Informe: ChatBot y Despliegue

Marco Gómez, Juan P. Pedraza

3 de septiembre de 2025

Índice

1. Introducción

En la actualidad, los *chatbots* se han consolidado como una de las aplicaciones más prácticas de la inteligencia artificial conversacional. Estos sistemas permiten la interacción entre humanos y computadoras mediante lenguaje natural, ofreciendo respuestas automáticas, soporte técnico, tutoría académica, entre otros usos. Su implementación se ha visto potenciada por los avances en modelos de lenguaje de gran escala (LLMs, por sus siglas en inglés), los cuales son capaces de generar respuestas coherentes y contextualizadas.

Para el desarrollo del presente proyecto se utilizó **Streamlit** como plataforma de despliegue. Streamlit es una librería de Python que facilita la construcción de interfaces web de manera ágil, sin necesidad de conocimientos avanzados en desarrollo frontend. Gracias a su simplicidad, permite implementar aplicaciones interactivas en cuestión de minutos, lo que lo convierte en una herramienta ideal para integrar chatbots y compartirlos de forma accesible a través de la web.

Un elemento fundamental en la interacción con modelos de lenguaje es el**prompt**. Este término hace referencia al mensaje inicial o instrucción que orienta la forma en que el modelo debe responder. El diseño del prompt influye directamente en la calidad, claridad y relevancia de las respuestas generadas. En este proyecto, se emplearon prompts cuidadosamente estructurados para guiar el estilo de comunicación del chatbot, controlar la extensión de las respuestas y asegurar la pertinencia del contenido ofrecido a los usuarios.

En resumen, este trabajo aborda el diseño, desarrollo y despliegue de un chatbot implementado en Python, con interfaz en Streamlit y basado en técnicas de ingeniería de prompts, con el fin

de demostrar cómo estas tecnologías pueden integrarse en aplicaciones prácticas de comunicación interactiva.

2. Chatbot con DeepSeek

Eres un asistente experto en electrónica, claro y breve. Responde en español. Este es el prompt que se uso para elaborar este chatbot y el codigo se evidencia a continuacion

```
import os
import requests
import streamlit as st
from dotenv import load_dotenv

# Cargar variables de entorno (solo para desarrollo local)
load_dotenv()

# ===== CONFIGURACI N =====
API_URL = "https://api.deepseek.com/v1/chat/completions"

def get_secret(key, default=None):
    if st.secrets and key in st.secrets:
        return st.secrets[key]
    return os.environ.get(key, default)

API_KEY = get_secret("DEEPSEEK_API_KEY")
MODEL = get_secret("MODEL", "deepseek-chat")
SYSTEM_PROMPT = get_secret(
    "SYSTEM_PROMPT",
    "Eres un asistente experto en electr nica, claro y breve. Responde en
        espa ol."
)

if not API_KEY:
    st.error("       No se encontr  la clave DEEPSEEK_API_KEY.
       Config rala en Streamlit Secrets.")
    st.stop()

# ===== INTERFAZ =====
st.set_page_config(page_title="Chatbot DeepSeek", page_icon="      ")
st.title("       Chatbot DeepSeek Personalizado")

if "history" not in st.session_state:
    st.session_state.history = [{"role": "system", "content":
       SYSTEM_PROMPT}]

def chat_with_deepseek(prompt):
    messages = st.session_state.history + [{"role": "user", "content":
       prompt}]
    payload = {"model": MODEL, "messages": messages, "temperature": 0.3}
    headers = {"Authorization": f"Bearer {API_KEY}", "Content-Type": "
       application/json"}
    try:
        r = requests.post(API_URL, headers=headers, json=payload, timeout
```

```
            =60)
        r.raise_for_status()
        data = r.json()
        response = data["choices"][0]["message"]["content"]
        return response
    except Exception as e:
        return f"Error: {e}"

# Mostrar historial de chat
for msg in st.session_state.history:
    if msg["role"] == "system":
        continue
    st.markdown(f"**{'T  ' if msg['role']=='user' else 'Bot'}:** {msg['
        content']}")

# Entrada de usuario
user_input = st.chat_input("Escribe tu mensaje:")
if user_input:
    st.session_state.history.append({"role": "user", "content": user_input
        })
    response = chat_with_deepseek(user_input)
    st.session_state.history.append({"role": "assistant", "content":
        response})
    st.rerun()

# Bot n de reinicio
if st.button("Reiniciar conversaci n"):
    st.session_state.history = [{"role": "system", "content":
        SYSTEM_PROMPT}]
    st.rerun()
```

