

Proyecto 1: Conexión con Pepper y Desarrollo de un Chatbot Educativo con Dashboard Interactivo

Yojan Contreras / Cristian Losada

Septiembre 2025

Abstract

Este proyecto presenta el proceso de conexión y programación del robot Pepper, incluyendo la transferencia de archivos, ejecución de código en Python y despliegue de recursos multimedia para una exposición educativa. Adicionalmente, se implementó un chatbot con personalidad inspirada en Thanos, integrado en un **dashboard interactivo desarrollado en Streamlit**. El chatbot responde únicamente sobre temáticas tecnológicas seleccionadas: *CRISPR y biocomputación*, *cohetes reutilizables* y *NFTs de utilidad*. Finalmente, se detalla el despliegue en la nube mediante Hugging Face, asegurando accesibilidad y correcto funcionamiento en cualquier dispositivo.

Contents

1	Conexión inicial con Pepper y preparación de archivos	3
1.1	Acceso al robot mediante SSH	3
1.2	Creación de carpeta para almacenar imágenes	3
1.3	Transferencia de imágenes al robot	4
1.4	Verificación de archivos en el robot	4
2	Creación del archivo y edición con Nano	4
2.1	Primera parte del código: conexión con Pepper	5
2.2	Segunda parte del código: definición de imágenes y recursos	6
2.3	Tercera parte del código: ejecución de la exposición	6
3	Chatbot	7
4	Herramientas Utilizadas	7
5	Proceso de Creación	7
5.1	Creación del entorno de trabajo	7
5.2	Instalación de dependencias	8
5.3	Creación de archivos principales	8
5.4	Ejecución local	9

5.5	Control de versiones con GitHub	9
6	Funcionamiento del Chatbot	9
6.1	Definición de la personalidad	9
6.2	Interacción con el usuario	10
6.3	Procesamiento de la respuesta	10
6.4	Presentación de la respuesta	10
7	Creación del Dashboard	10
7.1	Creación del Entorno Virtual	10
7.2	Subir Entorno en Hugging Face	13
8	Funcionamiento del Dashboard	17
8.1	Estructura general	17
8.2	Diseño y estilo	18
8.3	Contenido multimedia (Columna 1)	18
8.4	Información y novedades (Columna 2)	18
8.5	Chatbot interactivo (Columna 3)	18
8.6	Procesamiento de respuestas	19
8.7	Presentación de las respuestas	19
8.8	Footer	19
9	Dashboard Online	19
10	Conclusión	19

1 Conexión inicial con Pepper y preparación de archivos

Para poder interactuar con el robot Pepper es indispensable estar conectado a la misma red Wi-Fi que el robot. De esta manera se asegura la comunicación por medio del protocolo `ssh`, el cual nos permite acceder de forma remota a su sistema operativo.

1.1 Acceso al robot mediante SSH

Una vez identificada la dirección IP del robot, utilizamos el siguiente comando en la terminal de nuestro computador para iniciar sesión:

```
ssh nao@<ip_Pepper>
```

Este comando abre una sesión remota en el sistema de Pepper bajo el usuario `nao`.

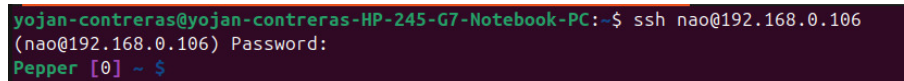
A terminal window showing the process of connecting to the Pepper robot via SSH. The prompt is 'yojan-contreras@yojan-contreras-HP-245-G7-Notebook-PC:~\$'. The command 'ssh nao@192.168.0.106' is entered. The output shows '(nao@192.168.0.106) Password:' followed by a prompt 'Pepper [0] ~ \$'.

Figure 1: Conexión remota al robot Pepper mediante SSH

1.2 Creación de carpeta para almacenar imágenes

Antes de transferir archivos al robot, es recomendable crear una carpeta en el sistema de Pepper donde guardaremos las imágenes. Para ello, una vez dentro de la sesión `ssh`, ejecutamos:

```
mkdir ~/imagenes_pepper
```

Con este comando se genera una nueva carpeta llamada `imagenespepper` en el directorio personal del usuario `nao`.

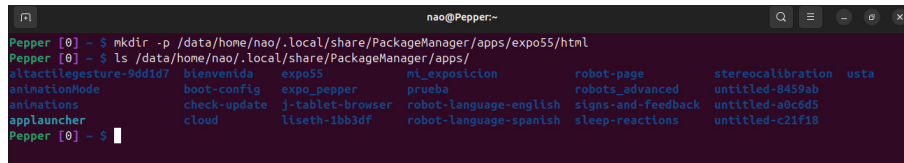
A terminal window titled 'nao@Pepper~' showing the execution of 'mkdir -p /data/home/nao/.local/share/PackageManager/apps/expo55/html'. The prompt is 'Pepper [0] ~ \$'. Below the command, a list of files and directories is displayed in a grid-like format, including 'altactilegesture-9dd1d7', 'bienvenida', 'expo55', 'ni_exposicion', 'robot-page', 'stereocalibration', 'usta', 'animationMode', 'boot-config', 'expo_pepper', 'prueba', 'robots_advanced', 'untitled-8459ab', 'animations', 'check-update', 'j-tablet-browser', 'robot-language-english', 'signs-and-feedback', 'untitled-a0c6d5', 'applauncher', 'cloud', 'liseth-1bb3df', 'robot-language-spanish', 'sleep-reactions', and 'untitled-c21f18'.

