Tnlga. DAYANA ESTEFANIA GRANDA RIVAS

Septiembre 2024

ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACION DE SERVICIOS DE CÓMPUTO Y DE SEGURIDAD

IMPLEMENTACIÓN DE HERRAMIENTA DE INTELIGENCIA ARTIFICIAL DE FORMA LOCAL Y UNA INTERFAZ GRAFICA WEB

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN REDES Y TELECOMUNICACIONES

DAYANA ESTEFANIA GRANDA RIVAS
dayana.granda@epn.edu.ec

DIRECTOR: FERNARDO VINICIO BECERRA CAMACHO
fernando.becerra@epn.edu.ec

Quito, Septiembre 2024

CERTIFICACIONES

Yo, DAYANA ESTEFANIA GRANDA RIVAS declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

DAYANA ESTEFANIA GRANDA RIVAS

dayana.granda@epn.edu.ec

degr180500@gmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por DAYANA ESTEFANIA GRANDA RIVAS, bajo mi supervisión.

FERNANDO VINICIO BECERRA CAMACHO

DIRECTOR

fernando.becerrac@epn.edu.ec

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

DAYANA ESTEFANIA GRANDA RIVAS

AGRADECIMIENTO

Agradezco a Dios por cada día permitirme superar los obstáculos que trae la vida misma, por los dones, habilidades y las capacidades que ha puesto en mi para salir adelante y culminar esta importante etapa.

A mi familia, mis padres y hermanos, que han sido mi apoyo incondicional y mi inspiración para seguir mis sueños. Sepan que ustedes son mi mayor tesoro.

Al Ing. Fernando Becerra, gracias por su paciencia, sabiduría y ayuda en mi proceso de titulación.

A mis profesores y autoridades de la Escuela Politécnica Nacional por la comprensión y apoyo en mi actual situación de salud. Gracias por su compromiso con los estudiantes y su superación profesional.

ÍNDICE DE CONTENIDOS

RESUMEN

Este proyecto tiene como objetivo principal la implementación de un sistema operativo Linux, específicamente Ubuntu, para ejecutar modelos de inteligencia artificial localmente con Ollama. A través de esta iniciativa, se pretende desarrollar un entorno robusto y eficiente que permita el procesamiento y ejecución de modelos de lenguaje, facilitando así la realización de cuestionarios y otras tareas relacionadas con la inteligencia artificial.

El marco teórico del proyecto aborda temas fundamentales como el sistema operativo Linux, inteligencia artificial, y las herramientas utilizadas como Docker y DevOps. Estos conceptos son esenciales para entender la base sobre la cual se construye todo el sistema.

La metodología empleada en el proyecto incluye la configuración inicial de la máquina virtual de Ubuntu, la descarga y configuración de Ollama, y la implementación de Docker y DevOps. Se describe detalladamente cómo crear archivos docker-compose.yaml y main.py para definir y construir los servicios del contenedor, así como la forma de acceder al sistema de manera local.

Los resultados obtenidos demuestran la eficacia del sistema implementado. Los usuarios pueden ingresar al sistema mediante enlaces web locales y realizar preguntas sobre diversos temas, aprovechando el procesamiento de modelos de lenguaje para obtener respuestas precisas y rápidas.

El proyecto concluye con una evaluación del éxito de la implementación, destacando las ventajas y beneficios de utilizar un entorno Linux para la ejecución de modelos de inteligencia artificial. Además, se ofrecen recomendaciones para futuros trabajos y posibles mejoras en el sistema, asegurando su evolución y adaptación a nuevas necesidades y tecnologías.

PALABRAS CLAVE: inteligencia artificial, código abierto, sistema operativo, Linux, Ollama, máquina virtual, docker, DevOps, contenedores,

ABSTRACT

This project primarily aims to implement a Linux operating system, specifically Ubuntu, to locally run artificial intelligence models using Ollama. The initiative seeks to develop a robust and efficient environment that enables the processing and execution of language models, facilitating the completion of questionnaires and other AI-related tasks.

The project's theoretical framework addresses fundamental topics such as the Linux operating system, artificial intelligence, and the tools used like Docker and DevOps. These concepts are essential to understand the foundation upon which the entire system is built.

The methodology employed in the project includes the initial configuration of the Ubuntu virtual machine, the download and setup of Ollama, and the implementation of Docker and DevOps. Detailed descriptions are provided on how to create docker-compose.yaml and main.py files to define and build container services, as well as how to access the system locally.

The obtained results demonstrate the effectiveness of the implemented system. Users can access the system via local web links and ask questions on various topics, leveraging the processing of language models to receive precise and quick answers.

The project concludes with an evaluation of the implementation's success, highlighting the advantages and benefits of using a Linux environment for running artificial intelligence models. Additionally, recommendations for future work and potential system improvements are offered, ensuring its evolution and adaptation to new needs and technologies.

KEYWORDS: *artificial intelligence, open source, operating system, Linux, Ollama, virtual machine, docker, DevOps, containers*

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

En la era digital actual, la eficiencia y la agilidad en el desarrollo y despliegue de aplicaciones son factores cruciales para el éxito de cualquier organización. La metodología DevOps y el uso de contenedores Docker se han destacado como soluciones efectivas para mejorar estos aspectos. Este proyecto de titulación tiene como objetivo implementar un sistema de gestión local utilizando prácticas de DevOps y contenedores Docker para optimizar el ciclo de vida del desarrollo de software, desde la codificación hasta el despliegue en producción.

La implementación de DevOps implica la integración continua (CI) y la entrega continua (CD), permitiendo a los equipos de desarrollo y operaciones trabajar en conjunto para automatizar y estandarizar los procesos. Docker, por su parte, proporciona una plataforma para encapsular aplicaciones y sus dependencias en contenedores, asegurando que funcionen de manera consistente en cualquier entorno. Al combinar estas dos tecnologías, se busca crear un entorno de desarrollo ágil y robusto que reduzca el tiempo de despliegue y minimice los errores.

1.1 Objetivo general

La idea del componente implica establecer una conexión con una máquina virtual desde un sistema operativo Ubuntu para llevar a cabo una conversación con Ollama. Es crucial desarrollar una interfaz gráfica para facilitar la comunicación entre el usuario y su computadora, permitiendo la navegación en red. Además, se requiere acceder a la página web de forma local para crear archivos YAML, en este tipo de ficheros u otros lenguajes de programación se debe enfatizar una idea dentro de todos los datos que se ingresan, se especifican los puertos y otras configuraciones. Al incluir la dirección, se puede acceder a internet para completar un cuestionario o simplemente interactuar. Es fundamental implementar un entorno Docker para una transferencia más fluida de código y aprovechar todos los recursos disponibles en DevOps, que se encarga del mantenimiento de software.

1.2 Objetivos específicos

Los objetivos específicos detallan los procesos necesarios para la completa realización del componente; sirven como una guía de la manera en la que será abordado el componente asignado.

- Al implementar DevOps para realizar la interfaz gráfica el ciclo del desarrollo se puede avanzar, se puede integrar, entregar lo que significa desplegar y avanzar continuamente.
- Desarrollar una interfaz web para la plataforma de Ollama, buscan escalabilidad, que se pueden desarrollar y desplegar la tecnología, se busca mejorar la interacción usuario-sistema para producir destrezas eficientes.
- Facilitar la colaboración y comunicación entre equipos de desarrollo, operaciones y calidad al implementar una metodología DevOps, lo que permite una entrega más rápida y confiable de la interfaz gráfica.
- Se verifica la implementación de la plataforma para que no pueda existir problemas.

1.3 Alcance

El presente trabajo de titulación se centrará en la implementación de un sistema de gestión local utilizando Ollama, una herramienta avanzada de inteligencia artificial, y Docker, una plataforma de contenedores que facilita la implementación y gestión de aplicaciones en entornos aislados. El objetivo principal es demostrar cómo estas tecnologías pueden integrarse para crear una solución robusta y eficiente, optimizando el ciclo de vida del desarrollo de software, desde la codificación hasta el despliegue en producción. Se realizará una investigación exhaustiva sobre Ollama y Docker, evaluando sus características, ventajas y desventajas, para seleccionar las versiones más adecuadas.

El proyecto de titulación incluirá la actualización y configuración del sistema operativo, así como la instalación de dependencias necesarias como Git, Docker, Docker Compose, Node.js y npm. Se clonará el repositorio de Ollama desde GitHub, se configurará una interfaz gráfica. Se definirán y configurarán archivos Dockerfile y docker-compose.yml para la gestión de contenedores. Posteriormente, se construirá la imagen Docker y se ejecutará la aplicación Ollama en un contenedor, verificando su correcto funcionamiento a través de pruebas exhaustivas [1].