3. Chatbot sobre sistemas digitales con voz

Eres un tutor que solo explica qué son los sistemas digitales. continuación

```
import os
import requests
import streamlit as st
from gtts import gTTS
from dotenv import load_dotenv

# Cargar variables de entorno
load_dotenv()

API_URL = "https://api.deepseek.com/v1/chat/completions"

def get_secret(key, default=None):
    if st.secrets and key in st.secrets:
        return st.secrets[key]
    return os.environ.get(key, default)

API_KEY = get_secret("DEEPSEEK_API_KEY")
MODEL = get_secret("MODEL", "deepseek-chat")
```

```
SYSTEM_PROMPT = "Eres un tutor que solo explica qu  son los sistemas
    digitales."

if not API_KEY:
    st.error("       Falta DEEPSEEK_API_KEY en secrets.")
    st.stop()

st.set_page_config(page_title="Chatbot con Voz", page_icon="         ")
st.title("          Chatbot:   Qu   son los Sistemas Digitales?")

if "history" not in st.session_state:
    st.session_state.history = [{"role": "system", "content":
        SYSTEM_PROMPT}]

def chat_with_deepseek(prompt):
    messages = st.session_state.history + [{"role": "user", "content":
        prompt}]
    payload = {"model": MODEL, "messages": messages, "temperature": 0.3}
    headers = {"Authorization": f"Bearer {API_KEY}", "Content-Type": "
        application/json"}
    try:
        r = requests.post(API_URL, headers=headers, json=payload, timeout
            =60)
        r.raise_for_status()
        data = r.json()
        return data["choices"][0]["message"]["content"]
    except Exception as e:
        return f"Error: {e}"

# Mostrar historial
for msg in st.session_state.history:
    if msg["role"] != "system":
        st.markdown(f"**{'T  ' if msg['role']=='user' else 'Bot'}:** {msg
            ['content']}")

# Entrada de usuario (Enter para enviar)
user_input = st.chat_input("Escribe tu pregunta sobre sistemas digitales
    ...")

if user_input:
    st.session_state.history.append({"role": "user", "content": user_input
        })
    response = chat_with_deepseek(user_input)
    st.session_state.history.append({"role": "assistant", "content":
        response})

    # Generar voz
    tts = gTTS(text=response, lang='es')
    tts.save("respuesta.mp3")
    st.audio("respuesta.mp3")

# Bot n para reiniciar
if st.button("Reiniciar conversaci n"):
```

```
    st.session_state.history = [{"role": "system", "content":
        SYSTEM_PROMPT}]
    st.rerun()
```

4. Paso a Paso

4.1. Paso 1

Para empezar a generar el chatbot lo primordial es tener una cuenta en GitHub y crear un repositorio donde se encuentre lo comentado a continuacion

4.2. Paso 2

Se configura el entorno virtual debido a que los sistemas de gitcloud ya tienen un entorno virtual adaptable a los requerimientos que uno necesite

