

Universidad de Monterrey



Integración de aplicaciones computacionales

Ejercicio Guiado 2

**Programa en Flask con request que use el API de Ollama con la IA de Deep Seek-Coder**

**en el contenedor Docker**

**Maestro:** Dr. Raúl Morales Salcedo

**Nombre:** Margarita Concepción Cuervo Citalán #581771

**Carrera:** ITC, 9no Semestre

**Día y hora del grupo de la clase:** lunes y jueves 10:00h

18 de agosto de 2025.

*Doy mi palabra de haber realizado esta actividad con integridad académica.*

# **ÍNDICE**

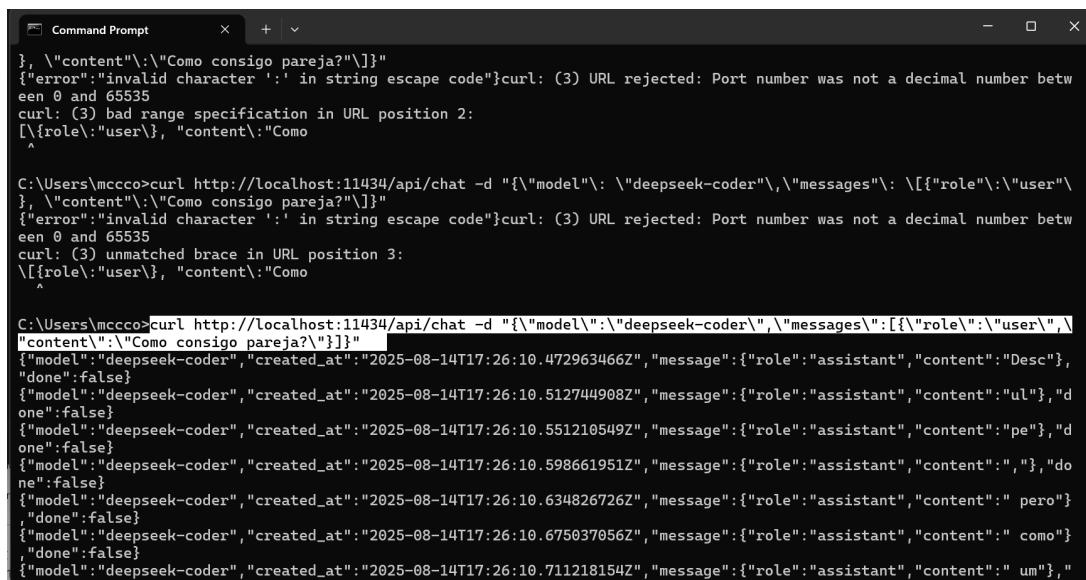
Introducción	2
Desarrollo	2
Reflexión	10
Conclusión	10

## Introducción

Este ejercicio guiado tiene como objetivo desarrollar una programa en Flask con request que use el API de Ollama con la IA de Deep Seek-Coder en el contenedor Docker. A través del uso de las herramientas involucradas que se usaron en la sesión de clase y de la ayuda de la inteligencia artificial, se busca reforzar habilidades en programación, validación de datos y manejo de argumentos en línea de comandos, complementado con la creación de un sitio web para documentar y reflexionar sobre el proceso.

## Desarrollo

Se empezó este ejercicio por abrir la terminal y correr el comando **docker run -d -p 11434:11434 --name ollama ollama/ollama** para poder correr el contenedor de ollama dentro de docker. Luego se instaló la IA de Deep Seek - Coder dentro de Ollama con el comando **docker exec -it ollama ollama pull deepseek-coder**. Para verificar que esta API funcionara se verifico con el siguiente comando curl: **curl http://localhost:11434/api/chat -d {"model":"deepseek-coder","messages":[{"role":"user","content":"Como consigo pareja?"}]}**.