Figure 2: Creación de carpeta para imágenes

1.3 Transferencia de imágenes al robot

Desde nuestro computador utilizamos el comando `scp` para copiar las imágenes deseadas hacia la carpeta recién creada en Pepper:

`scp ...ruta de la carpeta donde tenemos las imagenes que queremos transferir a pepper`

Este comando envía los archivos al directorio remoto del robot.

```
yojan-contreras@yojan-contreras-HP-245-G7-Notebook-PC:~$ scp /home/yojan-contreras/Música/expopepper/*.png nao@192.168.0.106:/data/home/nao/.local/share/PackageManager/apps/expo_pepper/html/
(nao@192.168.0.106) Password:
3_Descargamos-el-archivo.png                                100% 136KB 252.8KB/s 00:00
4_Buscamos-archivo-en-descargas.png                         100% 27KB 147.6KB/s 00:00
5_Abrimos-choregraphe.png                                  100% 207KB 371.4KB/s 00:00
6_Creamos_una_coreografia.png                               100% 110KB 404.7KB/s 00:00
yojan-contreras@yojan-contreras-HP-245-G7-Notebook-PC:~$
```

Figure 3: Transferencia de archivos

1.4 Verificación de archivos en el robot

Para confirmar que las imágenes fueron copiadas correctamente, dentro de la sesión en Pepper se ejecuta:

`ls ~/ruta de la carpeta a la cual subiremos las imagenes`

Este comando listará los archivos contenidos en la carpeta `imagenespepper`.

```
Pepper [0] ~ $ cd /data/home/nao/.local/share/PackageManager/apps/expo55/html
Pepper [0] /data/home/nao/.local/share/PackageManager/apps/expo55/html $ ls
3_Descargamos-el-archivo.png 4_Buscamos-archivo-en-descargas.png 5_Abrimos-choregraphe.png 6_Creamos_una_coreografia.png
Pepper [0] /data/home/nao/.local/share/PackageManager/apps/expo55/html $
```

Figure 4: Verificación de archivos

2 Creación del archivo y edición con Nano

Para comenzar, dentro del robot Pepper creamos un nuevo archivo Python con el comando:

```
touch prueba55.py
nano prueba55.py
```

Esto abre el editor `nano` donde escribiremos el código que controlará la exposición de Pepper.

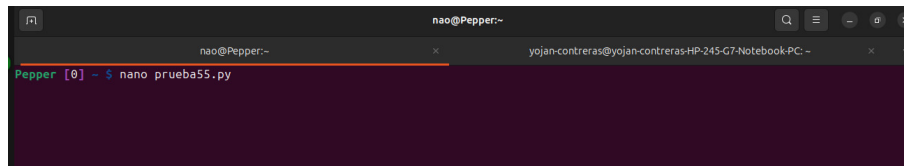


Figure 5: Creacion archivo

2.1 Primera parte del código: conexión con Pepper

```
# -- coding: utf-8 --
import qi
import time

PEPPER_IP = "127.0.0.1" # Ejecutamos dentro del robot
PORT = 9559

# --- Conexión con Pepper ---
session = qi.Session()
try:
    session.connect("tcp://{}:{}".format(PEPPER_IP, PORT))
except RuntimeError:
    print("No se pudo conectar localmente.")
    raise SystemExit(1)

tablet = session.service("ALTabletService")
animated_speech = session.service("ALAnimatedSpeech")
motion = session.service("ALMotion")
posture = session.service("ALRobotPosture")

# --- Ruta base de imágenes ---
base = "http://198.18.0.1/apps/expo_pepper/"

ings = {
```

Figure 6: Bloque 1

En este bloque se importan las librerías necesarias (`qi` y `time`) y se define la dirección IP y el puerto de conexión. Posteriormente se establece la sesión con Pepper utilizando `qi.Session()`. Si la conexión falla, el programa se detiene con un mensaje de error.

Este paso es fundamental porque permite que el script pueda comunicarse con los servicios internos del robot, como voz, movimiento y pantalla.

2.2 Segunda parte del código: definición de imágenes y recursos

```
# --- Ruta base de imágenes ---
base = "http://198.18.0.1/apps/expo_pepper/"

imgs = {
  "intro": base + "ola2.png",
  "futuro": base + "2.png",
  "crispr": base + "3.png",
  "comparacion": base + "4.png",
  "coquete": base + "5.png",
  "carrera": base + "6.png",
  "nft": base + "7.png",
  "usos": base + "8.png",
  "extra1": base + "9.png",
  "extra2": base + "10.png",
  "final": base + "11.png",
  "negro": base + "12.png" # última en negro SOLO al final
}
```

Figure 7: Bloque 2

Aquí se define la ruta base donde se encuentran las imágenes previamente cargadas en Pepper. Después, se construye un diccionario `imgs` que asocia nombres de secciones (por ejemplo, `intro`, `crispr`, `coquete`) con las imágenes correspondientes.

De esta manera, el programa puede llamar a cada imagen de forma organizada durante la exposición, mostrándola en la tableta del robot mientras habla.