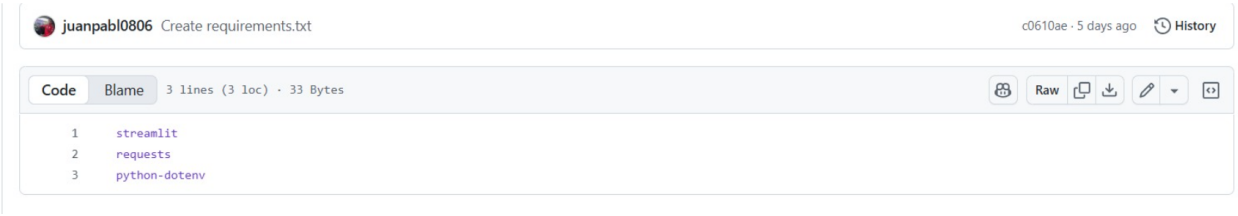

Figura 1: requerimientos solicitados

4.3. Paso 3

Se agrega un archivo donde se encuentre el codigo nombrandolo de forma que termine en .py (extension de python) para que al usar streamlit pueda detectar el archivo que requiere para hacer el chatbot.

Figura 2: Codigo subido a github

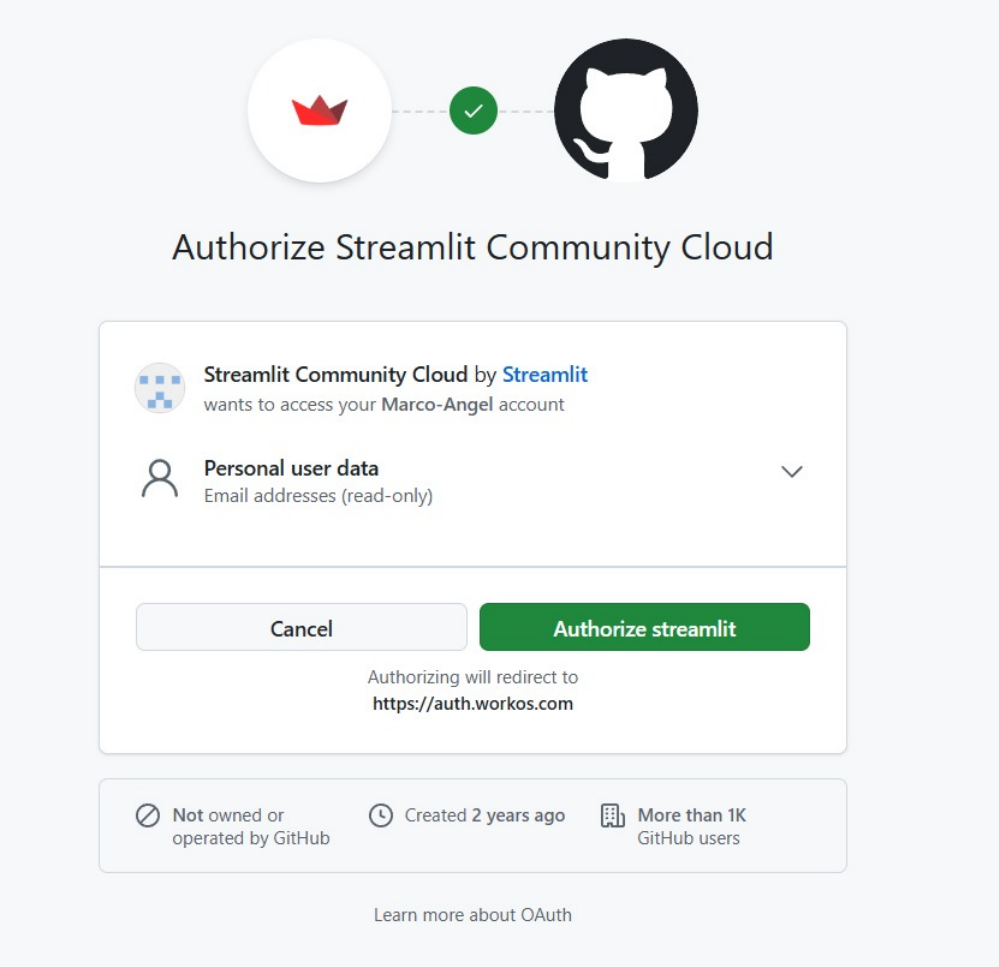

Figura 4: Autorizar vinculacion

4.4. Paso 4

A continuacion se ingresa a Streamlit y se inicia sesión vinculando directamente tu cuenta de GitHub (en este paso te van a enviar un correo con un codigo de verificacion)