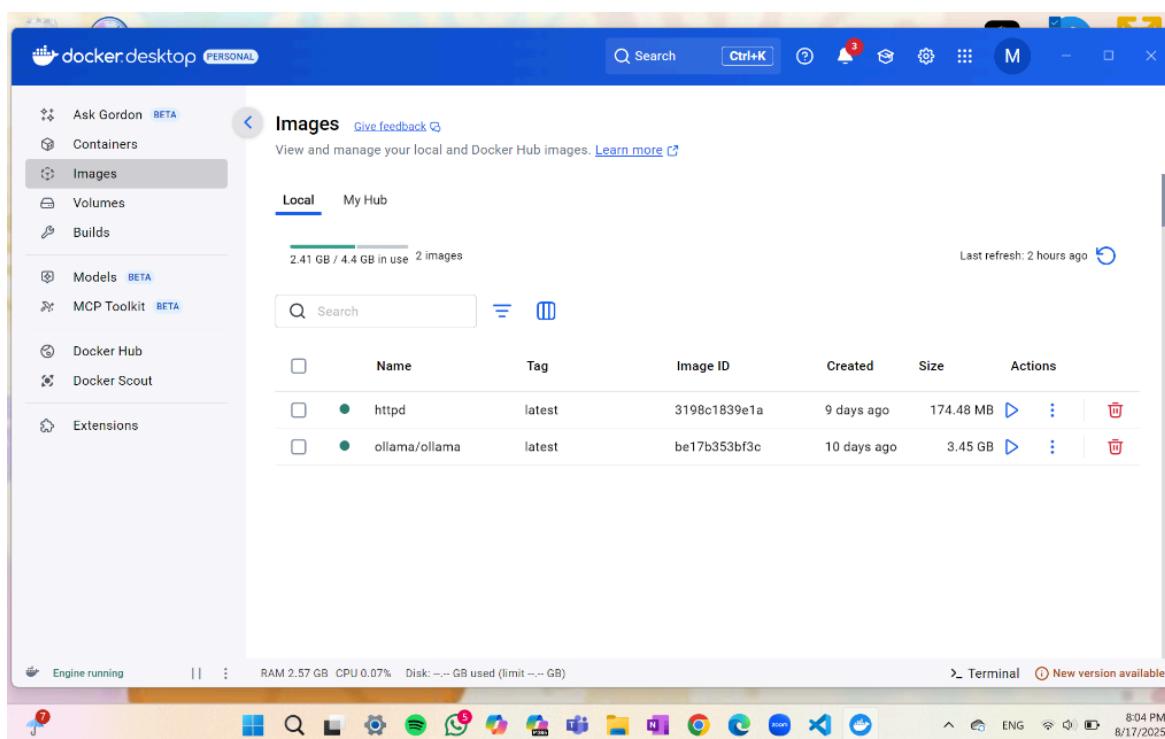
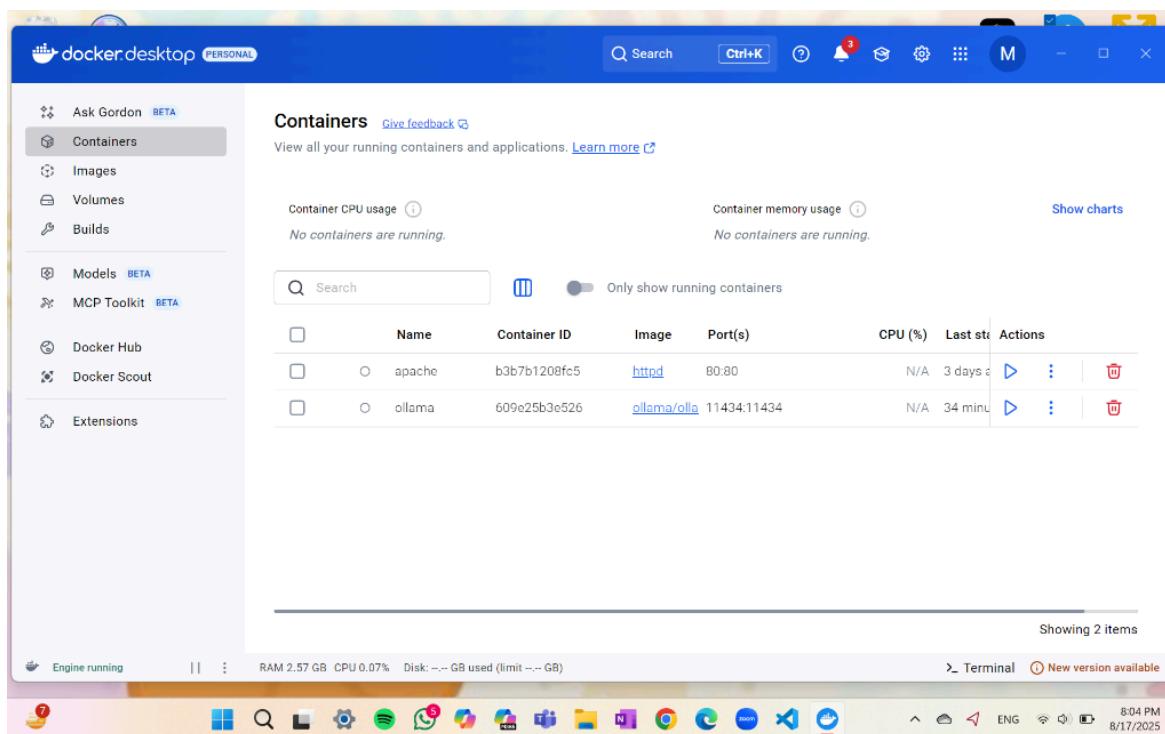


```
Command Prompt
C:\Users\mccco>curl http://localhost:11434/api/chat -d "{\"model\":\"deepseek-coder\",\"messages\": [{\"role\":\"user\", \"content\":\"Como consigo pareja?\"}]}"
{"error": "invalid character ':' in string escape code"}curl: (3) URL rejected: Port number was not a decimal number between 0 and 65535
curl: (3) bad range specification in URL position 2:
[\"{role\":\"user\"}, \"content\":\"Como
^

C:\Users\mccco>curl http://localhost:11434/api/chat -d "{\"model\":\"deepseek-coder\",\"messages\": [{\"role\":\"user\", \"content\":\"Como consigo pareja?\"}]}"
{"error": "invalid character ':' in string escape code"}curl: (3) URL rejected: Port number was not a decimal number between 0 and 65535
curl: (3) unmatched brace in URL position 3:
\[\"{role\":\"user\"}, \"content\":\"Como
^

C:\Users\mccco>curl http://localhost:11434/api/chat -d "{\"model\":\"deepseek-coder\",\"messages\": [{\"role\":\"user\", \"content\":\"Como consigo pareja?\"}]}"
{
  "model": "deepseek-coder",