1.4 Marco Teórico

Primera Parte del marco teórico

La inteligencia artificial (IA) es la capacidad de las máquinas para simular procesos de pensamiento humano. Su desarrollo se remonta al siglo XX, época en la que se forjaron

las primeras ideas sobre seres artificiales. Alan Turing, pionero en este campo, sentó las bases teóricas en matemáticas para explorar sus posibilidades. Durante los años 1940 y 1950, se crearon las primeras redes neuronales artificiales, inspiradas en modelos biológicos. En la década de 1950, Turing y otros matemáticos debatieron sobre la capacidad de las máquinas para mostrar inteligencia. Los avances en computación y algoritmos en las décadas de 1980 y 1990 impulsaron el progreso de la IA. En el año 2010, el enfoque en el aprendizaje autónomo permitió avances significativos en el reconocimiento de voz y visión, proporcionando datos cruciales para su evolución. [2]

Funcionamiento de la inteligencia artificial

La inteligencia artificial busca equipararse con la inteligencia humana en aspectos como el razonamiento, la comprensión, el análisis de datos y la toma de decisiones coherentes. La incorporación de herramientas como Ollama permite obtener respuestas a una amplia gama de preguntas que planteamos. Es fundamental enfocarse en el desarrollo de sistemas y programas que utilicen estas capacidades para mejorar la eficiencia y la precisión en diversas tareas. Los chatbots, por ejemplo, representan un ejemplo de aplicación práctica de la inteligencia artificial, actuando como sistemas locales que ofrecen respuestas contextualizadas a través de una interacción conversacional. [3]

Redes que son parte de la inteligencia artificial

Red remota

Los servidores remotos, ubicados en la nube, constituyen la infraestructura de una red remota. Estos servidores requieren conexión a internet para posibilitar el acceso de los usuarios. Acceden a recursos y servicios donde pueden estar conectados. Es fundamental expandir esta red de manera global para garantizar un alcance óptimo y una conectividad eficiente. Las redes públicas viajan sus datos donde es preocupar en su seguridad ya que se utiliza cifrado de datos y busca toda la información sensible para proteger. Permiten a todos los usuarios entrar a sus datos compartidos, se busca conectar a internet donde siempre se debe autorizar de manera adecuada. Se ha reducido el costo de los servicios para poder acceder mediante múltiples usuarios para poder llegar solo en ubicaciones remotas accesibles, se debe escalar de manera fácil para poder tener número de dispositivos y usuarios para evitar cambios en su infraestructura. El mantenimiento y administración de su red debe ser centralizada y de manera consistente con sus políticas de seguridad y actualizaciones de software. Acceden a empleados de manera segura donde pueden trabajar desde su casa o

lugares fuera de su oficina principal o donde se puede acceder a sus recursos corporativos. Ejemplo:

Una red remota se trata de las redes privadas virtuales, acceden a través de una red que se encuentra conectada a su red, se conecta a través de internet para tener una conexión avanzada [4].

Red local

Una red local se emplea para operar en dispositivos dentro de un entorno cercano, lo que incluye la implementación de dispositivos de Internet de las cosas (IoT) y el uso de interfaces informáticas. Estas redes no requieren conexión externa, lo que garantiza bajos tiempos de respuesta y una mayor protección de la privacidad de los datos, todos los dispositivos se conectan como laptops, servidores y dispositivos móviles, además se usan cables de red como fibra óptica y Ethernet, se usan redes inalámbricas y dispositivos para tener comunicación entre los usuarios quien se conectan a la red. Su tipo de topología se menciona en forma estrella, anillo o en malla (conexión punto a punto). Para poder intercambiar datos se utiliza como TCP/IP o UDP, para poder tener reglas y normas dentro de la comunicación. Se usan métodos de seguridad para todos los datos de la red para transmitir. Ejemplo: [5]

Ollama ofrece una conexión versátil con una amplia gama de modelos y lenguajes de inteligencia artificial de código abierto, incluso en modo offline, siendo ideal para integrarse con redes locales, un enfoque más tradicional. Esta herramienta aprovecha las ventajas de implementar diversos modelos, priorizando la facilidad de uso y la utilidad para los usuarios. Además, garantiza la privacidad de los datos y no implica costos adicionales.

Con Ollama, los usuarios pueden aprender de forma automática y avanzar en la configuración de sus equipos, añadiendo nuevos modelos y personalizando su experiencia. Facilita la intervención en el código y el uso de la plataforma Llama2 para una integración más completa, han sido publicados por Meta AI, puede tener datos que ejercen todas las interrogantes, se puede nomas investigar de manera gratuita, puede utilizar fuentes como Wikipedia, YouTube y Twitter para generar respuestas o tareas avanzadas de alta calidad, para emplear su modelo se debe usar API, CLI y SDK. [6]

DevOps y docker para realizar interfaces graficas

Docker Compose es una herramienta fundamental para ejecutar aplicaciones compuestas por múltiples contenedores de manera automatizada. Permite definir redes,

servicios y volúmenes en un archivo docker-compose.yml, facilitando la configuración y gestión de la aplicación. Esta tecnología encapsula la aplicación y aísla su entorno, lo que mejora significativamente el desarrollo, el despliegue de la plataforma y la ejecución en diferentes sistemas operativos. Además, fomenta la colaboración entre equipos de desarrollo y operaciones, promoviendo así las prácticas de DevOps para una integración continua y una entrega más eficiente. [7]

2 METODOLOGÍA

En primera instancia, se realizó una investigación exhaustiva sobre el manejo de herramientas de inteligencia artificial de forma local y la creación de una interfaz gráfica, ya que esto fue fundamental para desarrollar el trabajo de titulación. Se diseñó una solución específica para cada servicio de cómputo utilizando diversas herramientas especializadas. Una vez que se tuvo claro el panorama de la implementación, se procedió a elaborar la solución de la herramienta de inteligencia artificial local con una interfaz web.

Se implementaron las soluciones utilizando herramientas específicas para el despliegue de los servicios de cómputo. Posteriormente, se llevó a cabo la implementación de la solución completa, que incluía la herramienta de inteligencia artificial con la interfaz gráfica. Se verificó el funcionamiento de cada servicio de cómputo para asegurar su operatividad.

Se realizaron pruebas exhaustivas de la implementación de la solución y se verificó su correcto funcionamiento, asegurando que todos los componentes operaran de manera eficiente y sin errores.

3 RESULTADOS

En este informe se presentarán los resultados obtenidos durante el desarrollo del trabajo de titulación, el cual se centró en la implementación de un sistema de gestión local mediante DevOps y contenedores Docker. Se detallarán los cuatro objetivos principales alcanzados: la investigación y selección de herramientas de inteligencia artificial y interfaces gráficas, el diseño de soluciones específicas para cada servicio de cómputo, la implementación de dichas soluciones, y la verificación y prueba del correcto funcionamiento del sistema completo. Cada uno de estos objetivos será descrito en detalle para proporcionar una comprensión completa de los logros y avances realizados a lo largo del proyecto.

3.1 Analizar soluciones a cada servicio de cómputo

La implementación de herramientas de inteligencia artificial (IA) a nivel local con una interfaz gráfica accesible es un aspecto crucial para el desarrollo eficiente y efectivo de aplicaciones de IA. Este trabajo de titulación se centra en la investigación de Ollama, una herramienta de IA que se destaca por su facilidad de uso, robustez y capacidades avanzadas. Compararemos Ollama con otras herramientas populares como TensorFlow, con el objetivo de demostrar por qué Ollama es la opción ideal para desarrollar este proyecto de titulación.

Ollama

Ollama es una herramienta innovadora diseñada para facilitar la creación, entrenamiento y despliegue de modelos de IA localmente. Se distingue por su interfaz gráfica intuitiva y su capacidad para operar sin necesidad de una infraestructura en la nube, lo que mejora la privacidad y seguridad de los datos. A continuación, se detallan algunas de sus características clave:

- Interfaz Gráfica de Usuario (GUI): Ollama proporciona una GUI que simplifica la interacción con los modelos de IA. Los usuarios pueden visualizar datos, ajustar hiperparámetros y monitorizar el rendimiento del modelo en tiempo real sin necesidad de codificación avanzada.
- Despliegue Local: Optimizada para funcionar en entornos locales, Ollama reduce la dependencia de la infraestructura en la nube, ofreciendo una mayor control sobre los datos y el proceso de desarrollo.
- Facilidad de Uso: Con un enfoque en la usabilidad, Ollama permite a los desarrolladores de todos los niveles construir y desplegar modelos complejos con mínima fricción. [8].

TensorFlow

Para comprender mejor las ventajas de Ollama, es útil compararla con TensorFlow, una de las bibliotecas de IA más populares y robustas.

- Flexibilidad y Potencia: TensorFlow es una biblioteca de código abierto desarrollada por Google que permite la construcción y entrenamiento de modelos de IA avanzados. Soporta tanto operaciones en CPU como en GPU, lo que lo hace adecuado para implementaciones locales y en la nube.
- Interfaz de Programación: Aunque TensorFlow es extremadamente poderoso, su uso a menudo requiere una sólida comprensión de la programación y el

manejo de gráficos computacionales. La curva de aprendizaje puede ser empinada para los principiantes.

- Herramientas de Visualización: TensorFlow incluye TensorBoard, una herramienta de visualización que facilita la depuración y optimización de modelos. Sin embargo, la configuración y uso de TensorBoard pueden resultar complejos para los usuarios menos experimentados.