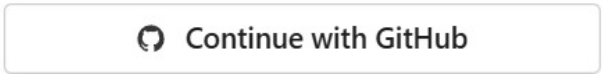

Figura 3: boton de vinculacion con github

4.5. Paso 5

Al hacer los pasos anteriores ya estaras en la pagina principal de Streamlit, en la esquina superior derecha se encuentra el boton para generar tu ChatBot. Luego te detectara todos los repositorios que tienes y seleccional el que creaste para este proposito si todo se encuentra en orden te aparecere un cartel en verde diciendo "Domain is available"

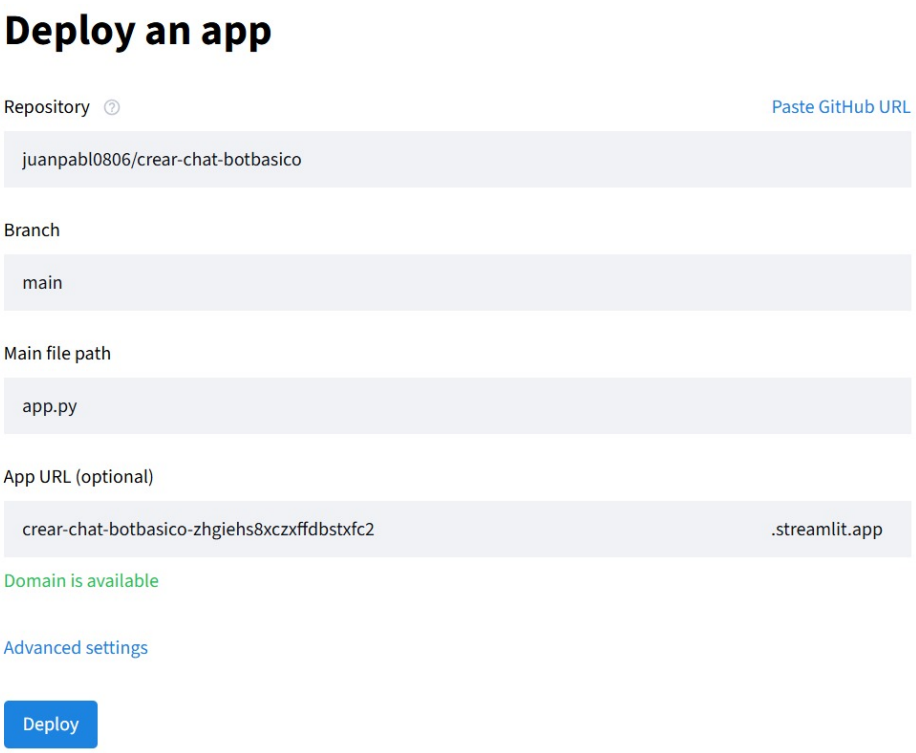

Figura 6: Evidencia del ChatBot Basico

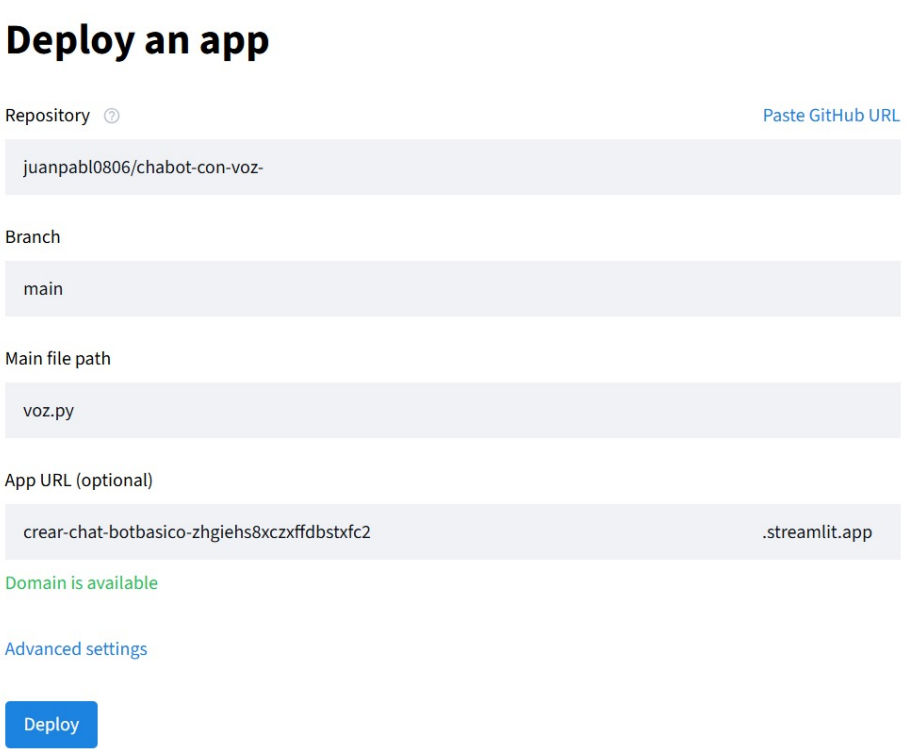

Figura 5: Evidencia del ChatBot con voz

4.6. Paso 6