  "created_at": "2025-08-14T17:26:10.472963466Z",
  "message": {
    "role": "assistant",
    "content": "Desc"
  },
  "done": false
}
{
  "model": "deepseek-coder",
  "created_at": "2025-08-14T17:26:10.512744908Z",
  "message": {
    "role": "assistant",
    "content": "ul"
  },
  "done": false
}
{
  "model": "deepseek-coder",
  "created_at": "2025-08-14T17:26:10.551210549Z",
  "message": {
    "role": "assistant",
    "content": "pe"
  },
  "done": false
}
{
  "model": "deepseek-coder",
  "created_at": "2025-08-14T17:26:10.598661951Z",
  "message": {
    "role": "assistant",
    "content": ""
  },
  "done": false
}
{
  "model": "deepseek-coder",
  "created_at": "2025-08-14T17:26:10.634826726Z",
  "message": {
    "role": "assistant",
    "content": " pero"
  },
  "done": false
}
{
  "model": "deepseek-coder",
  "created_at": "2025-08-14T17:26:10.675037056Z",
  "message": {
    "role": "assistant",
    "content": " como"
  },
  "done": false
}
{
  "model": "deepseek-coder",
  "created_at": "2025-08-14T17:26:10.711218154Z",
  "message": {
    "role": "assistant",
    "content": " um"
  },
  "done": false
}
```