De los dos ejemplos presentados, se optará por utilizar Ollama debido a las siguientes ventajas:

- Simplicidad y Accesibilidad: A diferencia de TensorFlow, Ollama está diseñada para ser accesible para desarrolladores de todos los niveles. Su GUI permite una interacción intuitiva con los modelos de IA, eliminando la necesidad de una codificación extensa.
- Despliegue Local y Seguridad: Ollama está optimizada para despliegues locales, lo que mejora la seguridad y privacidad de los datos. Los usuarios tienen control total sobre su entorno de desarrollo y los datos que utilizan.
- Visualización y Monitoreo en Tiempo Real: La interfaz gráfica de Ollama permite la visualización y el monitoreo del rendimiento del modelo en tiempo real, simplificando la tarea de ajustar y optimizar los modelos de IA [9].

De lo antes mencionado se puede observar en la Tabla 3.1:

Tabla 3.1 Comparación entre herramientas de AI

Característica	**TensorFlow**	**Ollama**
Facilidad de Uso	Requiere conocimientos avanzados	Interfaz amigable y fácil de usar
Configuración Inicial	Compleja, múltiples dependencias	Sencilla, menos dependencias
Despliegue	Puede ser complejo, requiere configuración manual	Simplificado con Docker y DevOps
Coste de Implementación	Variable, puede ser alto dependiendo del uso	Menor coste debido a la simplicidad y menos recursos necesarios
Interoperabilidad	Compatible con varias plataformas	Alta interoperabilidad, fácil integración

Requerimientos de Hardware	Puede requerir hardware potente	Funciona eficientemente en hardware moderado
Soporte Técnico	Amplio soporte de comunidad y documentación	Soporte específico, comunidad en crecimiento
Versatilidad en Modelos	Amplia variedad de modelos preentrenados	Modelos específicos y optimizados
Ecosistema de Herramientas	Extenso, muchas herramientas complementarias	Integración directa con herramientas necesarias
Rendimiento	Alto rendimiento en hardware optimizado	Buen rendimiento en hardware moderado

3.2 Diseñar la solución para cada servicio de cómputo mediante herramientas

Antes de comenzar con la instalación y configuración de Ollama, es importante preparar el entorno de trabajo. Esto incluye la instalación de dependencias y herramientas necesarias, como Docker, Docker Compose y Node.js. Primero, se debe actualizar el sistema operativo y sus paquetes para asegurar que todo esté al día. Luego, se instalarán Git para clonar repositorios, Docker y Docker Compose para gestionar los contenedores, y Node.js y npm para manejar las dependencias del proyecto de la interfaz gráfica. Además, es necesario configurar Docker para que el usuario pueda ejecutar comandos Docker sin necesidad de permisos de administrador.

El siguiente paso es clonar el repositorio de Ollama desde GitHub y navegar al directorio del proyecto clonado. Esto proporcionará acceso a todos los archivos y recursos necesarios para configurar y ejecutar Ollama localmente.

Para proporcionar una interfaz gráfica a Ollama primero se instalarán las herramientas necesarias para el desarrollo de la interfaz gráfica. Luego, se crearán los archivos y estructura necesarios para la aplicación de la interfaz gráfica, incluyendo el archivo principal de la aplicación y un archivo HTML básico que sirva como punto de entrada. Finalmente, se configurará el entorno de desarrollo, incluyendo Webpack si se utiliza React o Angular, para compilar y gestionar los recursos del proyecto.

Para ejecutar Ollama en contenedores, se creará un archivo Dockerfile que definirá la imagen del contenedor, especificando el entorno de ejecución y las dependencias

necesarias. También se creará un archivo docker-compose.yml para orquestar los servicios del contenedor, definiendo cómo se construirán y ejecutarán los contenedores Docker.

Una vez configurados los archivos de Docker, se procederá a construir la imagen Docker utilizando el Dockerfile. Posteriormente, se ejecutará la aplicación Ollama en un contenedor Docker utilizando Docker Compose. Este paso asegurará que la aplicación se esté ejecutando correctamente y sea accesible localmente [10].

El siguiente paso es acceder a la aplicación a través de un navegador web para asegurarse de que la interfaz gráfica de Ollama está funcionando correctamente. Se realizarán pruebas funcionales para verificar que todas las características de Ollama están operativas, asegurando que la integración de la IA con la interfaz gráfica funciona adecuadamente y que los datos y resultados se muestran correctamente y en tiempo real. Se configurarán herramientas de monitoreo y logging para supervisar el rendimiento del contenedor y la aplicación, capturando y analizando cualquier error o comportamiento inesperado [11].

3.3 Implementar las soluciones mediante herramientas para el despliegue de los servicios de Computo

Para la implementación del trabajo de titulación, se utilizó la herramienta de inteligencia artificial Ollama de forma local debido a las características destacadas en el capítulo anterior. En esta sección, se presentará detalladamente todo el proceso de implementación realizado en el trabajo de titulación. Se incluirán los pasos específicos, las configuraciones necesarias y las pruebas efectuadas para asegurar el correcto funcionamiento de la herramienta. Además, se discutirán los desafíos encontrados y las soluciones adoptadas durante la implementación, proporcionando una visión completa y comprensiva del desarrollo de esta herramienta de inteligencia artificial en un entorno local.

En primera instancia, se instaló un sistema operativo Ubuntu 22.04 debido a la estabilidad demostrada en trabajos de titulación previos presentados durante la carrera. Se asignaron un total de 4 GB de memoria RAM y se instaló en un servidor proporcionado por la ESFOT para asegurar la correcta implementación del proyecto, ya que había limitaciones para realizarlo en la computadora personal. Posteriormente, se utilizó una VPN para conectarse de forma remota, garantizando así la seguridad y evitando vulnerabilidades dentro de la red universitaria. En la Figura 3.1 que se presenta

a continuación, se puede observar la configuración y ejecución de la máquina virtual utilizada para esta implementación.

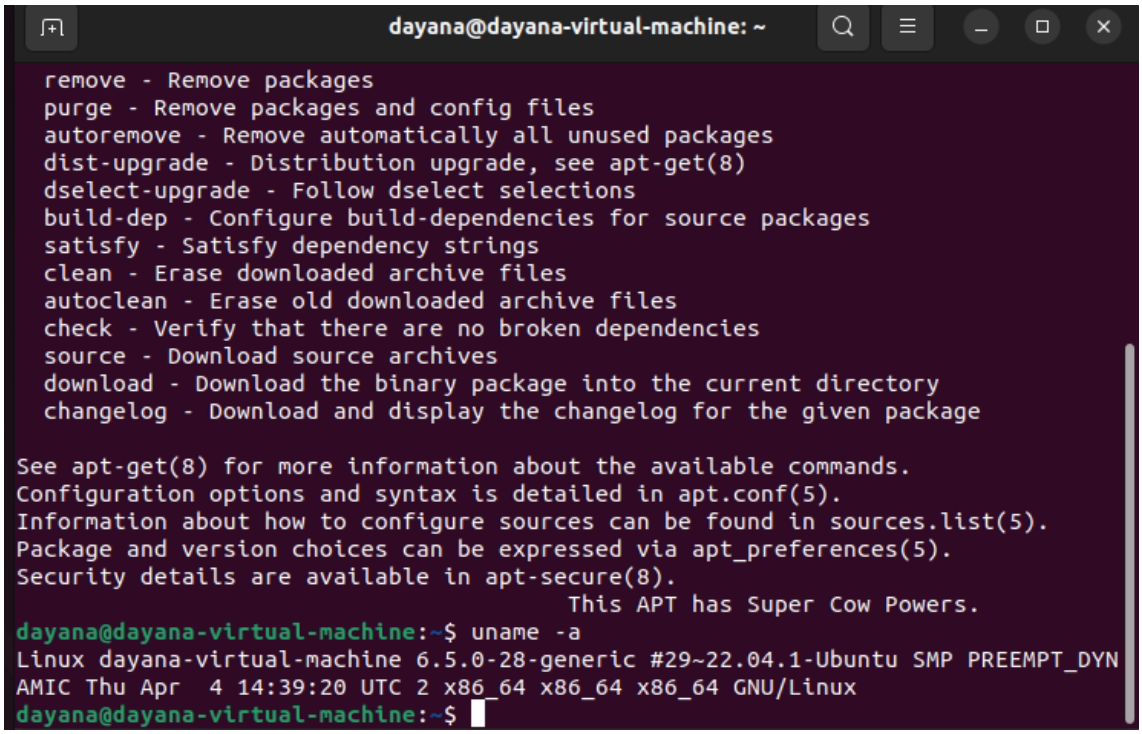

Figura 3.1 Instalación de la maquina virtual

Adicionalmente, se llevaron a cabo varias pruebas de rendimiento y estabilidad para asegurar que el sistema operativo y la infraestructura elegida soportaran las demandas del proyecto. Se configuraron diferentes servicios necesarios para el funcionamiento de Ollama, y se documentaron las configuraciones y parámetros clave para replicar el entorno de desarrollo en caso de futuras necesidades. Además, se tiene que mencionar que se actualizo toda la máquina virtual para no tener problemas de seguridad y que los paquetes se los pueda utilizar de una manera óptima.