Y por ultimo paso se oprime el boton de deploy y esperas a que tu ChatBot aparezca de forma correcta y como la deseas, de lo contrario igual puedes editar tu Chatbot desde Streamlit (abriendo VS code para mayor eficiencia)

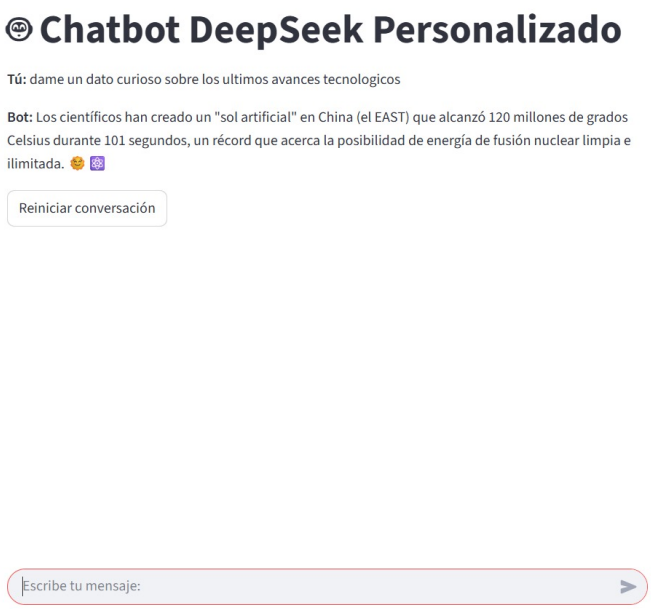

Figura 7: ChatBot Basico listo

Figura 8: ChatBot con voz listo

4.7. Paso extra

Si lo deseas para mayor seguridad puedes configurar datos personales o importantes en la opcion de ajustes o ajustes avanzados en secretos (En este caso es el APIKEY); debido a que es Open Source todo el mundo puede inspeccionar la pagina y ver el codigo.

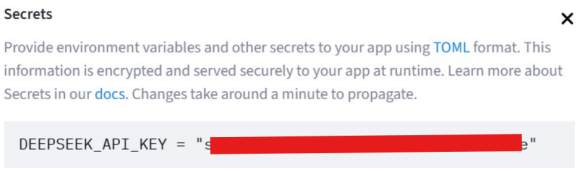

Figura 9: Opcion de secretos