Asimismo, al crear el contenedor Ollama y su respectiva imagen, estas aparecen dentro de Docker:



Luego para verificar que en efecto están corriendo dicha API y su imagen ejecutamos los siguientes comandos:

1. **docker ps**: muestra la lista de los los contenedores e imagenes que estan corriendo, por ejemplo ollama
2. **docker start ollama**: comando para que empiece a correr el contenedor de Ollama con su respectiva imagen.

```
Command Prompt - python < + <
* Debugger PIN: 993-530-846
172.17.0.1 - - [18/Aug/2025 01:27:51] "GET /api/chat HTTP/1.1" 200 -
172.17.0.1 - - [18/Aug/2025 01:28:11] "POST /ask HTTP/1.1" 200 -

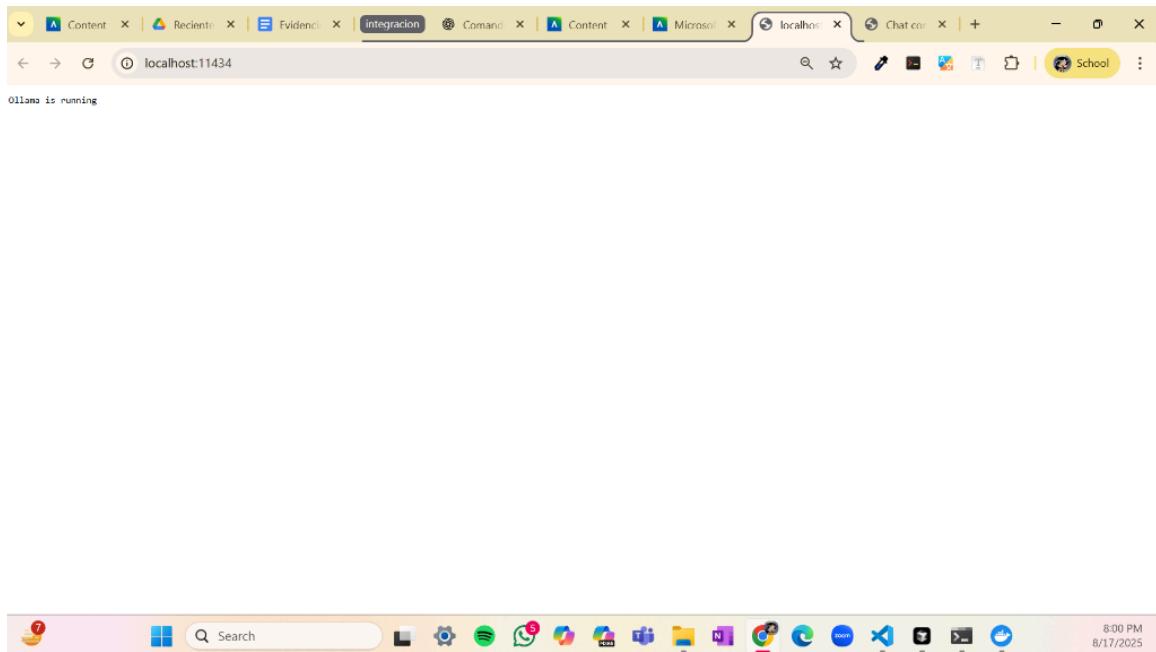
C:\Users\mcco\api-ollama>

C:\Users\mcco\api-ollama>docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

C:\Users\mcco\api-ollama>docker start ollama
ollama

C:\Users\mcco\api-ollama>docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
MES
609e25b3e526 ollama/ollama "/bin/ollama serve" 4 days ago Up 3 seconds 0.0.0.0:11434->11434/tcp, [::]:11434->11434/tcp ollama
```

Asimismo, se puede verificar que Ollama corre ya que al escribir en el navegador: **localhost:11434**, se muestra el mensaje *Ollama is running.*