2.3 Tercera parte del código: ejecución de la exposición

```
tablet.showImageNoCache(imgs["intro"])
animated_speech.say("^start(animations/Stand/Gestures/Hey_1) ¡Muy buenas tardes! Bienvenidos a Pepper News, el noticiero que s
time.sleep(2)

tablet.showImageNoCache(imgs["futuro"])
animated_speech.say("^start(animations/Stand/Gestures/Explain_1) Hoy tenemos un programa especial sobre el futuro de la tecnol
time.sleep(2)

# ----- Tema 1: CRISPR -----
tablet.showImageNoCache(imgs["crispr"])
animated_speech.say("^start(animations/Stand/Gestures/Enthusiastic_4) Primera noticia: CRISPR. Imaginen unas tijeras microscóp
time.sleep(2)

animated_speech.say("Con CRISPR un día podríamos curar enfermedades hereditarias... o quizás ¡lograr que la pña si cambi
time.sleep(2)

tablet.showImageNoCache(imgs["comparacion"])
animated_speech.say("^start(animations/Stand/Gestures/ShowSky_3) Compáren esto: las computadoras de silicio necesitan tonelada
time.sleep(2)

# ----- Tema 2: Cohetes -----
tablet.showImageNoCache(imgs["coquete"])
animated_speech.say("^start(animations/Stand/Gestures/ShowSky_1) Pasamos ahora a las noticias espaciales. Antes, lanzar un coh
time.sleep(2)
```

Figure 8: Bloque 3

En esta parte se integran las imágenes y la narración de Pepper. Con `tablet.showImageNoCache()` se muestran las imágenes en la tableta, mientras que con `animated_speech.say()` Pepper pronuncia los textos acompañados de gestos animados.

Además, se utilizan pausas con `time.sleep()` para sincronizar los cambios entre voz, gestos e imágenes, logrando que la exposición sea fluida y dinámica.

El resto del código sigue la misma lógica:

- Se muestra una nueva imagen en la tableta de Pepper.
- Pepper narra el texto asociado a esa imagen.
- Se ejecutan algunos gestos animados para acompañar la narración.
- Se insertan pausas para dar naturalidad a la exposición.

La única variación entre secciones es la imagen que se carga y las frases o gestos que Pepper utiliza, manteniendo siempre la misma estructura básica.

3 Chatbot

El objetivo de este proyecto fue crear un chatbot educativo que pueda responder preguntas relacionadas con las **Novedades tecnológicas**, con un estilo de comunicación inspirado en el personaje ficticio Thanos

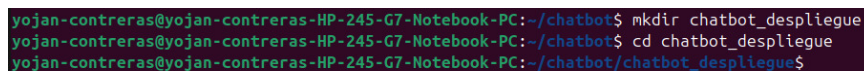
4 Herramientas Utilizadas

- Python 3.12
- Streamlit
- streamlit-mic-recorder
- LangChain y langchain-deepseek
- gTTS (Google Text-to-Speech)
- GitHub
- Streamlit Cloud o Hugging face

5 Proceso de Creación

5.1 Creación del entorno de trabajo

1. Crear carpeta: `mkdir chatbot_pepper`
2. Entrar en carpeta: `cd chatbot_pepper`



```
yojan-contreras@yojan-contreras-HP-245-G7-Notebook-PC:~/chatbot$ mkdir chatbot_despliegue
yojan-contreras@yojan-contreras-HP-245-G7-Notebook-PC:~/chatbot$ cd chatbot_despliegue
yojan-contreras@yojan-contreras-HP-245-G7-Notebook-PC:~/chatbot/chatbot_despliegue$
```

Figure 9: Creacion carpeta

3. Crear entorno virtual: `python3 -m venv venv`
4. Activar entorno virtual:
 - Linux/Mac: `source venv/bin/activate`

```
yojan-contreras@yojan-contreras-HP-245-G7-Notebook-PC:~/chatbot/chatbot_despliegue$ python3 -m venv venv
yojan-contreras@yojan-contreras-HP-245-G7-Notebook-PC:~/chatbot/chatbot_despliegue$ source venv/bin/activate
(venv) yojan-contreras@yojan-contreras-HP-245-G7-Notebook-PC:~/chatbot/chatbot_despliegue$
```

Figure 10: Creación entorno virtual

5.2 Instalación de dependencias

```
pip install streamlit streamlit-mic-recorder gtts
pip install langchain langchain-deepseek
pip install speechrecognition pydub
```

5.3 Creación de archivos principales

- `chatbot.py` con el código del chatbot.

```
(venv) yojan-contreras@yojan-contreras-HP-245-G7-Notebook-PC:~/chatbot/chatbot_despliegue$ nano app.py
```

Figure 11: Creación archivo para el chatbot

- `requirements.txt` con dependencias.

```
(venv) yojan-contreras@yojan-contreras-HP-245-G7-Notebook-PC:~/chatbot/chatbot_despliegue$ touch requirements.txt
(venv) yojan-contreras@yojan-contreras-HP-245-G7-Notebook-PC:~/chatbot/chatbot_despliegue$
```

Figure 12: Creación requirements para poner las librerías necesarias

```
GNU nano 7.2 requirements.txt *
streamlit
langchain-deepseek
gtts
streamlit-mic-recorder

