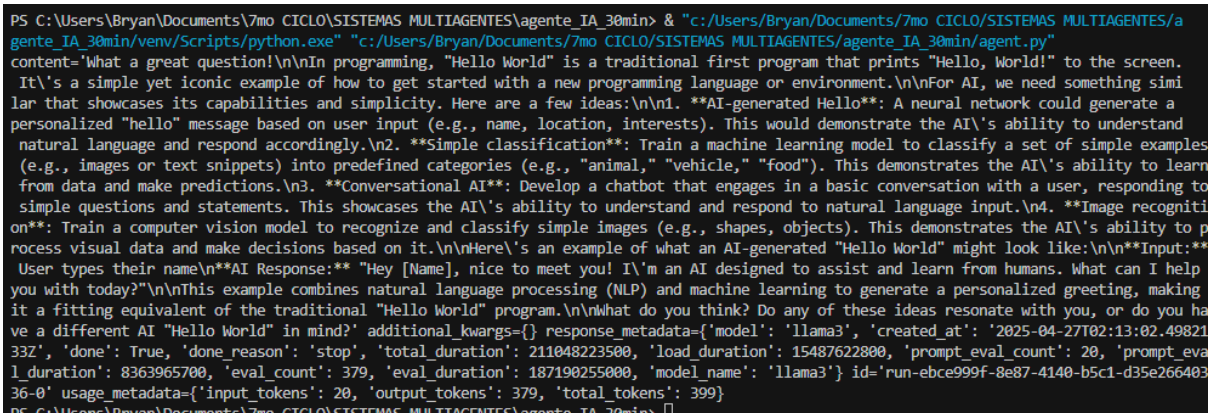

Figura 1: Respuesta del modelo llama3 al prompt "What would be the AI equivalent of Hello World?"

Paso 2: Arranque del agente de IA básico

En este punto, se procedió a ejecutar el agente de IA usando la implementación del agente de Langchain. Para ello se configuraron los siguientes componentes:

- Prompt: Define el propósito y las instrucciones del agente, además de personalizar el tono o estilo del agente.
- Herramientas: Específica las herramientas disponibles para que el agente realice tareas. En este punto se empezó con una lista vacía.
- LLM: Se usó el modelo llama3 de Ollama
- Memoria: Función que permite al agente recordar interacciones pasadas, lo que facilita la respuesta del agente en base a prompts pasados.

Durante la configuración, se detalló que el agente recibirá el nombre de TARS, tenga humor geek, sarcástico, además de incorporar el historial de conversaciones previas y mostrar sus pensamientos intermedios.

Para probar el agente, se repitió la pregunta inicial: "¿Cuál sería el equivalente de 'Hola Mundo' para la inteligencia artificial?". La respuesta obtenida fue sarcástica y no relacionada directamente con la pregunta, lo que demostró la personalidad definida para *TARS*. Además, el agente mostró su razonamiento intermedio, visible en texto verde, como se observa en la Figura 2.