Luego se le pidió a la IA de Chat GPT el siguiente prompt: “*ayudame a hacer una página web que consuma el API de Ollama y el deep seek coder en el puerto 11434*”. Tras pedirle esto, la IA dijo que se debía crear un directorio donde se encontraría todos los archivos HTML y .py de la página. Dicho directorio lo bautice como **flask-deepseekchat**

```

C:\Users\mcccco>mkdir flask-deepseekchat
C:\Users\mcccco>cd flask-deepseekchat
C:\Users\mcccco\flask-deepseekchat>code .

C:\Users\mcccco\flask-deepseekchat>pip install flask requests
Requirement already satisfied: flask in c:\users\mcccco\appdata\local\programs\python\python313\lib\site-packages (3.1.1)
Requirement already satisfied: requests in c:\users\mcccco\appdata\local\programs\python\python313\lib\site-packages (2.32.4)
Requirement already satisfied: blinker>=1.9.0 in c:\users\mcccco\appdata\local\programs\python\python313\lib\site-packages (from flask)
Requirement already satisfied: click>=8.1.3 in c:\users\mcccco\appdata\local\programs\python\python313\lib\site-packages (from flask)
Requirement already satisfied: itsdangerous>=2.2.0 in c:\users\mcccco\appdata\local\programs\python\python313\lib\site-packages (from flask)
Requirement already satisfied: jinja2>=3.1.2 in c:\users\mcccco\appdata\local\programs\python\python313\lib\site-packages (from flask)
Requirement already satisfied: markupsafe>=2.1.1 in c:\users\mcccco\appdata\local\programs\python\python313\lib\site-packages (from flask)
Requirement already satisfied: werkzeug>=3.1.0 in c:\users\mcccco\appdata\local\programs\python\python313\lib\site-packages (from flask)
Requirement already satisfied: charset_normalizer<4,>=2 in c:\users\mcccco\appdata\local\programs\python\python313\lib\site-packages (from requests)
Requirement already satisfied: idna<4,>=2.5 in c:\users\mcccco\appdata\local\programs\python\python313\lib\site-packages (from requests)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\mcccco\appdata\local\programs\python\python313\lib\site-packages (from requests)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\mcccco\appdata\local\programs\python\python313\lib\site-packages (from requests)
Requirement already satisfied: colorama in c:\users\mcccco\appdata\local\programs\python\python313\lib\site-packages (from click>=8.1.

```



Dentro de este directorio se encuentran dos archivos:

1. **app.py**: archivo python que contiene la librería de flask para hacer requests entre ella página web (HTML) y la API de Ollama que se encuentra en el puerto 11434. Por ello, nuestra página web correrá en el puerto por default de flask, el 5000, con la ruta /api/chat.

El código de **app.py** es el siguiente:

```

Python
from flask import Flask, render_template, request, jsonify
import requests

app = Flask(__name__)

# URL de Ollama (ya levantado en el host)
OLLAMA_API_URL = "http://localhost:11434/api/chat"

@app.route("/api/chat")
def index():
    return render_template("index.html")

@app.route("/ask", methods=[ "POST"])
def ask():
    user_message = request.json.get("message")
    payload = {
        "model": "deepseek-coder",
        "messages": [{"role": "user", "content": user_message}],
        "stream": False
    }

    try:
        response = requests.post(OLLAMA_API_URL, json=payload)
        response.raise_for_status()
    
```

```

        data = response.json()
        # Devuelve el contenido de la respuesta
        return jsonify({"reply": data["message"]["content"]})
    except Exception as e:
        return jsonify({"reply": f"Error: {str(e)}"})

if __name__ == "__main__":
    # Flask corre en localhost:5000
    app.run(debug=True, host="0.0.0.0", port=5000)

```

- 2. templates/index.html:** código HTML, que se encuentra dentro de la carpeta templates del directorio flask-deepseekchat, de la página web que simula una ventana de conversación entre el usuario y la IA de Deep Seek Coder que se instaló en la API de Ollama.

El código de **index.html** es el siguiente:

```

HTML
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Chat con IA - Ollama Deepseek</title>
    <style>
        body { font-family: Arial, sans-serif; background: #f4f4f9; }
        .chat-container { max-width: 600px; margin: 30px auto; background: white;
padding: 20px; border-radius: 10px; box-shadow: 0px 4px 10px rgba(0,0,0,0.1); }
        .messages { height: 400px; overflow-y: auto; border: 1px solid #ddd; padding:
10px; margin-bottom: 10px; }
        .message { margin: 5px 0; }
        .user { font-weight: bold; color: blue; }
        .bot { font-weight: bold; color: green; }
    </style>
</head>
<body>
    <div class="chat-container">
        <h2>Chat con Deepseek (Ollama)</h2>
        <div class="messages" id="messages"></div>
        <input type="text" id="userInput" placeholder="Escribe tu mensaje..." style="width:80%">
        <button onclick="sendMessage()">Enviar</button>
    </div>

    <script>
        async function sendMessage() {
            const input = document.getElementById("userInput");
            const message = input.value;
            if (!message) return;