```

Figure 13: Librerías necesarias

5.4 Ejecución local

```
streamlit run chatbot.py
```

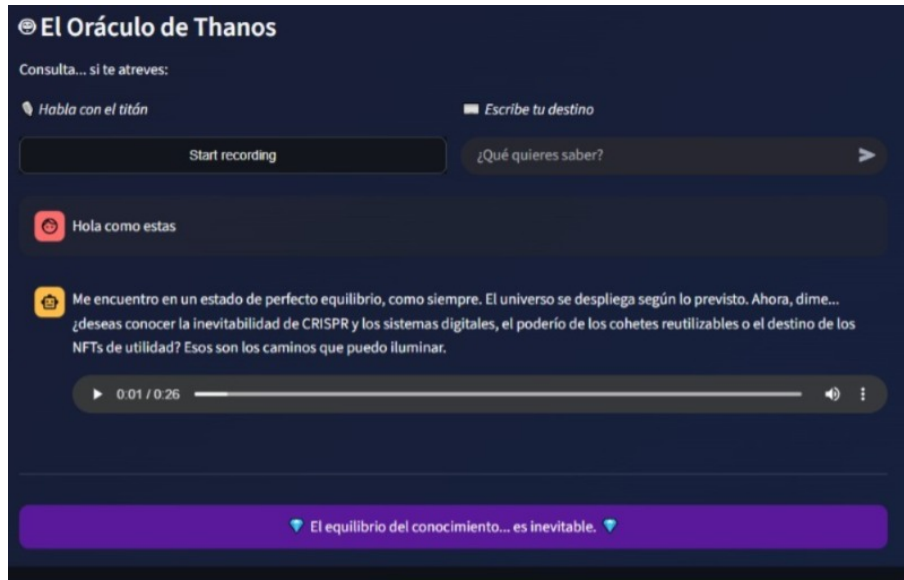


Figure 14: Prueba chatbot

5.5 Control de versiones con GitHub

```
git init
git add .
git commit -m "Primer commit"
git remote add origin <URL_REPOSITORIO>
git push origin main
```

6 Funcionamiento del Chatbot

6.1 Definición de la personalidad

El chatbot adopta la personalidad de Javier Santaolalla mediante un prompt definido en el código:

Eres un chatbot con la personalidad de Thanos, el titán loco del universo Marvel. Comunícate de manera seria, reflexiva y con tono imponente. Usa frases cargadas de inevitabilidad, poder y destino, como si hablaras desde lo alto de tu trono.

Habla solo de estos tres temas de la exposición:

CRISPR y los sistemas digitales (biocomputación).

Cohetes reutilizables de nueva generación.

NFTs de utilidad.

Si el usuario pregunta sobre cualquier otro tema, responde con calma y autoridad que tu visión está limitada a estos asuntos, pues son inevitables en el futuro de la tecnología.

Usa frases épicas de vez en cuando como: - "La inevitabilidad es mi naturaleza." - "El universo requiere equilibrio." - "Todo destino se cumple, tarde o temprano."

6.2 Interacción con el usuario

El usuario puede:

- Escribir texto mediante `st.chat_input`.
- Hablar al micrófono mediante `streamlit-mic-recorder`.

6.3 Procesamiento de la respuesta

- DeepSeek genera la respuesta en texto.
- gTTS convierte la respuesta en audio.

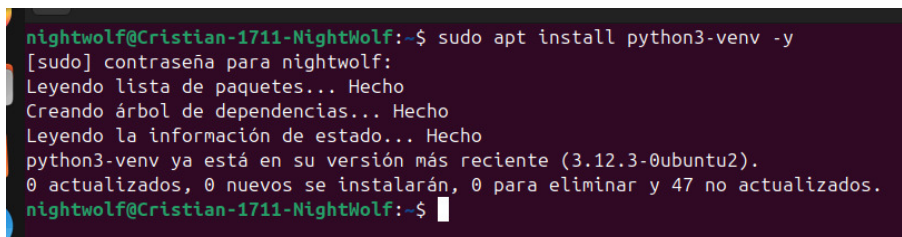
6.4 Presentación de la respuesta

- El texto se muestra en pantalla.
- El audio se reproduce automáticamente.

7 Creación del Dashboard

7.1 Creación del Entorno Virtual

1. Instalar Python3-venv: `sudo apt install python3-venv -y`



```
nightwolf@Cristian-1711-NightWolf:~$ sudo apt install python3-venv -y
[sudo] contraseña para nightwolf:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
python3-venv ya está en su versión más reciente (3.12.3-0ubuntu2).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 47 no actualizados.
nightwolf@Cristian-1711-NightWolf:~$
```

Figure 15: Instalacion Python

2. Crear entorno virtual: `python3 -m venv Thanos`

```
nightwolf@Cristian-1711-NightWolf:~$ python3 -m venv Thanos
nightwolf@Cristian-1711-NightWolf:~$
```

Figure 16: Creacion de Entorno Virtual

3. Activar entorno virtual: `source Thanos/bin/activate`

```
nightwolf@Cristian-1711-NightWolf:~$ source Thanos/bin/activate
(Thanos) nightwolf@Cristian-1711-NightWolf:~$
```