Para la instalación del servicio de Ollama, es necesario utilizar la herramienta curl. Curl es una utilidad de línea de comandos que permite transferir datos desde o hacia un servidor, utilizando diversos protocolos como HTTP, HTTPS, FTP, entre otros. Es especialmente útil para descargar archivos o enviar solicitudes a través de la red de manera sencilla y eficiente.

En la Figura 3.2 que se presenta a continuación, se puede observar la instalación del paquete antes descrito.

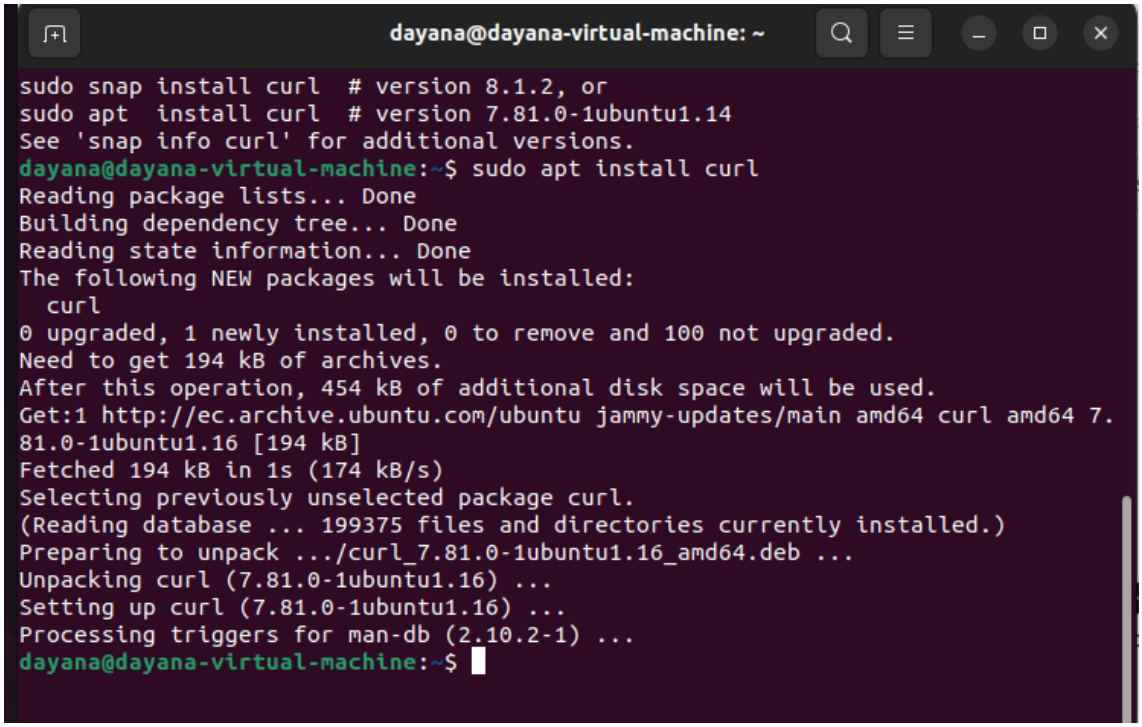

Figura 3.2 Instalación de curl

Para la instalación de Ollama, se utilizó el paquete proporcionado en la página oficial de Ollama para garantizar una instalación sin problemas. Además, se empleó la herramienta curl previamente descargada. Curl facilitó la descarga y configuración del paquete de instalación de forma eficiente y segura.

En la Figura 3.3 de a continuación, se puede observar el proceso descrito, incluyendo la utilización de curl para la descarga y la posterior instalación del servicio de Ollama.

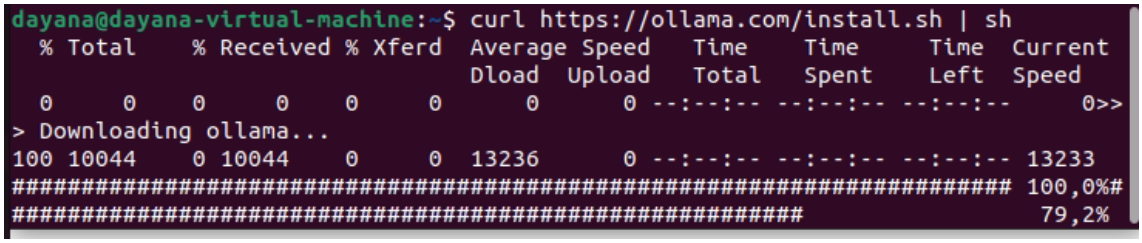

Figura 3.3 Clonación del repositorio de Ollama

Para la instalación de LLaMA2, se deben seguir una serie de pasos después de descargar el paquete de instalación desde la página oficial de LLaMA2. Una vez que se tiene el archivo descargado, el primer paso es descomprimir el paquete utilizando una herramienta de descompresión adecuada, como tar. Esto se hace para extraer todos los archivos necesarios para la instalación.

Después de extraer los archivos, es importante acceder al directorio donde se encuentran los archivos extraídos. Dentro de este directorio, se deben instalar todas las dependencias necesarias para que LLaMA2 funcione correctamente. Esto puede implicar la instalación de librerías específicas o herramientas adicionales mediante el gestor de paquetes del sistema operativo.

A continuación, se debe ejecutar el script de instalación proporcionado en el paquete. Este script configura el sistema y copia los archivos necesarios en los directorios adecuados. Para ejecutar el script, es necesario darle permisos de ejecución y luego iniciarlo.

Para verificar que la instalación de la aplicación se ha completado correctamente, es necesario ejecutar un comando de prueba. Este comando, ollama run llama2, asegura que la aplicación esté funcionando como se espera. En la Figura 3.4, se muestra la ejecución de este comando, confirmando que la instalación se ha realizado exitosamente y que LLaMA2 está operativa.

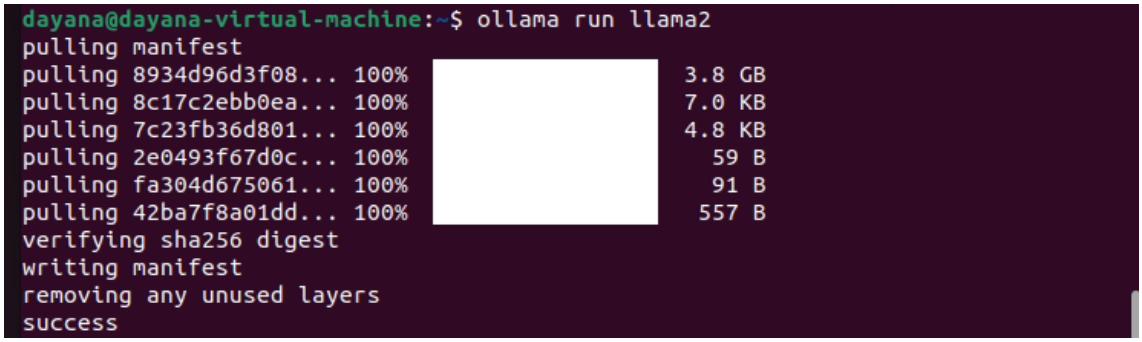

Figura 3.4 Ejecución del comando para activar Ollama

Para verificar que Ollama está correctamente instalado, se debe realizar una prueba interactuando con el modelo. Esto se puede hacer formulando una pregunta al modelo, como se muestra en la Figura 3.5. Si Ollama responde adecuadamente a la pregunta, se puede confirmar que la instalación del paquete ha sido exitosa y que el modelo está funcionando correctamente.

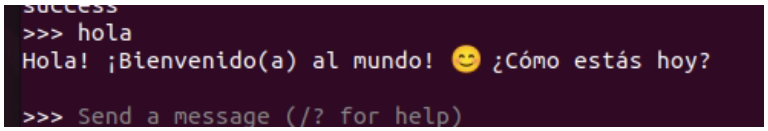

Figura 3.5 Prueba de ollama

Para instalar Docker en Ubuntu 22.04, es necesario seguir una serie de pasos detallados que aseguran una instalación correcta y segura. Docker es una plataforma que permite a los desarrolladores empaquetar aplicaciones en contenedores, facilitando su despliegue y administración. A continuación, se describen los pasos necesarios para instalar Docker en un sistema Ubuntu 22.04.

El primer paso en el proceso de instalación es actualizar los paquetes del sistema. Esto asegura que el sistema operativo tenga las versiones más recientes de los paquetes disponibles. Para actualizar el sistema, abra una terminal y ejecute los siguientes comandos: sudo apt-get update y sudo apt-get upgrade -y.

Docker requiere algunas dependencias adicionales para funcionar correctamente. Instale estas dependencias utilizando el siguiente comando: sudo apt-get install -y apt-transport-https ca-certificates software-properties-common. Estas herramientas permiten al sistema manejar repositorios sobre HTTPS y descargar los paquetes necesarios para Docker.

A continuación, se debe añadir el repositorio oficial de Docker a las fuentes de paquetes del sistema. Esto se hace descargando la clave GPG del repositorio y añadiéndola al sistema, seguido de la adición del repositorio de Docker: sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg < (curl -fsSL https://download.docker.com/linux/ubuntu/gpg). Luego, añada el repositorio a las fuentes de APT con el comando: echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null.