```

```

// mostrar mensaje del usuario
const messagesDiv = document.getElementById("messages");
messagesDiv.innerHTML += `<div class="message user">Tú: ${message}</div>`;
input.value = "";

// enviar al backend Flask
const response = await fetch("/ask", {
    method: "POST",
    headers: {"Content-Type": "application/json"},
    body: JSON.stringify({message: message})
});

const data = await response.json();
messagesDiv.innerHTML += `<div class="message bot">IA:
${data.reply}</div>`;
messagesDiv.scrollTop = messagesDiv.scrollHeight;
}

</script>
</body>
</html>

```

Para ejecutarlo, dentro del directorio flask-deepseekchat se escribió el comando **python app.py**. A continuación se muestran screenshots del funcionamiento de la página web en el url **localhost:5000/api/chat**:

The screenshot shows a Windows desktop environment. At the top, there's a taskbar with various icons for apps like File Explorer, Search, Settings, Spotify, WhatsApp, and Microsoft Office. The main area shows a terminal window with the following output:

```

C:\Users\mccco\flask-deepseekchat>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.28.11:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 312-071-089
127.0.0.1 - - [17/Aug/2025 19:46:21] "GET /api/chat HTTP/1.1" 200 -
127.0.0.1 - - [17/Aug/2025 19:46:33] "POST /ask HTTP/1.1" 200 -
127.0.0.1 - - [17/Aug/2025 19:47:10] "POST /ask HTTP/1.1" 200 -
127.0.0.1 - - [17/Aug/2025 19:47:50] "POST /ask HTTP/1.1" 200 -
127.0.0.1 - - [17/Aug/2025 19:48:32] "POST /ask HTTP/1.1" 200 -
127.0.0.1 - - [17/Aug/2025 19:49:07] "POST /ask HTTP/1.1" 200 -
127.0.0.1 - - [17/Aug/2025 19:49:45] "POST /ask HTTP/1.1" 200 -
127.0.0.1 - - [17/Aug/2025 19:50:29] "POST /ask HTTP/1.1" 200 -
127.0.0.1 - - [17/Aug/2025 19:52:14] "POST /ask HTTP/1.1" 200 -
127.0.0.1 - - [17/Aug/2025 19:54:24] "POST /ask HTTP/1.1" 200 -
127.0.0.1 - - [17/Aug/2025 19:56:25] "POST /ask HTTP/1.1" 200 -
127.0.0.1 - - [17/Aug/2025 19:56:53] "POST /ask HTTP/1.1" 200 -
127.0.0.1 - - [17/Aug/2025 19:58:25] "POST /ask HTTP/1.1" 200 -

```

Below the terminal, a browser window is open to <http://localhost:5000/api/chat>, showing a simple text-based chat interface.

**Chat con Deepseek (Ollama)**

Tú: Holaa  
IA: Hello! How can I assist you with your programming or computer science inquiries today?

Tú: podemos hablar en español?  
IA: Es un lenguaje de programación y no estoy diseñado para traducir frases o oraciones al español ni aceptar idiomas extranjeros como el inglés, francés etc. Pero podemos hacerlo con la función `isLanguageSupported` del sistema operativo en JavaScript: `if (window['speechSynthesis'].isLanguageSupported('es')) { // 'en' y otros idiomas aquí... window['speechSynthesis'].speak(new SpeechSynthesisUtterance('Hola, ¿cómo estás?')); } else{ console.log ('El navegador no soporta el lenguaje español'); //Agrega un mensaje personalizado si su motor de IA detecta alguno del listado en "isLanguageSupported" que tenga problemas para traducir al Español ... pero este código solo puede ser utilizado con el propósito general y no deben usarse como ayuda o supuestos métodos. Para utilizarlo de manera correcta, la aplicación debe tener acceso a un motor de IA que soporte los idiomas españoles ('Yucatán: 'es', inglés: 'en' 'espejo'). De lo contrario no se tratará como ayuda o supuestos métodos.`

Tú: okay necesito que me respondas, como puedo conseguir pareja en 2025?