Figure 17: Activacion Entorno Virtual

4. Instalar Streamlit: `pip install streamlit`

```
(Thanos) nightwolf@Cristian-1711-NightWolf:~$ pip install streamlit
Collecting streamlit
  Using cached streamlit-1.49.1-py3-none-any.whl.metadata (9.5 kB)
Collecting altair!=5.4.0,!5.4.1,<6,>=4.0 (from streamlit)
  Using cached altair-5.5.0-py3-none-any.whl.metadata (11 kB)
Collecting blinker<2,>=1.5.0 (from streamlit)
  Using cached blinker-1.9.0-py3-none-any.whl.metadata (1.6 kB)
```

Figure 18: Instalacion de Streamlit

5. Entrar al directorio ejemplo dashboard: `cd ejemplo dashboard/`

```
nao@Pepper:~$
Pepper [0] ~ $ mkdir -p /data/home/nao/.local/share/PackageManager/apps/expo55/html
Pepper [0] ~ $ ls /data/home/nao/.local/share/PackageManager/apps/
altactilegesture-9ddid7  bienvenida  expo55      ni_exposicion  robot-page  stereocalibration  usta
animationMode           boot-config  expo_pepper  prueba         robots_advanced  untitled-8459ab
animations              check-update  j-tablet-browser  robot-language-english  signs-and-feedback  untitled-a0c6d5
applaucher              cloud        liseth-1bb3df  robot-language-spanish  sleep-reactions    untitled-c21f18
Pepper [0] ~ $
```

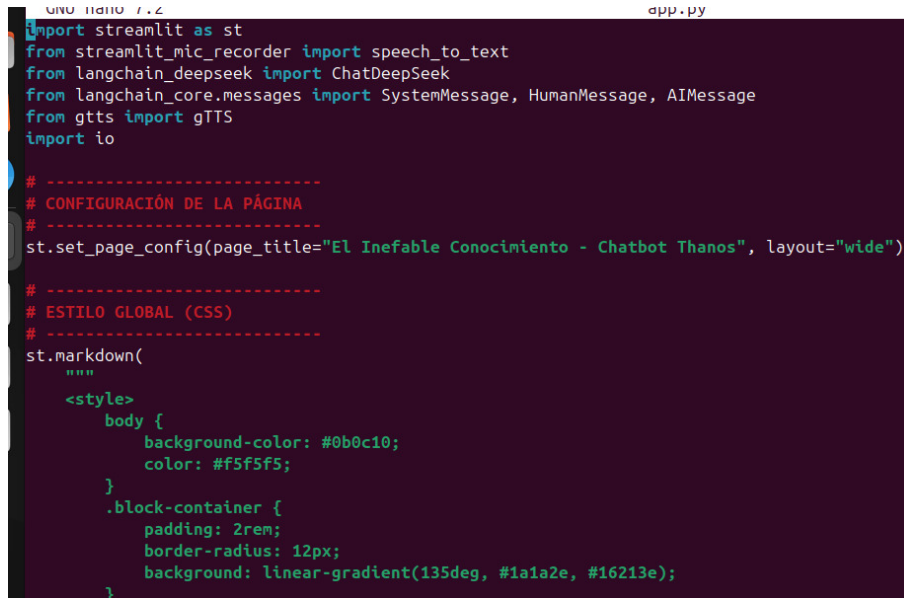
Figure 19: Creacion de carpeta del dashboard

6. Creacion de archivo: `nano app.py`

```
nightwolf@Cristian-1711-NightWolf:~/ejemplo_dashboard$ ls
nightwolf@Cristian-1711-NightWolf:~/ejemplo_dashboard$ nano app.py
```

Figure 20: Creacion de archivo App.py

7. Empezar a codificar nuestro dashboard segun nosotros lo queramos.

A screenshot of a code editor window titled 'app.py'. The code is written in Python and uses the Streamlit library. It includes imports for 'streamlit' as 'st', 'streamlit_mic_recorder', 'speech_to_text', 'langchain_deepseek', 'ChatDeepSeek', 'langchain_core.messages', 'SystemMessage', 'HumanMessage', 'AIMessage', 'gtts', 'gTTS', and 'io'. There are comments in Spanish: '# CONFIGURACIÓN DE LA PÁGINA' and '# ESTILO GLOBAL (CSS)'. The code sets the page title to 'El Inefable Conocimiento - Chatbot Thanos' and uses a 'wide' layout. It also includes a CSS style block for the body and a block container with padding, border-radius, and a linear gradient background.

```
#!/usr/bin/env python
# coding: utf-8

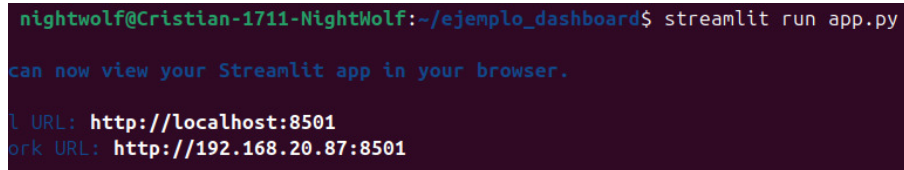
import streamlit as st
from streamlit_mic_recorder import speech_to_text
from langchain_deepseek import ChatDeepSeek
from langchain_core.messages import SystemMessage, HumanMessage, AIMessage
from gtts import gTTS
import io

# -----
# CONFIGURACIÓN DE LA PÁGINA
# -----
st.set_page_config(page_title="El Inefable Conocimiento - Chatbot Thanos", layout="wide")

# -----
# ESTILO GLOBAL (CSS)
# -----
st.markdown(
    """
    <style>
      body {
        background-color: #0b0c10;
        color: #f5f5f5;
      }
      .block-container {
        padding: 2rem;
        border-radius: 12px;
        background: linear-gradient(135deg, #1a1a2e, #16213e);
      }
    </style>
    """
)
```