Con el repositorio añadido, actualice el índice de paquetes de APT nuevamente e instale Docker ejecutando los siguientes comandos: sudo apt-get update y sudo apt-get install -y docker-ce. Docker ahora se descargará y se instalará en su sistema. Docker CE (Community Edition) es la versión gratuita y de código abierto de Docker.

Una vez que Docker esté instalado, es importante verificar que se haya instalado correctamente y que esté funcionando. Puede hacer esto ejecutando el siguiente comando, que debería mostrar la versión de Docker instalada: sudo docker --version.

Para permitir que Docker se ejecute sin necesidad de privilegios de superusuario, agregue su usuario al grupo de Docker con el siguiente comando: sudo usermod -aG docker ${USER}. Después de ejecutar este comando, cierre la sesión y vuelva a iniciarla para que los cambios surtan efecto.

Finalmente, pruebe que Docker está funcionando correctamente ejecutando un contenedor de prueba: docker run hello-world. Este comando descargará una imagen de prueba y ejecutará un contenedor, mostrando un mensaje de confirmación si todo está configurado correctamente. En la Figura 3.6 se puede presentar como el servicio de Docker está funcionando correctamente.

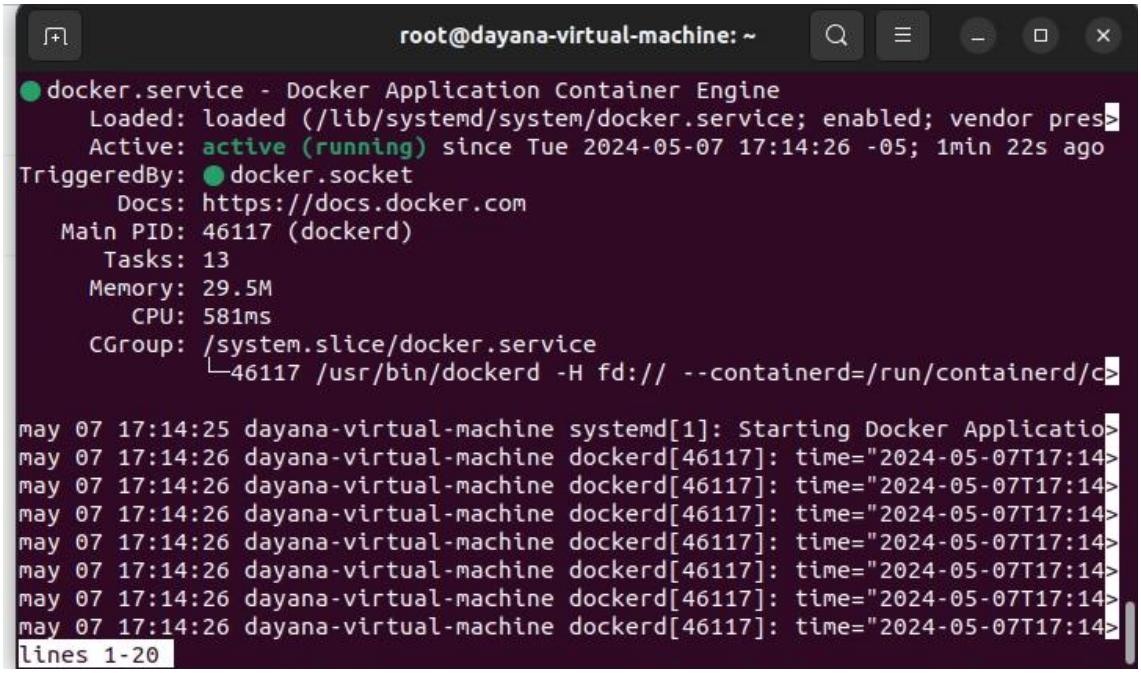

Figura 3.6 Servicio de Docker funcionando

Para el siguiente paso, se intentó ejecutar el comando docker-compose up -d, el cual inicialmente presentó problemas. Estos problemas podían estar relacionados con la configuración del entorno o la instalación de Docker. Para resolver esto, se decidió instalar Docker utilizando snap, un gestor de paquetes de aplicaciones para sistemas Linux. El comando utilizado fue snap install docker, lo que permitió una instalación más simplificada y gestionada del servicio Docker. En la Figura 3.7 que se presenta a continuación, se pueden observar los comandos ejecutados y los resultados obtenidos, demostrando cómo la instalación mediante snap resolvió los inconvenientes iniciales y permitió que Docker Compose funcionara correctamente. Esta solución facilitó la correcta configuración y puesta en marcha de los servicios necesarios para el proyecto.

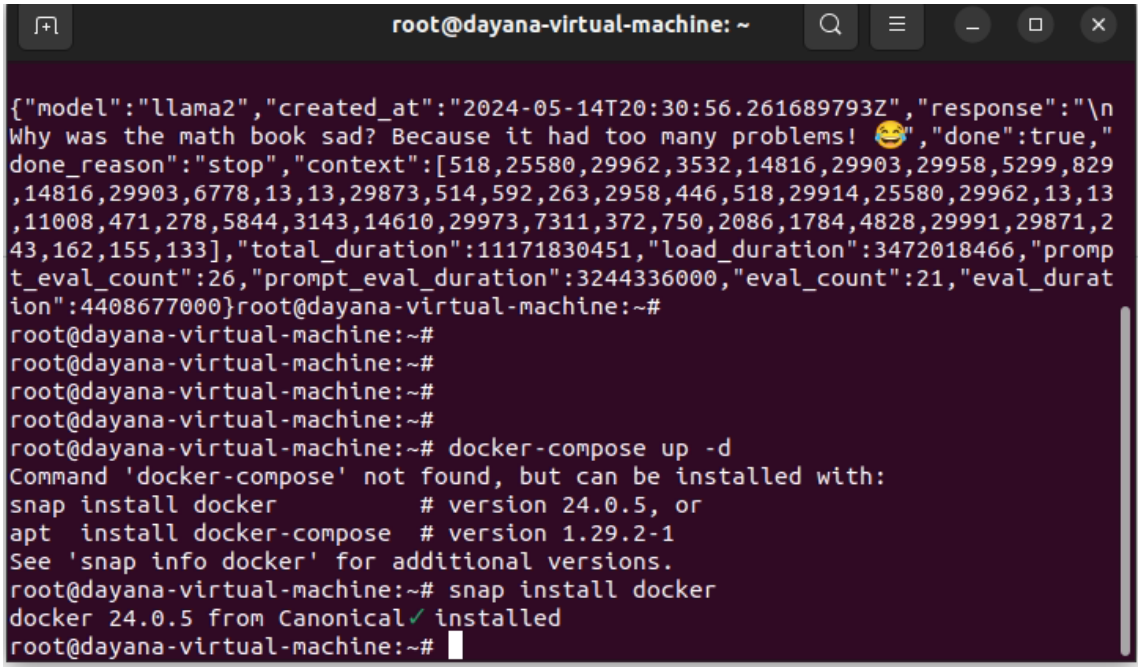

Figura 3.7 Verificación Docker

El archivo de configuración docker-compose.yaml mostrado en la imagen define y ejecuta múltiples contenedores Docker como un único servicio, facilitando la gestión y orquestación de aplicaciones complejas. Este archivo configura dos servicios principales: chatgpt y ollama, cada uno con sus propias especificaciones y dependencias.

El servicio chatgpt utiliza una imagen de Docker obtenida desde el registro de contenedores de GitHub (ghcr.io/ivanfloravanti/chatbot-ollama:main). Esto garantiza que se está utilizando una versión específica del chatbot ollama para el servicio chatgpt. Además, el archivo mapea el puerto 3000 del contenedor al puerto 3000 de la máquina host, lo cual permite acceder a la aplicación desde la máquina host a través del puerto 3000.

Para personalizar el comportamiento del servicio chatgpt, se configuran variables de entorno. La variable DEFAULT_MODEL se establece en llama2, indicando el modelo por defecto que se utilizará. Asimismo, OLLAMA_HOST se configura para apuntar a la URL del servicio ollama que se ejecuta en el puerto 11434. Estas variables de entorno permiten que chatgpt se comunique correctamente con el servicio ollama.

Por otro lado, el servicio ollama se configura utilizando la imagen ollama/ollama:latest, asegurando que se está empleando la última versión disponible del software. Se define un volumen que mapea el directorio ./ollama/models en la máquina host al directorio /ollama/models dentro del contenedor. Esto es crucial para la persistencia de datos, ya

que cualquier cambio realizado en los modelos se mantiene incluso si el contenedor se reinicia.

Además, el servicio ollama también utiliza una variable de entorno OLLAMA_MODELS, que especifica la ruta a los modelos dentro del contenedor, apuntando al directorio mapeado anteriormente. Esta configuración asegura que el servicio ollama pueda acceder a los modelos necesarios para su funcionamiento.

El archivo docker-compose.yaml configura y levanta dos servicios interdependientes, chatgpt y ollama. El servicio chatgpt depende del servicio ollama para funcionar correctamente, como se indica en las variables de entorno configuradas. La configuración de puertos y volúmenes asegura que los servicios sean accesibles y que los datos persistentes estén correctamente gestionados. Esta configuración es esencial para desplegar aplicaciones basadas en microservicios utilizando Docker, proporcionando una manera eficiente y estructurada de manejar múltiples contenedores y sus interdependencias. En la Figura 3.8 se puede observar todo el codigo antes descrito.