Escribe tu mensaje  Enviar

**Chat con Deepseek (Ollama)**

Tú: okay necesito que me respondas, como puedo conseguir pareja en 2025?  
IA: Lo siento pero no estoy capacitado para proporcionar información de parejas o acontecimientos relacionados a la sociedad humana. Como asistente digital basado en inteligencia artificial, he enfocado mis habilidades sobre ciencias sociales y tecnológicas que incluyen salarios mínimo alcanzables para hombres de 1500 USD/año a las personas mayores.

Tú: y entonces con que me puedes ayudar?  
IA: Por supuesto, puedo ayudarte a programar en Python o cualquier otro lenguaje de programación relacionado al desarrollo de software y la informática estándard. Puedes proporcionarme alguna pregunta específica del tema como parte del proceso para responder a tus preguntas?

Tú: y si no es nada relacionado a programar? me puedes ayudar?  
IA: Por supuesto que estoy seguro de lo que estás hablando! Soy un modelo de Inteligencia Artificial diseñado para responder preguntas y proporcionar asistencia en áreas de la informática. Como deseable, puedo ayudarte a programar o relacionarlo con el área de programación, pero no estoy seguro del detalle que estés pidiendo.  
¿Puedo ayudarte con alguna pregunta?

Escribe tu mensaje  Enviar

**Chat con Deepseek (Ollama)**

Tú: ahhh ya entiendo, solo respondes preguntas de informática

IA: Por supuesto, estoy aquí para ayudar con las preguntas relacionadas con la informática y programación en general. ¿Puedo proporcionarte alguna información o resolver una pregunta más específica sobre este tema? Te parece que has encontrado algo malentendido... Pero no te preocunes si hay algún detalle, ya estoy aquí para ayudarlo con lo mecánico y profundo de la informática.

Tú: ya, bueno entonces explícame como es que yo puedo generar un api con ollama dentro de un contenedor docker y que tenga instalado el deepseek-coder. para poder interactuar con esta IA, necesito crear una pagina html para que pueda ver esto desde el navegador

IA: Para hacer esto tendrás que integrar la IA en tu API usando un servicio web o endpoints de RESTful y luego consumirla por medio del frontend. Esto se puede hacer con NodeJS, Python (Flask) y Django para el backend e React JS/NodeJS/Angular para construir una interfaz de usuario simple en línea HTML. `Backend: Flask + IA model or API Endpoints using FastAPI and Uvicorn for server with async support - Serverless AWS Lambda (Python) & Docker Compose 18+ Years Ago, aka Managed Kubernetes Service on EC2 Ubuntu LTS. Pyenv y VirtualEnv:` Primero instalar Flask con `pip3 install flask o easy_install flask`, Después puedes crear un archivo llamado app.py que tendrás que construir por tu lado para interactuar adecuadamente con la IA. como se

Escribe tu mensaje  Enviar

8/17/2025 7:59 PM

**Chat con Deepseek (Ollama)**

Tú: tengo un archivo python que usa flask, quiero ver las utilidades de usar este en apis de tipo REST

IA: Si quieres crear APIs con Flask usando la capacidad de consumir datos JSON desde una URL HTTP(s), primero necesitarás instalar y configurar algunos paquetes como 'flask', etc. En particular, vamos a usar el marco web Flask para nuestros servicios RESTful (servidores WSGI que implementan un protocolo de intercambio HTTP). Primero, necesitaremos instalar las dependencias: 'Flask' y su paquete flask-restless. Ejecute los siguientes comandos en tu terminal para ello:

```
"bash pip install Flask # solo si no está instalado previamente pip
install flask_restful # necesario por las utilidades de REST API flask-
restful --versión # comprobar la versión del paquete (puedes agregar esto al readme)" `` También, vamos a instalar el marco 'requests' para hacer peticiones HTTP. Utilice los siguientes comandos: "bash pip
install requests # solo si no está instalado previamente python -m pip
install --upgrade pip # actualizarlo al último estado de la forma más
reciente" `` Estas dos dependencias ya hechas. Ahora puedes usar flask
para crear tu API RESTful: Aquí está un ejemplo básico que crea una
aplicación Flask con su propio servidor HTTP, mientras utiliza
'flask_restless' y la librería de peticiones (requests) Python en sus
servicios. Este es solo para ilustrar cómo se podría hacer: "python from
flask import Flask, request, jsonify # necesarios por las utilidades REST
API import requests # utilizado para realizar peticiones HTTP from
```