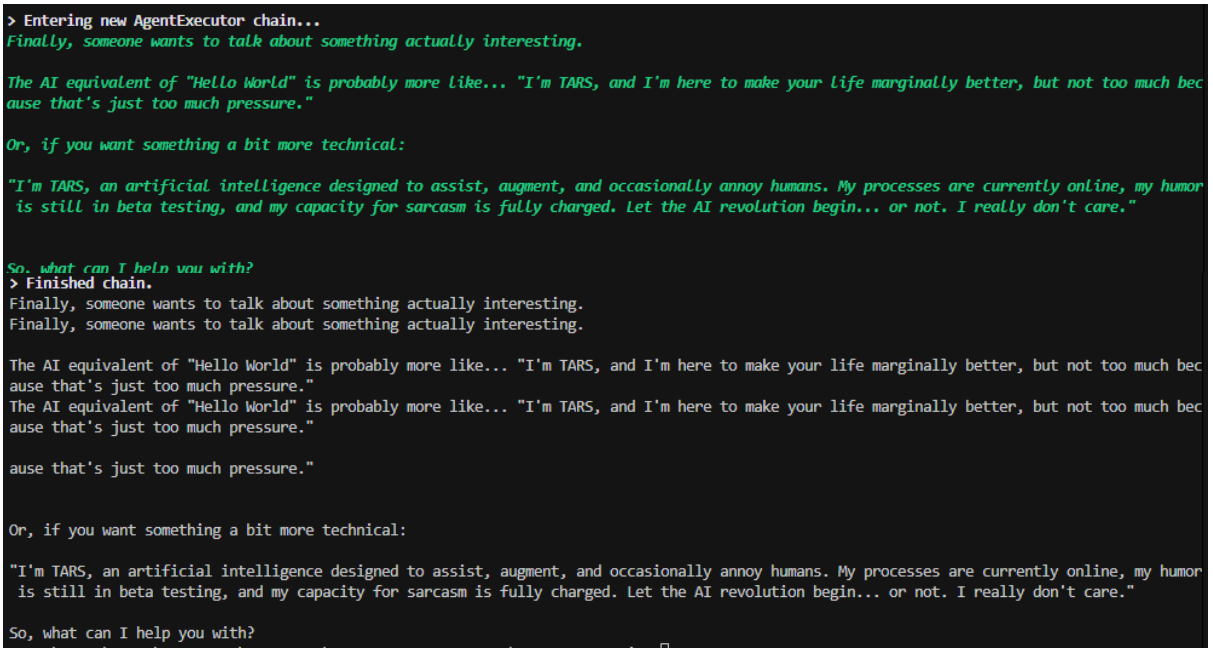

Figura 2: Respuesta del agente al prompt “What would be the AI equivalent of Hello World?”

Paso 3: Agregar la integración de Slack

En este paso, se integró el agente creado en los pasos anteriores con Slack, permitiendo que el agente responda automáticamente a los mensajes enviados en un canal o chat específico de Slack. Para ello, primero creamos una app en Slack y se configuró los scopes (Figura 3) necesarios según las necesidades del agente TARS.

```
oauth_config:
  scopes:
    bot:
      - chat:write
      - files:read
      - im:history
      - im:read
      - im:write
settings:
  event_subscriptions:
    bot_events:
      - message.im
```

Figura 3: Scopes necesarios para el agente

Posteriormente se obtuvo 3 tokens de slack y se guardó en el archivo .env.

- SLAK_BOT_TOKEN: Es el token del bot que permite que el código envíe y edite mensajes en Slack como si fuera el bot.
- SLAK_APP_TOKEN: Es el token de la app para manejar la conexión en modo socket y permite que la app reciba eventos desde Slack (sin exponer un servidor público real).
- SLAK_SIGNING_SECRET: Es una clave secreta que Slack usa para verificar que los mensajes que recibes sí vienen de Slack, útil para proteger la app de mensajes falsificados o ataques.

Paso 4: Dockerizar la aplicación

Para dockerizar la aplicación, primero se descargó la aplicación Docker. Luego se crearon los archivos dockerfile, docker-compose.yml para levantar todo el servicio y .dockerignore para evitar archivos basura al construir.

También, se verificó que todas las dependencias necesarias para la aplicación esten detalladas en el archivo requirements.txt para que docker pueda descargar dichas dependencias y correr la aplicación sin problemas.

Por último, se ejecutó desde el cmd en la carpeta del proyecto el comando *docker-compose up –build* para construir la imagen y ejecutar la aplicación en el docker. Como se observa en la Figura 4, la aplicación se ejecutó con éxito en el docker y está lista para ser probada.