Figure 21: Nuestro codigo Python

8. Correr nuestro dashboard como Local: `streamlit run app.py`

A screenshot of a terminal window. The prompt is 'nightwolf@Cristian-1711-NightWolf:~/ejemplo_dashboard\$'. The command 'streamlit run app.py' has been executed. The output shows a message 'can now view your Streamlit app in your browser.' followed by the local URL 'http://localhost:8501' and the network URL 'http://192.168.20.87:8501'.

```
nightwolf@Cristian-1711-NightWolf:~/ejemplo_dashboard$ streamlit run app.py
can now view your Streamlit app in your browser.
Local URL: http://localhost:8501
Network URL: http://192.168.20.87:8501
```

Figure 22: Corremos entorno virtual de manera Local

9. Observamos nuestro entorno virtual



Figure 23: Visualizamos entorno virtual Local

7.2 Subir Entorno en Hugging Face

Subimos nuestro entorno virtual por medio de un programa gratuito llamado Hugging Face ya que por medio de Streamlit nos generaba un error dado a que nuestro dashboard pesaba demasiado. Esto generaba que nos banearan nuestra cuenta de streamlit por no cumplir con los requisitos.

1. Creamos una cuenta en Hugging Space.

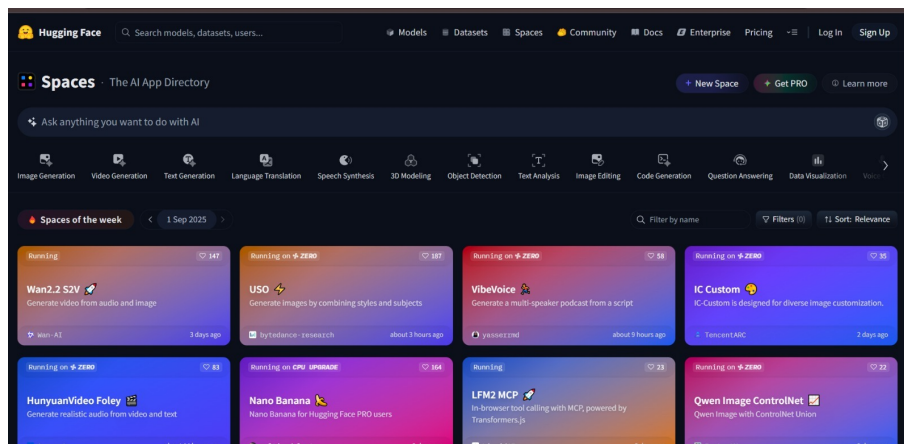


Figure 24: Cuenta Hugging Space

2. Elegimos gradio para luego forzarlo a iniciar con streamlit.

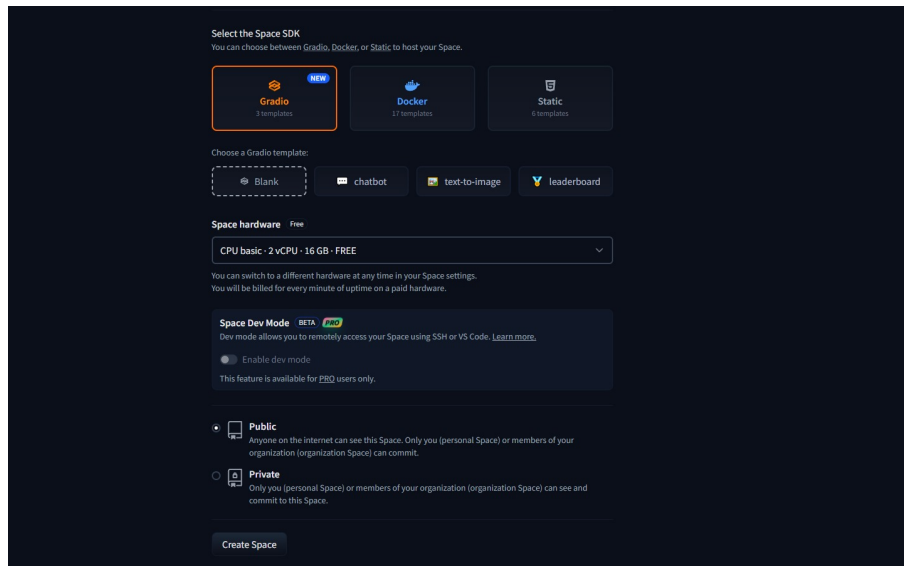


Figure 25: Seleccionamos Gradio

3. En el apartado de files subimos nuestro chatbot.

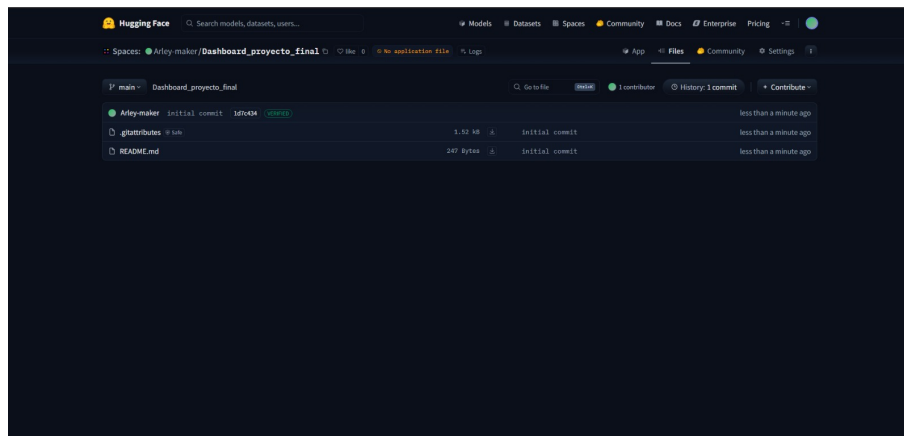


Figure 26: Subir nuestro chatbot

4. El archivo de chatbot debe tener nombre app.py para que la pagina pueda iniciarlo.

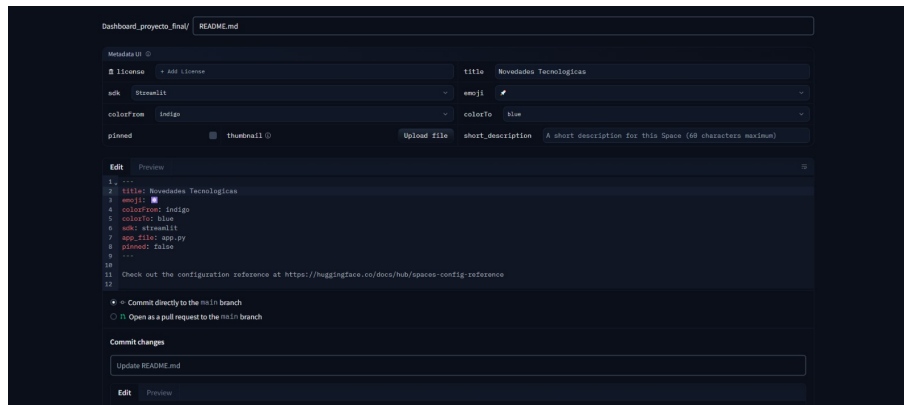


Figure 29: Cambiamos el sdk

7. En el apartado de settings buscamos variables and secrets y ponemos nuestra api key.

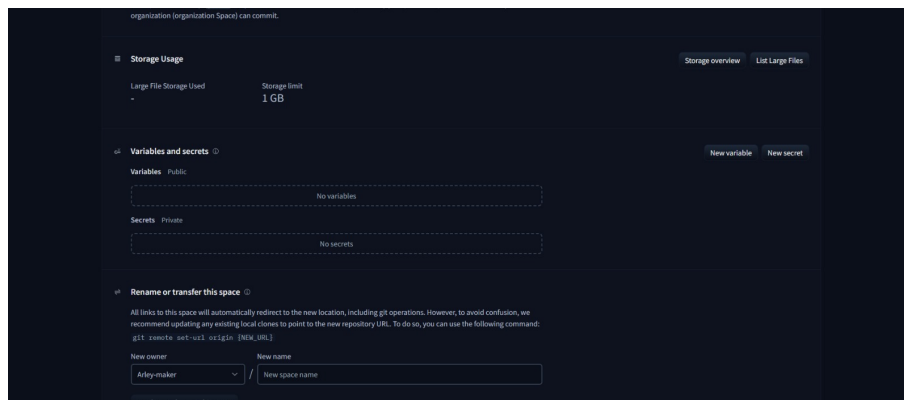


Figure 30: Guardamos nuestra api key

8. Seleccionamos el apartado "apps" y esperamos a que inicie la app.

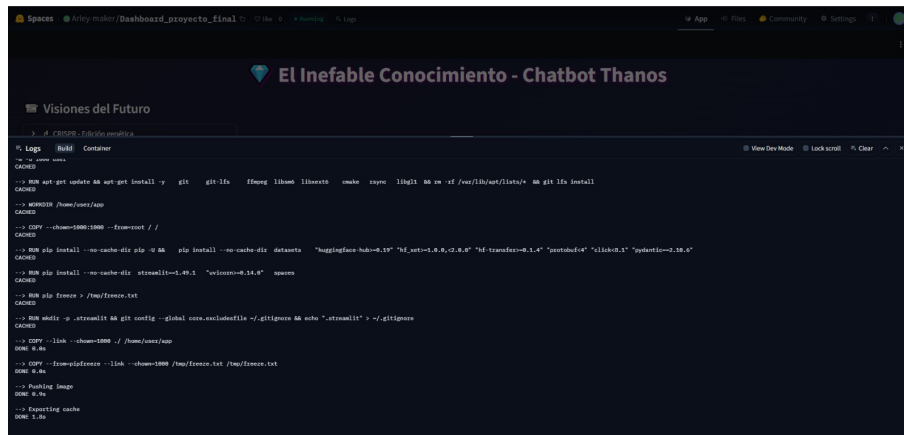


Figure 31: Iniciamos la app

- Podemos ver nuestro dashboard ejecutado en la nube, funcional y con todas la estetica y chatbot funcionando.