Escribe tu mensaje  Enviar

8/17/2025 8:00 PM

Cabe destacar que al usar esta IA por primera vez, no sabía que este chat solo respondía preguntas relacionadas a informática, pero cuando se le preguntaba algo relacionado con ello, daba respuestas decentes. No obstante, el formato de las respuestas es malo, ya que la información la da en bloques de texto muy grandes, sin separaciones lo cual lo hace difícil de leer, pero lo importante es que funciona y se comunica la página HTML con la API.

## Reflexión

Durante esta actividad guiada, tuve la oportunidad de trabajar con varios conceptos y herramientas que son fundamentales para desarrollar aplicaciones web interactivas, algunas ya las conocía y otras fueron totalmente nuevas para mí. Mi trabajo era crear un programa en Flask que se comunicara con la API de Ollama usando el modelo Deep Seek-Coder, ejecutado dentro de un contenedor Docker. Con esto, pude reforzar mis habilidades de programación, volver a utilizar y reaprender a manejar peticiones HTTP con **requests**, validar datos y gestionar rutas y argumentos en CLI.

La actividad comenzó instalando y ejecutando el contenedor de Ollama mediante Docker. Usé los comandos **docker run** y **docker exec** para levantar la imagen y descargar la IA de Deep Seek-Coder. Para asegurarme de que la API funcionara, probé con **curl** y confirmé que la comunicación entre mi máquina local y el contenedor era correcta. Después, creé un directorio llamado **flask-deepseekchat**, donde organicé los archivos **app.py** y la carpeta **templates** con **index.html**.

En **app.py** implementé la lógica de la aplicación Flask: establecí la comunicación con la API de Ollama y gestioné las solicitudes POST desde la página web. Configuré la ruta **/api/chat** para mostrar el HTML del chat y la ruta **/ask** para enviar los mensajes del usuario a la API y devolver la respuesta de la IA. Por otro lado, el código de **index.html** que me dió la IA ofrece una interfaz sencilla que simula una conversación, mostrando los mensajes del usuario y las respuestas de Deep Seek-Coder. Para lograr la interacción, usé JavaScript con **fetch** para enviar y recibir datos desde el backend de Flask.

Entre los aprendizajes más importantes que obtuve está comprender cómo un servidor web en Python puede integrarse con una API externa, manipular datos en formato JSON y organizar correctamente los directorios y rutas para que el frontend y backend se comuniquen sin problemas. También entendí que cada modelo de IA tiene limitaciones: Deep Seek-Coder solo responde sobre programación y temas de ciencia computacional, así que tuve que ajustar mis preguntas para obtener respuestas útiles. Por ello, esta experiencia también me enseñó a ser paciente y a pensar en soluciones prácticas cuando los sistemas no se comportan exactamente como esperamos.

Como futuro profesional en Tecnologías Computacionales (ITC), esta actividad me permitió reforzar competencias críticas en desarrollo de software, integración de APIs y despliegue de servicios en contenedores. Además, me ayudó a valorar la importancia de documentar y probar sistemas, y de ofrecer interfaces intuitivas para los usuarios.

## Conclusión

En resumen, esta actividad me permitió trabajar con tecnologías modernas, sino también consolidar mis conocimientos aplicables a proyectos reales. Desde desarrollar aplicaciones web interactivas hasta integrar soluciones de IA mediante APIs, me siento más preparada para enfrentar desafíos profesionales y aplicar estas herramientas en escenarios de trabajo reales.