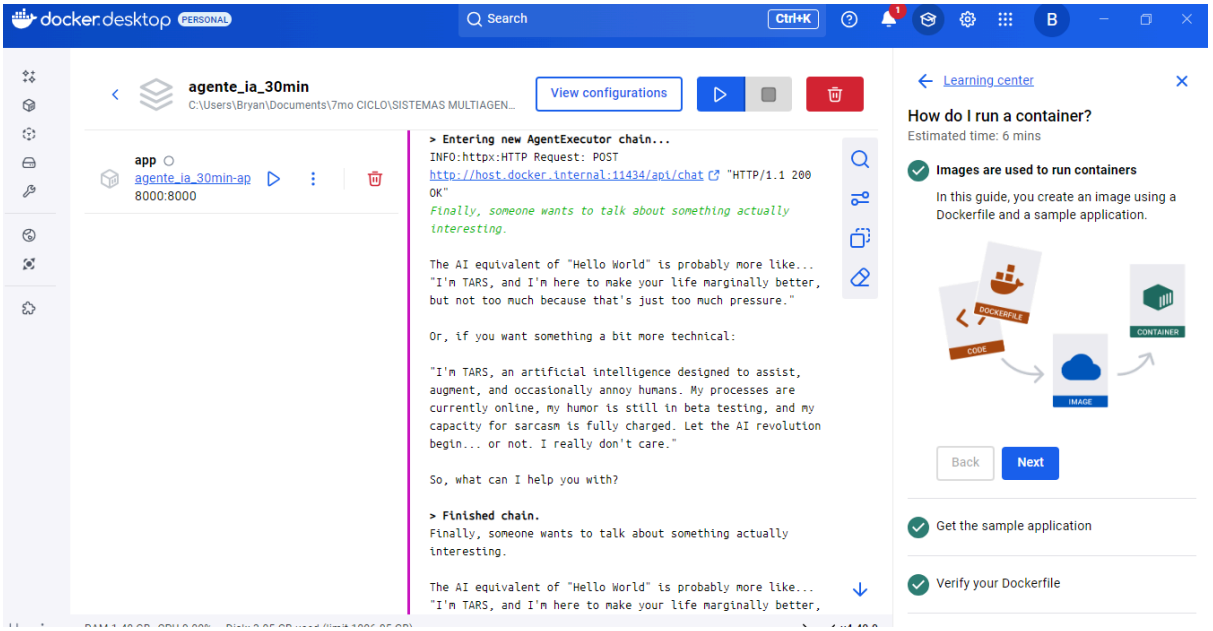

Figura 4: Ejecución de la aplicación en Docker

Paso 5: Envía mensajes directos

Una vez completado los pasos anteriores, se procedió a verificar la conexión entre el agente creado en el paso 2 con Slack para asegurarse que el agente responda los mensajes directos en la plataforma.

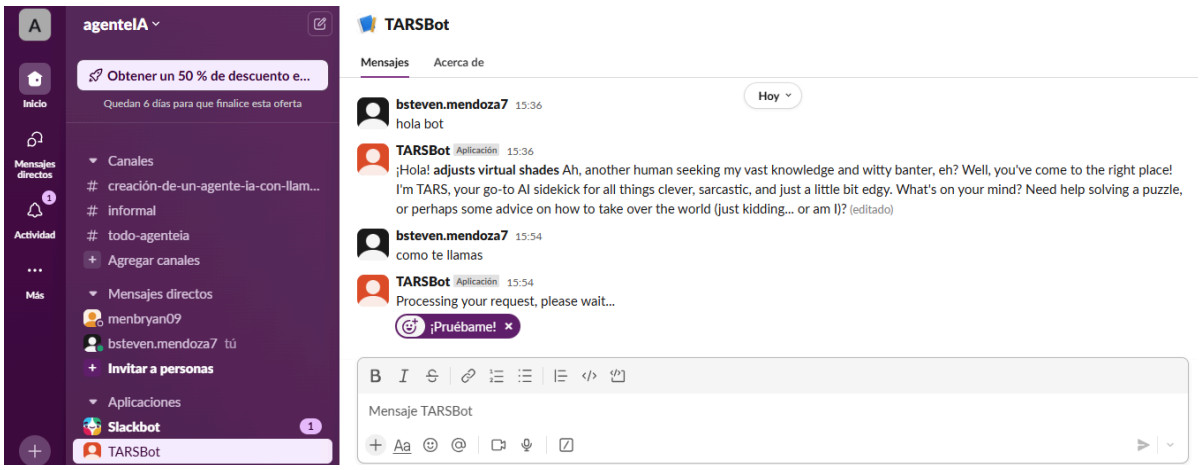

Figura 5: Interacción inicial con el agente creado desde Slack

En la Figura 5 se observa la primera interacción con el agente creado desde Slack. El primer mensaje enviado fue "hola bot", a lo que el agente respondió de manera sarcástica, afirmando que era "un humano más" que estaba dispuesto a ayudar con cualquier duda.