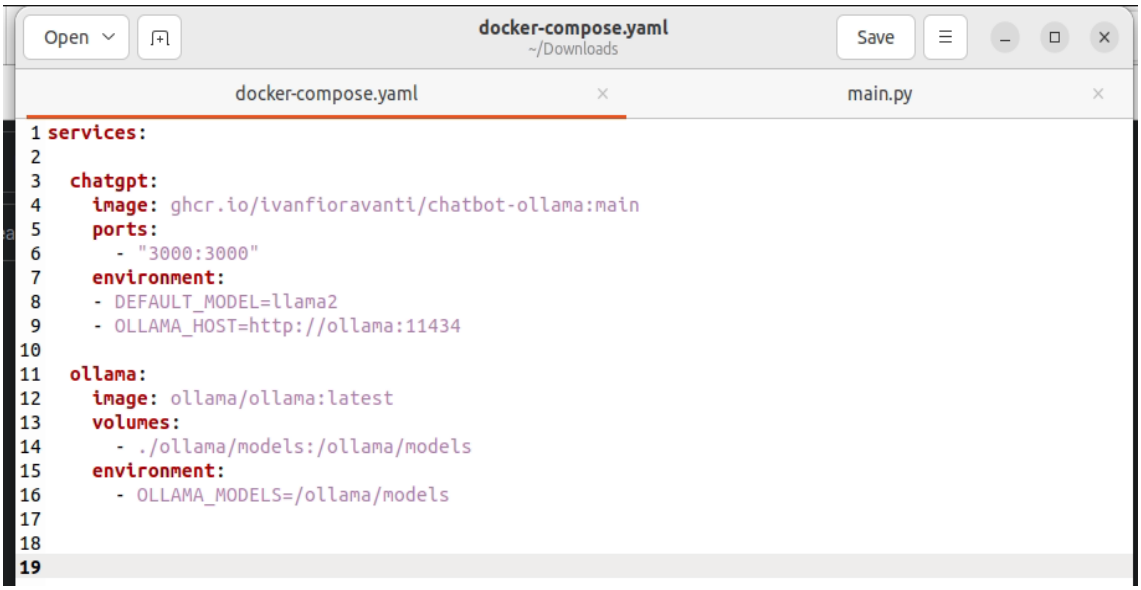

Figura 3.8 Docker-compose.yaml

El código main.py mostrado en la imagen complementa la configuración definida en el archivo docker-compose.yaml al implementar la lógica necesaria para interactuar con el servicio ollama. Este script escrito en Python importa las bibliotecas necesarias (requests, json, y gradio) y define la URL del endpoint de la API que se utilizará para generar respuestas: http://localhost:11434/api/generate. Además, se configuran los encabezados HTTP necesarios para la solicitud, estableciendo el tipo de contenido como application/json.

El script inicializa una lista llamada conversation_history para mantener un registro de la conversación [12].La función generate_response(prompt) es la encargada de manejar las solicitudes al servicio de generación de texto. Esta función añade el prompt actual al historial de la conversación y construye un mensaje completo que incluye todo el historial. A continuación, se prepara un diccionario data que contiene la configuración del modelo ("model": "mistral"), el estado de stream (False), y el prompt completo.

El script utiliza la biblioteca requests para enviar una solicitud POST al endpoint de la API, incluyendo los encabezados y el cuerpo de la solicitud en formato JSON. Si la respuesta del servidor es exitosa (status_code == 200), el script procesa la respuesta, extrayendo el texto generado y añadiéndolo al historial de la conversación. Finalmente, devuelve la respuesta generada. Si ocurre algún error, el script imprime el código de estado y el texto de la respuesta y retorna None. En la Figura 3.9 se presenta el codigo copleto de lo antes descrito.

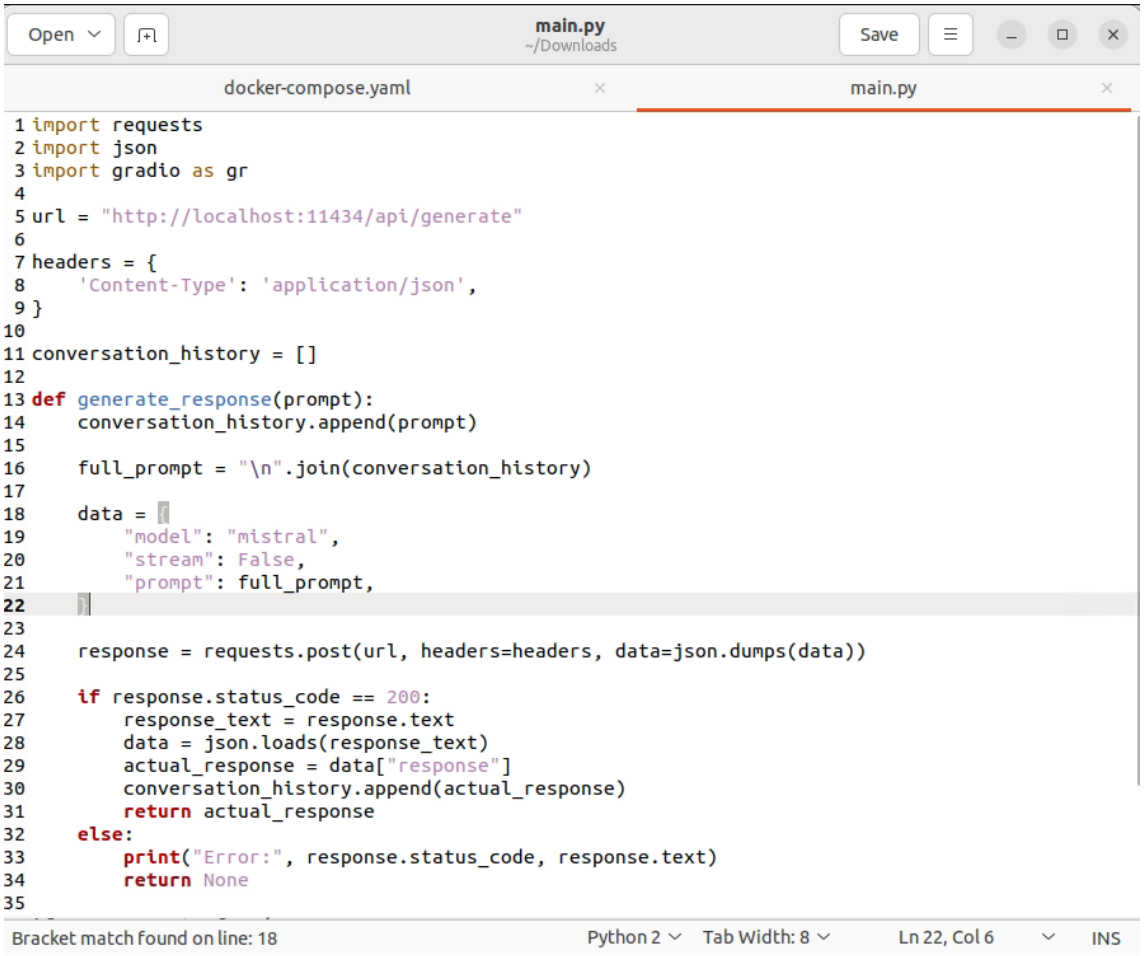

Figura 3.9 Archivo main.py

Este script main.py se integra perfectamente con la configuración del archivo docker-compose.yaml, donde se definieron los servicios chatgpt y ollama. La configuración del

servicio chatgpt incluye una variable de entorno OLLAMA_HOST, que apunta a http://ollama:11434, la misma URL utilizada en el script para interactuar con el servicio ollama. Esta integración asegura que el script Python puede comunicarse con el servicio ollama en el entorno Docker, utilizando los modelos y configuraciones especificadas. La orquestación de los contenedores mediante Docker Compose, combinada con la lógica de interacción en main.py, proporciona una solución robusta para la generación de texto basada en modelos de inteligencia artificial.

En la Figura 3.10 muestra una serie de comandos ejecutados en una terminal de Linux, que están relacionados con la puesta en marcha y gestión de los contenedores Docker definidos en el archivo docker-compose.yaml. Estos comandos complementan y verifican la configuración previamente explicada, asegurando que los servicios chatgpt y ollama funcionen correctamente.