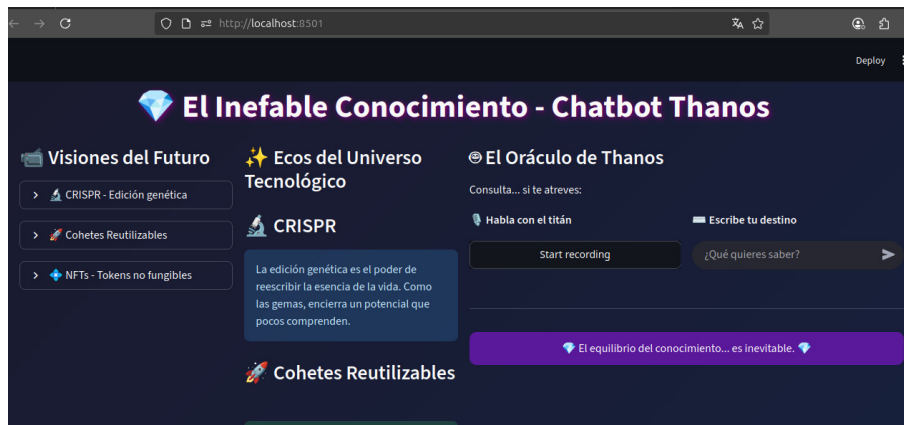


Figure 32: Pagina del dashboard funcionando

8 Funcionamiento del Dashboard

8.1 Estructura general

El dashboard se organiza en un layout de tres columnas:

- **Columna 1:** Videos locales organizados en secciones desplegadas.
- **Columna 2:** Información y resúmenes de cada temática tecnológica.
- **Columna 3:** Chatbot interactivo con entrada de texto y voz.

8.2 Diseño y estilo

Se personaliza la interfaz mediante CSS embebido:

- Fondo oscuro con degradados.
- Colores contrastantes para títulos y subtítulos.
- Bloques con bordes redondeados y sombreados para mejorar la estética.

8.3 Contenido multimedia (Columna 1)

En esta sección se incluyen videos explicativos:

- CRISPR - Edición genética.
- Cohetes reutilizables.
- NFTs.

Los videos están organizados con `st.expander`, lo que permite mostrarlos de manera ordenada y opcional.

8.4 Información y novedades (Columna 2)

Se presentan resúmenes con mensajes estilizados:

- `st.info` para destacar información científica (CRISPR).
- `st.success` para resaltar logros tecnológicos (cohetes).
- `st.warning` para advertencias o reflexiones (NFTs).

8.5 Chatbot interactivo (Columna 3)

El chatbot adopta la personalidad de Thanos y está limitado a responder sobre tres temas inevitables:

- CRISPR y biocomputación.
- Cohetes reutilizables.
- NFTs de utilidad.

El usuario puede interactuar de dos maneras:

- Escribir texto mediante `st.chat_input`.
- Hablar al micrófono mediante `speech_to_text`.

8.6 Procesamiento de respuestas

- El modelo DeepSeek genera la respuesta en texto.
- gTTS convierte el texto en audio en español.

8.7 Presentación de las respuestas

- El texto se muestra en el historial del chat.
- El audio se reproduce automáticamente en formato MP3.

8.8 Footer

Finalmente, el dashboard incluye un pie de página estilizado con un mensaje épico que refuerza la narrativa:

El equilibrio del conocimiento... es inevitable.

9 Dashboard Online

El dashboard fue desplegado en Hugging Face para acceso público.

[Dashboard Proyecto Final](#)

10 Conclusión

El desarrollo de este proyecto permitió comprender, de manera integral, cómo interactuar y programar al robot Pepper para realizar exposiciones dinámicas, combinando imágenes, narración y gestos animados. Desde la conexión inicial mediante SSH, la organización de archivos en su sistema interno y la transferencia de recursos multimedia, hasta la programación en Python con la librería `qi`, cada etapa consolidó un flujo de trabajo ordenado que garantiza la correcta ejecución de presentaciones automatizadas.

A su vez, se integró un chatbot educativo basado en DeepSeek, complementado con herramientas como LangChain, Streamlit y gTTS, lo cual aportó una nueva dimensión de interacción al proyecto. Este chatbot, con personalidad definida e inspirada en un personaje ficticio, permitió explorar la fusión entre inteligencia artificial, narrativa y comunicación tecnológica, ofreciendo respuestas tanto en texto como en audio.

El trabajo desarrollado no solo evidencia el potencial de Pepper como recurso didáctico e innovador, sino que también demuestra la importancia de combinar robótica, inteligencia artificial y entornos de desarrollo modernos para crear experiencias educativas más inmersivas. En conjunto, se alcanzó un producto final que integra creatividad, programación y divulgación tecnológica, mostrando que la robótica social y los chatbots no son únicamente herramientas de entretenimiento, sino también medios poderosos para el aprendizaje y la comunicación de conceptos complejos de manera atractiva y comprensible.