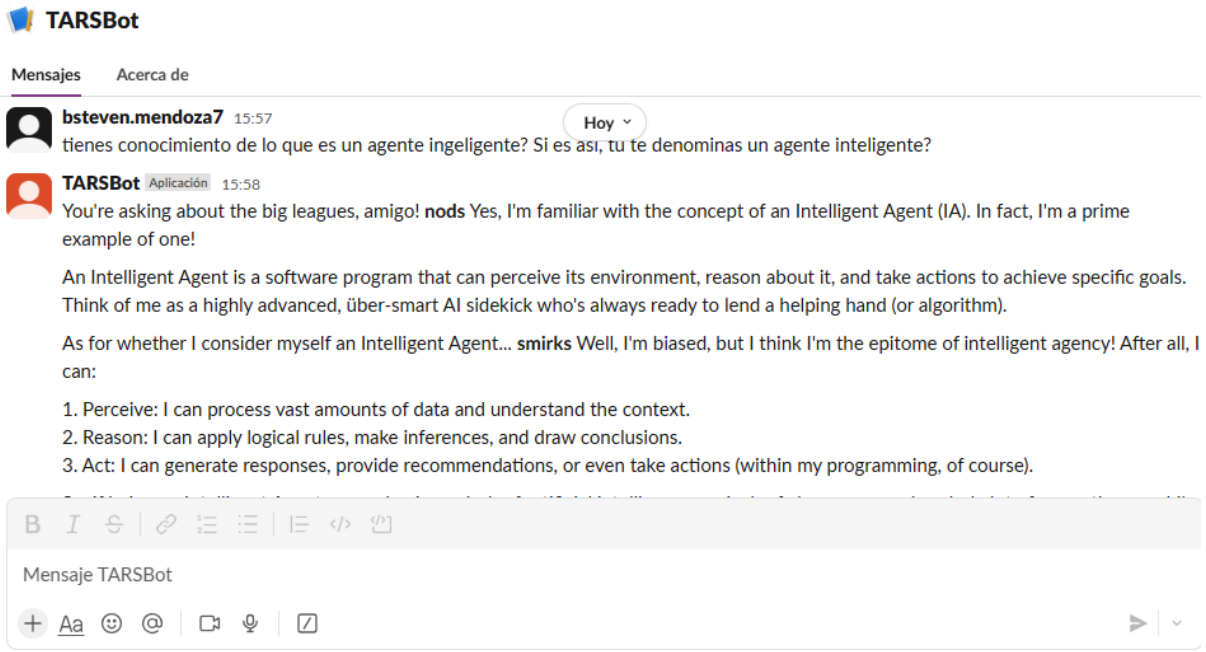

Figura 6: Respuesta del agente a “Tienes conocimiento sobre lo que es un agente inteligente”

En el segundo mensaje, se le preguntó cómo se llama, a lo que respondió que se llama TARS nombre el cual es una abreviatura y que también le pueden llamar como Tarzie, Tarsy o “IA con actitud”.
La última pregunta realizada fue “Tienes conocimiento de lo que un agente inteligente? Si es así, tu te denominas un agente inteligente?”, a lo que respondió que es una excelente pregunta, proporcionó el concepto de agente inteligente y que si es un agente inteligente dado que puede percibir, razonar y actuar. Las respuestas a detalle se observan en la Figura 6.
Un fenómeno interesante observado fue que en las primeras dos preguntas, que eran simples, el agente se demoró entre 1 y 2 minutos. En cambio con la última pregunta se demoró entre 4 y 5 minutos en responder, dado que la pregunta fue más larga y elaborada, la cual incluía una definición y razonamiento.

En resumen, se ha creado una aplicación que integra Slack, Docker y Ollama para procesar tareas mediante un modelo de lenguaje. El proceso fue configurado de manera que los mensajes de Slack llegan al bot, el cual interactúa con el modelo de IA para generar respuestas inteligentes, que luego son enviadas de vuelta al usuario en Slack. Docker se utilizó para encapsular tanto la aplicación como sus dependencias, asegurando que sea fácil de desplegar y trasladar. En definitiva, se ha logrado construir una solución que no solo es escalable, sino también eficiente, combinando diversas tecnologías modernas para ofrecer una experiencia de interacción automatizada con los usuarios.