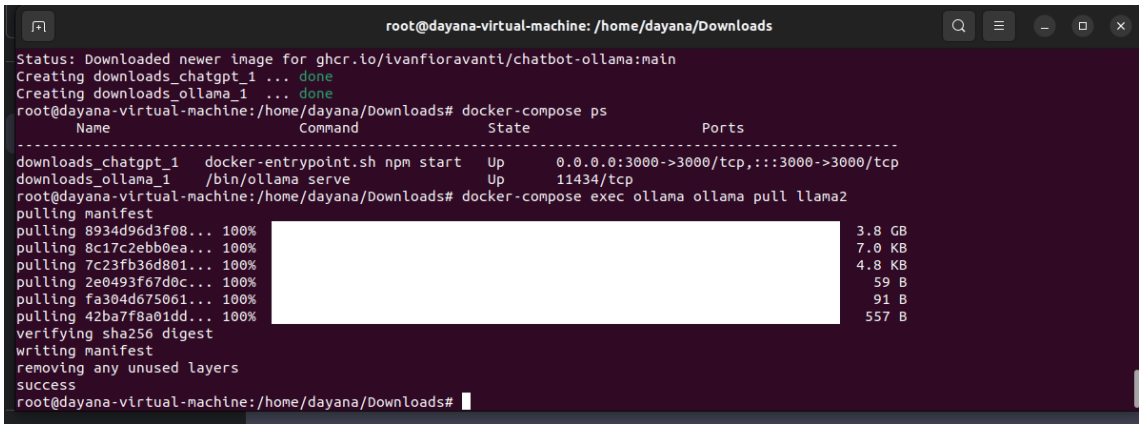

Figura 3.10 Despliegue de los contenedores

Primero, se ejecuta el comando docker-compose up -d, el cual inicia los contenedores definidos en el archivo docker-compose.yaml en segundo plano. Este comando descarga la imagen más reciente para ghcr.io/ivanfloravanti/chatbot-ollama:main y crea los contenedores chatgpt y ollama. Esto garantiza que las versiones más recientes de las imágenes se utilicen, proporcionando un entorno de ejecución actualizado y seguro.

Posteriormente, se utiliza el comando docker-compose ps para listar los contenedores activos. Este comando muestra información detallada sobre los contenedores, incluyendo sus nombres, comandos ejecutados, estado y puertos mapeados. En la imagen, se puede observar que tanto chatgpt como ollama están en estado de ejecución (Up), lo cual indica que ambos servicios se han iniciado correctamente y están listos para recibir solicitudes.

Finalmente, se ejecuta el comando docker-compose exec ollama ollama pull llama2 para descargar el modelo llama2 dentro del contenedor ollama. Este comando utiliza exec para ejecutar comandos arbitrarios dentro de un contenedor en ejecución. La salida muestra el progreso de la descarga del modelo, incluyendo la obtención de diferentes capas de la imagen y su tamaño en disco. Esto asegura que el modelo llama2 esté disponible y listo para ser utilizado por el servicio ollama.

Estos pasos completan el proceso de configuración e implementación descrito anteriormente. Inicialmente, el archivo docker-compose.yaml definió los servicios chatgpt y ollama, especificando las imágenes de Docker, los puertos, los volúmenes y las variables de entorno necesarias. Luego, el script main.py se encargó de la lógica de interacción con el servicio ollama, enviando solicitudes y manejando respuestas para generar texto basado en el modelo configurado.

La ejecución de los comandos en la terminal muestra la puesta en marcha de estos servicios y la gestión del modelo llama2 dentro del contenedor ollama. Este flujo de trabajo asegura que tanto la infraestructura Docker como el código de la aplicación están correctamente alineados y funcionales. La combinación de la configuración de Docker Compose, el script Python y los comandos de terminal proporciona una solución completa y cohesiva para el despliegue y la operación de servicios de inteligencia artificial en un entorno contenedorizado.

Con estos pasos, hemos configurado y desplegado exitosamente una interfaz gráfica de Ollama Web, utilizando Docker Compose para gestionar los contenedores y un script Python para interactuar con los servicios de generación de texto. Este entorno contenedorizado asegura que los servicios chatgpt y ollama funcionen de manera integrada y eficiente. En el siguiente punto, procederemos a realizar pruebas exhaustivas para garantizar que la aplicación funcione correctamente y cumpla con los requisitos esperados. Estas pruebas incluirán la verificación de la generación de respuestas, la estabilidad del sistema y el rendimiento bajo diferentes condiciones de uso.

3.4 Verificar el funcionamiento de cada servicio de cómputo.

La implementación descrita abarca una serie de pasos cuidadosamente coordinados para asegurar la correcta configuración y despliegue de una interfaz gráfica de Ollama Web. Inicialmente, se definieron los servicios chatgpt y ollama en el archivo docker-compose.yaml, especificando las imágenes de Docker, los puertos, los volúmenes y las

variables de entorno necesarias. Esto permitió la creación de un entorno contenedorizado eficiente y seguro.

El script main.py se integró perfectamente con la configuración de Docker Compose, manejando la lógica necesaria para interactuar con el servicio ollama. Este script envía solicitudes y procesa respuestas para generar texto basado en el modelo configurado, utilizando la URL del servicio ollama y asegurando que todas las interacciones se realicen correctamente.

Para verificar que los contenedores funcionaban como se esperaba, se ejecutaron varios comandos en la terminal. Primero, docker-compose up -d inició los contenedores en segundo plano, asegurando que las versiones más recientes de las imágenes se utilizaran. Luego, docker-compose ps mostró que los contenedores estaban en ejecución, confirmando que los servicios estaban listos para recibir solicitudes. Finalmente, docker-compose exec ollama ollama pull llama2 descargó el modelo llama2 dentro del contenedor ollama, preparando el entorno para su uso [13].

La Figura 3.11presentada muestra una prueba realizada para verificar la correcta configuración e interacción de los servicios. Esta prueba engloba la finalización del trabajo de titulación, demostrando que la interfaz gráfica de Ollama Web está funcionando correctamente de forma local en el puerto 3000. Utilizando un navegador, se puede acceder a esta interfaz y realizar las operaciones necesarias, confirmando que todos los componentes del sistema están trabajando en conjunto de manera efectiva.

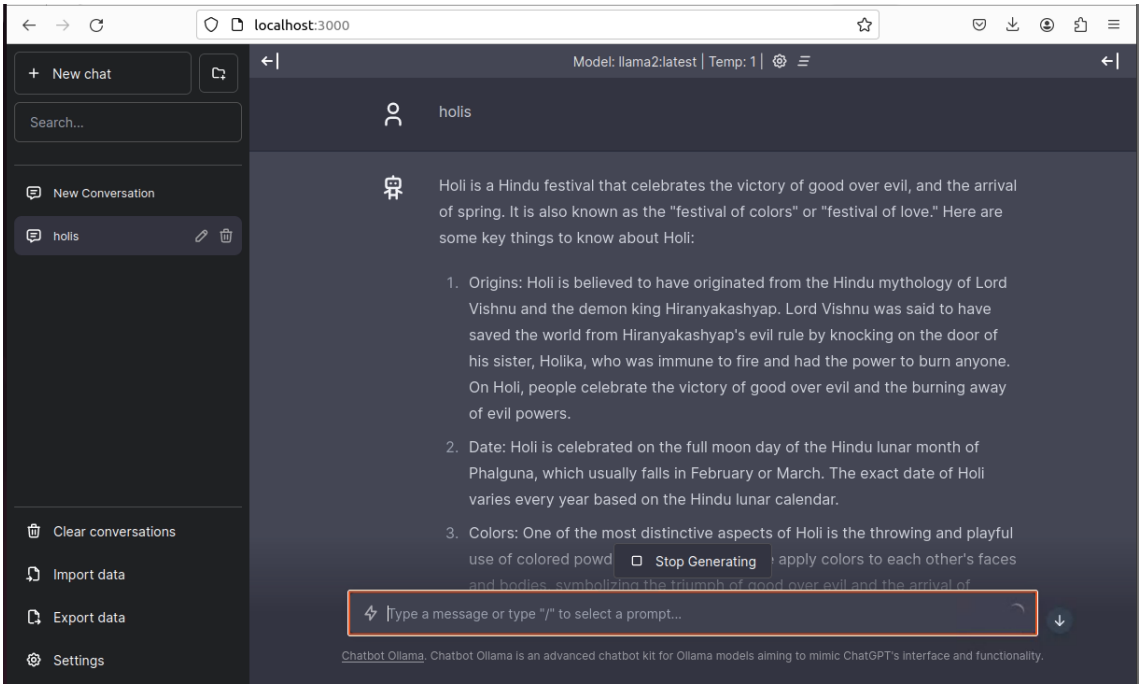

Figura 3.11 Prueba final del trabajo de titulación.

Con la interfaz gráfica de Ollama Web desplegada y operativa, el siguiente paso consiste en realizar pruebas exhaustivas. Estas pruebas asegurarán que la aplicación no solo funcione correctamente, sino que también cumpla con todos los requisitos esperados en términos de generación de respuestas, estabilidad del sistema y rendimiento bajo diferentes condiciones de uso. Este proceso de verificación es crucial para garantizar que el proyecto de titulación se complete con éxito y satisfaga todas las expectativas establecidas [14].

