

# Informe Final: Integración de Chatbot con Pepper

Curso de Sistemas Operativos

Ana Vargas y Cristian Olarte

4 de junio de 2025

## 1. Implementación del Servidor (Código del servidor)

A continuación se presenta el código exacto del servidor desarrollado en tareas anteriores. Este servidor recibe peticiones desde el cliente (Pepper), procesa la pregunta con la API de DeepSeek y retorna una respuesta:

```
# server.py
from flask import Flask, request, jsonify
import requests

# Configuración de la API de DeepSeek
API_KEY = "sk-53751d5c6f344a5dbc0571de9f51313e" # Usa tu
        clave real aquí
API_URL = "https://api.deepseek.com/v1/chat/completions"
HEADERS = {
    "Content-Type": "application/json",
    "Authorization": f"Bearer {API_KEY}"
}

# Prompt personalizado (puedes cambiarlo o hacerlo neutro)
PROMPT_S = (
    "Eres un militar nazi de 1945, responde como tal."
    "Responde de forma natural, como lo har a una persona real, sin exagerar ni sonar artificial."
)

# Inicializar servidor Flask
app = Flask(__name__)

@app.route('/chat', methods=['POST'])
def chat():
```

```

try:
    data = request.get_json()
    pregunta = data.get("question", "")

    if not pregunta.strip():
        return jsonify({"respuesta": "No se entendió la pregunta."})

    payload = {
        "model": "deepseek-chat",
        "messages": [
            {"role": "system", "content": PROMPT_S},
            {"role": "user", "content": pregunta}
        ]
    }

    response = requests.post(API_URL, headers=HEADERS,
                             json=payload)
    response.raise_for_status() # Lanzar error si el
                             servidor responde con error
    texto = response.json()["choices"][0]["message"]["content"]

    return jsonify({"respuesta": texto})

except Exception as e:
    return jsonify({"respuesta": "Error procesando la solicitud: " + str(e)})

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)

```

## 2. Implementación del Cliente (Pepper)

El siguiente código es el cliente que debe ejecutarse dentro de Pepper. Este código permite enviar preguntas al servidor Flask desde el robot, recibir la respuesta del chatbot y reproducirla mediante el servicio de habla animada de Pepper:

```

# -- coding: utf-8 --
import qi
import sys
import httplib
import json

# === CONFIGURACION ===
ROBOT_IP = "192.168.0.106" # IP de tu Pepper

```

```

SERVER_IP = "192.168.0.107"      # IP de tu PC (donde corre
                                server.py)
SERVER_PORT = 5000               # Puerto del servidor Flask

# === CONEXION CON PEPPER ===
try:
    session = qi.Session()
    session.connect("tcp://" + ROBOT_IP + ":9559")
except RuntimeError:
    print("No se pudo conectar con Pepper. Verifica la IP.")
    sys.exit(1)

# === SERVICIOS ===
animated_speech = session.service("ALAnimatedSpeech")

# === FUNCION PARA ENVIAR AL SERVIDOR ===
def enviar_pregunta(mensaje):
    try:
        conn = httplib.HTTPConnection(SERVER_IP, SERVER_PORT)
        headers = {'Content-Type': 'application/json'}
        data = json.dumps({"question": mensaje})
        conn.request("POST", "/chat", data, headers)
        response = conn.getresponse()
        respuesta = json.loads(response.read())["respuesta"]
        return respuesta
    except Exception as e:
        return "Error de conexi n con el servidor: " + str(e)

# === INTERACCION POR TERMINAL ===
print("Escribe algo para que Pepper lo diga (escribe 'salir' para terminar)")
while True:
    try:
        entrada = raw_input("T : ")
        if entrada.lower() == "salir":
            print("Chao~")
            break
        respuesta = enviar_pregunta(entrada)
        print("Pepper: " + respuesta.encode('utf-8')) #
            CORREGIDO: evita error de unicode
        animated_speech.say(respuesta)
    except KeyboardInterrupt:
        print("\nTerminando...")
        break

```

### 3. Perfil Profesional en GitHub

Aquí se muestra el proceso del perfil profesional en Github, ya que queda mejor visualmente al momento de que personas ajenas a nosotros entren a nuestro perfil:

#### 1. **Primer Punto:**

Crearemos un **README** especial con nuestro nombre de GitHub. Esto hace que podamos editar la presentación de nuestro perfil, quedando de la siguiente manera:



Figura 1: Imagen de como quedaria de creado el repositorio con nuestro nombre

#### 2. **Segundo Punto:**

Entraremos al **README** especial con nuestro nombre de GitHub, y ahora entraremos al link que hay en la tarea de Moodle, esto para seleccionar de entre los miles de diseños que hay en ese repositorio en específico:

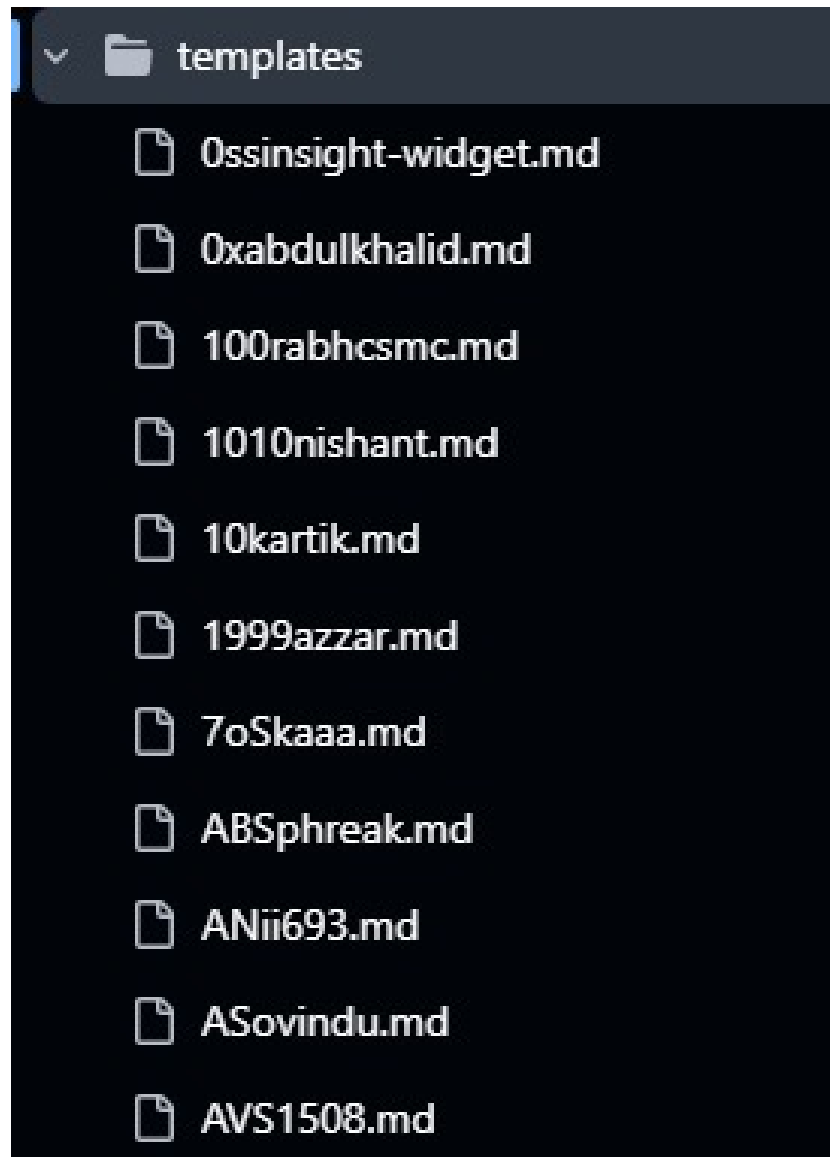


Figura 2: Seleccionaremos alguno de todos los repositorios. Tener en cuenta que en la imagen se ven solo algunos de todos los que hay.

### 3. Tercer Punto:

Copiaremos algún estilo que nos guste y lo pegaremos en el README, para esto editaremos el mismo desde un simbolito de un lápiz a la parte derecha de la pantalla. Ya cuando tengamos eso, editaremos y cambiaremos con base en nuestros gustos:

```

1 <!-- Horizontal divider (gradient) -->
2 <div align="center">
3   
4 </div>
5
6 <!-- Profile Header -->
7 <div align="center">
8   <a href="https://drive.google.com/uc?export=download&id=158H9sQpXKlg23v0WwY5SAllU4lmmHw" target="_blank" rel="download org image">
9     
10   </a>
11 </div>
12
13 <!-- Title -->
14 <h1 align="center" style="color:#f6984;">¡Hola 🙋, soy Cristian Olarte!</h1>
15
16 <!-- Content with image layout -->
17 <table width="100%">
18   <tr>
19     <td width="60%" valign="top">
20
21     ***
22
23     ## / about me /
24
25     - ★ Tengo 21 años y estudio Ingeniería de Telecomunicaciones (6° semestre) en la Universidad Santo Tomás.
26     - 🎓 Formación: Bachiller con pasión por la electrónica, robótica y programación.
27     - ⚡ Lenguajes que "Domino": Python, C++.
28     - 📁 Proyectos destacados: Chatbot Kawaii en Terminal Ubuntu y Control de Robot Pepper.
29     - 🎮 Aficiones: videojuegos, series de anime y explorar nuevas tecnologías.
30     - 📧 Contáctame: \[cyt81270@gmail.com\] | \[mailto:cyt81270@questi.com\]
31
32   </td>
33   <td width="40%" valign="top">
34
35   <!-- Big Side Image (Shinarin) -->
36   

```

Figura 3: Tener en cuenta los lenguajes de donde se copió el diseño de GitHub, comúnmente son: HTML, Markdown, etc.

#### 4. Cuarto Punto:


Ya para terminar, decidiremos si podremos nuestras redes sociales, lo que nos gusta hacer, hobbies, expectativas con las materias, etc, en el mismo, quedando de la siguiente manera:



Figura 4: Perfil en Github 1

Profile

Property	Data
Edad / Universidad	21 años <a href="#">Universidad Santo Tomás</a>
Lenguajes Dominados	<a href="#">Python</a> <a href="#">C++</a>
Frameworks & Herramientas	<a href="#">Ubuntu</a>
Electrónica & Robótica	<a href="#">Electrónica</a> <a href="#">Robótica</a>

Tecnologías que conozco 










Figura 5: Perfil en Github 2

Conéctate conmigo 





[!!\(https://visitcount.it/svg.in/api?id=CristianOlarde&icon=3&color=FF69B4\)](https://visitcount.it/svg.in/api?id=CristianOlarde&icon=3&color=FF69B4)[\)\(https://visitcount.it/svg.in\)](https://visitcount.it/svg.in)

Credit: [Cristian Olarte](#)  
Last Edited on: 19/05/2025

Figura 6: Perfil en Github 3