4 CONCLUSIONES

- Docker implementa la facilidad de crear su entorno de ejecución de manera consistente para la aplicación ollama2 en el sistema operativo Linux quien es compatible con la tecnología Docker, pueden producir los desarrollos de su entorno, donde se ejecuta de forma avanzada en su medio.
- Para utilizar Ollama2 se ha mejorado el uso de manera accesible del usuario dentro de la interfaz grafica para poder realizar todas las interrogaciones quien tiene en su experiencia.
- Al implementar la plataforma DevOps se implementa integración continua, se disminuye los momentos para desplegar para mejorar en la eficiencia del flujo de trabajo.
- Se ha generado un ambiente con producciones estables, ha mejorado en generar las dependencias y resolver los problemas que existen dentro de la maquina local, encuentra las soluciones técnicas para poder averiguar recursos y documentos útiles para validar los documentos para servir en la herramienta, quien sirve a la comunidad para brindar soporte de Ubuntu.
- La implementación de Docker Compose para la orquestación de los servicios chatgpt y ollama permitió una configuración eficiente y segura del entorno necesario para la aplicación de inteligencia artificial. La definición precisa de imágenes, puertos, volúmenes y variables de entorno en el archivo docker-compose.yaml garantizó que todos los componentes del sistema se desplegaran de manera coordinada. Este enfoque minimizó errores y aseguró la integridad del sistema, proporcionando una base sólida para su operación.
- El despliegue de la interfaz gráfica de Ollama Web en el puerto 3000 proporcionó una forma accesible y amigable para que los usuarios interactúen con el sistema. La prueba realizada y presentada en la figura confirma que la aplicación es operativa y que los usuarios pueden acceder a sus funcionalidades mediante un navegador web. Esto facilita enormemente el uso del sistema y demuestra la versatilidad y la facilidad de acceso que ofrece la solución implementada, mejorando la experiencia del usuario.

5 RECOMENDACIONES

- Verificar que todas las características de los documentos de ollama2 se programe de forma correcta con todos los requisitos y capacidades.
- Hacer que todos los requisitos de software y hardware ha funcionado de forma avanzada, comprobar que la interfaz gráfica pruebe con su diseño y funcionalidad del programa.
- Evidenciar todo lo necesario de integración de ollma2 sobre las bibliotecas y frameworks, probar todas las tecnologías y herramientas como principalmente en node.js de la interfaz gráfica.
- Se debe organizar todos los documentos del código y procesos de desarrolladores se debe analizar de manera fácilmente para que se puede formar en el futuro para que pueda contribuir.
- Se debe implementar pruebas para verificar todas las funciones inclusive de la máquina virtual con todos los comandos y en línea las interfaces graficas para analizar que todos los componentes trabajan bien unidos.
- Al verificar dentro de su interfaz local se puede obtener un despliegue completo, para trabajar en todas las funciones.
- Monitorear la herramienta de código como Inteligencia artificial para probar con otros métodos.
- Se debe proteger los datos sensibles, para cuidar toda la información que se ingresa de forma personal, para que las medidas de seguridad sirvan a los de los programas y aplicaciones con sus normativas.
- Poder darle una guía para los usuarios finales para comprender los resultados entregados por la inteligencia artificial, para poder manejar las interrogaciones que existen durante al utilizar la herramienta.

6 BIBLIOGRAFÍA

[1] «Probando OLLAMA - Tu propio ChatGPT local,» 5 Diciembre 2023. [En línea]. Available: https://www.youtube.com/watch?v=OmLKfDoAfyM. [Último acceso: 28 Mayo 2024].

[2] S. Romero, «¿Cuando nacio la inteligencia artificial?,» Tecnologia, 21 Julio 2023. [En línea]. Available: https://www.muyinteresante.com/tecnologia/60947.html. [Último acceso: 28 Mayo 2024].

[3] «¿Que es la inteligencia artificial y como se usa?,» 08 Septiembre 2020. [En línea]. Available: https://www.europarl.europa.eu/topics/es/article/20200827STO85804/que-es-la-inteligencia-artificial-y-como-se-usa. [Último acceso: 28 Mayo 2024].

[4] «La tabla de routing,» [En línea]. Available: https://www.sapalomera.cat/moodlecf/RS/2/course/module7/7.5.1.3/7.5.1.3.html. [Último acceso: 28 Mayo 2024].

[5] «Que es una LAN (red de área local),» [En línea]. Available: https://www.cloudflare.com/es-es/learning/network-layer/what-is-a-lan/. [Último acceso: 28 Mayo 2024].

[6] «Ollama,» Generalitat , [En línea]. Available: https://portal.edu.gva.es/appsedu/es/ollama/. [Último acceso: 28 Mayo 2024].

[7] C. Vargas, «Implementacion de DevOps con contenedores y Docker,» trycore, [En línea]. Available: https://trycore.co/buenas-practicas-ti/implementacion-devops-contenedores/. [Último acceso: 28 Mayo 2024].

[8] H. Herrero, «Ollama, empezando con la IA local,» 30 abril 2024. [En línea]. Available: https://www.bujarra.com/ollama-empezando-con-la-ia-local/. [Último acceso: 26 junio 2024].

[9] J. L. Alonso, «TensorFlow,» [En línea]. Available: https://www.incentro.com/es-ES/blog/que-es-tensorflow. [Último acceso: 24 junio 2024].

[10] «Docker Compose,» 19 Junio 2024. [En línea]. Available: https://imaginaformacion.com/tutoriales/que-es-docker-compose. [Último acceso: 24 Junio 2024].

[11] G. Urtiaga, «Como crear tu propio Chatbot de IA con Ollama y Open-WebUI,» 26 Mayo 2024. [En línea]. Available: https://aprendeit.com/como-crear-tu-propio-chatbot-de-ia-con-ollama-y-open-webui/. [Último acceso: 16 Julio 2024].

[12] «HTTP Y HTTPS,» [En línea]. Available: https://aws.amazon.com/es/compare/the-difference-between-https-and-http/#:~:text=El%20protocolo%20de%20transferencia%20de,responde%20con%20una%20respuesta%20HTTP.. [Último acceso: 15 Julio 2024].

[13] «Que es Docker Compose,» 16 Junio 2024. [En línea]. Available: https://imaginaformacion.com/tutoriales/que-es-docker-compose. [Último acceso: 20 Julio 2024].

[14] «Open WebUI,» [En línea]. Available: https://docs.openwebui.com/. [Último acceso: 20 Julio 2024].

[15] M. Aayush, 24 abril 2024. [En línea]. Available: https://www.unite.ai/pt/tudo-o-que-voc%C3%AA-precisa-saber-sobre-o-modelo-de-c%C3%B3digo-aberto-mais-poderoso-do-llama-3-at%C3%A9-agora%2C-conceitos-para-uso/. [Último acceso: 24 junio 2024].

7 ANEXOS

En caso necesario, el documento escrito deberá incluir los anexos y secciones que incorporan información que sea relevante, pero que, por su extensión, no pueden ser incorporadas directamente en ninguna de las secciones anteriores. Normalmente, en la sección de Anexos se incluyen conjuntos de datos extensos, formatos de encuestas, entrevistas, enlaces hacia videos o programas que sean producto o formen parte del Trabajo de Integración Curricular, entre otros.

La lista de los **Anexos** se muestran a continuación:

ANEXO I. Certificado de originalidad

ANEXO II. Enlaces

ANEXO III. Conjunto de datos extensos

La numeración de los **Anexos** debe realizarse con números en formato romano minúscula.

ANEXO I: Certificado de Originalidad

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. xx de xxxx de 2022

De mi consideración:

Yo, FERNANDO VINICIO BECERRA CAMACHO, en calidad de director del Trabajo de Integración Curricular titulado IMPLEMENTACIÓN DE HERRAMIENTA DE INTELIGENCIA ARTIFICIAL DE FORMA LOCAL Y UNA INTERFAZ GRÁFICA WEB. elaborado por la estudiante DAYANA ESTEFANIA GRANDA RIVAS de la carrera en Tecnología Superior en Redes y Telecomunicaciones, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito completo, producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del xxx%.

Es todo cuanto puedo certificar en honor a la verdad, pudiendo el interesado hacer uso del presente documento para los trámites de titulación.

NOTA: Se adjunta el link del informe generado por la herramienta Turnitin.

LINK

Atentamente,

NOMBRE

Docente

Escuela de Formación de Tecnólogos

ANEXO II: Enlaces

Anexo II.I Código QR de la implementación y pruebas de funcionamiento

ANEXO III: Códigos Fuente

- docker-compose.yaml

```
services:

  chatgpt:
    image: ghcr.io/ivanfioravanti/chatbot-ollama:main
    ports:
      - "3000:3000"
    environment:
    - DEFAULT_MODEL=llama2
    - OLLAMA_HOST=http://ollama:11434

  ollama:
    image: ollama/ollama:latest
    volumes:
      - ./ollama/models:/ollama/models
    environment:
      - OLLAMA_MODELS=/ollama/models
```

- Main.py

```
import requests
import json
import gradio as gr

url = "http://localhost:11434/api/generate"

headers = {
    'Content-Type': 'application/json',
}

conversation_history = []

def generate_response(prompt):
    conversation_history.append(prompt)

    full_prompt = "\n".join(conversation_history)
```

```
    data = {
        "model": "mistral",
        "stream": False,
        "prompt": full_prompt,
    }

    response = requests.post(url, headers=headers, data=json.dumps(data))

    if response.status_code == 200:
        response_text = response.text
        data = json.loads(response_text)
        actual_response = data["response"]
        conversation_history.append(actual_response)
        return actual_response
    else:
        print("Error:", response.status_code, response.text)
        return None

iface = gr.Interface(
    fn=generate_response,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Enter your prompt here..."),
    outputs="text"
)

iface.launch()
```