

Proceso de Creación e Implementación de un Chatbot en Linux

Laura Cruz, Andres Saldaña, Jonny Vargas

8 de septiembre de 2025

1. Introducción

Breve descripción del propósito del proyecto: - Uso de Pepper para exposiciones interactivas. - Desarrollo de un chatbot especializado en los temas asignados. - Creación de un Dashboard integrado con videos y chatbot. - Uso de Git y GitHub como herramienta colaborativa.

2. Primer Punto: Interactuando con Pepper

2.1. Objetivo

Exponer tres temas de novedades tecnológicas mediante una presentación de 3 a 5 minutos con Pepper.

2.2. Creación de Carpetas y Archivos

Explicación paso a paso: - Cómo se creó la carpeta del proyecto. - Cómo se organizaron los archivos de código y recursos (ejemplo: `movimientos.py`, `audio/`, `videos/`).

Para organizar y transferir archivos correctamente, es necesario identificar primero la ubicación del directorio donde se encuentra el archivo en nuestro computador. Además, se debe conocer la dirección IP del robot Pepper y la ruta del directorio de destino dentro de este.

```
scp [Archivo.png] nao@192.168.0.104:home/nao/.local/share
/PackageManager/apps/usta/html
```

Para verificar donde se encuentra el archivo, lo único que debemos realizar es ir al terminal directo de pepper y buscar el archivo.

```
home/nao/.local/share/PackageManager/apps/usta/html
```

2.3. Desarrollo del Código

Para desarrollar el código de funcionamiento principal se debe tener en cuenta los movimientos que quieres realizar con pepper, las imágenes de las diapositivas y la información que debe realizar. Para realizar los movimientos se tuvo en cuenta la función:

```
animated_speech_service = session.service("
    ALAnimatedSpeech") animation_service = session.service
    ("ALAnimationPlayer")
animation_service.run("animations/Stand/Gestures/Hey_1")
```

Para realizar la conexión con la tablet y que esta deje ver la foto o diapositiva que se necesita, lo que debemos hacer es comunicar la función con la IP de la tablet y verificar que el archivo este subido a los directorios de pepper. Como adicional se uso un Sleep para la velocidad de la diapositiva

```
tablet_service = session.service("ALTabletService")
tablet_service.showImage("http://198.18.0.1/apps/usta/
    intro.jpg") time.sleep(3)
```

Como ultimo, para que Pepper hable el texto deseado, se usa la función con la cual se acompaña el texto informativo.

```
animated_speech_service.say(u"[Texto]")
animated_speech_service_1=session.service("
    ALAnimatedSpeech")
```

```

import qi
import sys
import time

def main(session):

    motion_service = session.service("ALMotion")
    posture_service = session.service("ALRobotPosture")
    animated_speech_service = session.service("ALAnimatedSpeech")
    animation_service = session.service("ALAnimationPlayer")
    tablet_service = session.service("ALTabletService")

    configuration = {"bodyLanguageMode": "contextual"}

    # — INICIO DE EXPOSICIÓN —

    # Postura inicial
    posture_service.goToPosture("StandInit", 0.5)

    # Introducción con saludo
    animation_service.run("animations/Stand/Gestures/Hey_1")
    animated_speech_service.say(u"Buenos días, hoy les voy a hablar sobre un tema que está revolucionando la medicina regenerativa: las terapias con exosomas modificados.", configuration)
    tablet_service.showImage("http://198.18.0.1/apps/usta/intro.jpg")
    time.sleep(3)

    # Explicación de qué son los exosomas
    animation_service.run("animations/Stand/Gestures/Explain_1")
    animated_speech_service.say(u"Los exosomas son pequeñas vesículas que liberan casi todas las células. Transportan proteínas, lípidos y material genético. Funcionan como mensajeros y tienen la capacidad de modificar la actividad de otras células.", configuration)
    tablet_service.showImage("http://198.18.0.1/apps/usta/exosomas.jpg")
    time.sleep(3)

    # Innovación
    animation_service.run("animations/Stand/Gestures/Enthusiastic_4")
    animated_speech_service.say(u"La verdadera innovación está en que podemos modificarlos. Existen tres formas principales: ingeniería genética de las células madre, carga activa post-aislamiento y modificación de la superficie para dirigirlos a células específicas como las tumorales.", configuration)
    tablet_service.showImage("http://198.18.0.1/apps/usta/innovacion.jpg")
    time.sleep(3)

    # Ventajas
    animation_service.run("animations/Stand/Gestures/Thinking_4")
    animated_speech_service.say(u"¿Qué ventajas tienen frente a terapias convencionales? Son biocompatibles, atraviesan barreras biológicas difíciles, protegen su carga y minimizan los efectos secundarios.", configuration)
    tablet_service.showImage("http://198.18.0.1/apps/usta/ventajas.jpg")
    time.sleep(3)

    # Desafíos
    animation_service.run("animations/Stand/Gestures/CalmDown_1")
    animated_speech_service.say(u"Pero aún existen desafíos: producirlos a gran escala, establecer normas claras de regulación y comprender mejor cómo actúan en el cuerpo.", configuration)
    tablet_service.showImage("http://198.18.0.1/apps/usta/desafios.jpg")
    time.sleep(3)

    # Futuro
    animation_service.run("animations/Stand/Gestures/Enthusiastic_5")
    animated_speech_service.say(u"En el futuro, los exosomas tendrán un papel clave en enfermedades como el Alzheimer, el Parkinson, los infartos cardíacos, enfermedades renales y pulmonares.", configuration)
    tablet_service.showImage("http://198.18.0.1/apps/usta/futuro.jpg")
    time.sleep(3)

    # Conclusión
    animation_service.run("animations/Stand/Gestures/YouKnowWhat_1")
    animated_speech_service.say(u"En conclusión, los exosomas modificados representan el inicio de una nueva era en la medicina. Mas que curar, nos permiten instruir al cuerpo para que se regenere y se defienda.", configuration)
    tablet_service.showImage("http://198.18.0.1/apps/usta/conclusion.jpg")
    time.sleep(3)

    # Despedida
    animation_service.run("animations/Stand/Gestures/Hey_3")
    animated_speech_service.say(u"Muchas gracias por su atención.", configuration)
    tablet_service.showImage("http://198.18.0.1/apps/usta/gracias.jpg")
    time.sleep(3)

    tablet_service.hide()
    posture_service.goToPosture("StandZero", 0.5)

if __name__ == "__main__":
    try:
        connection_url = "tcp://192.168.0.109:9559"
        app = qi.Application(["ExposicionExosomas", "--qi-url=" + connection_url])
    except RuntimeError:
        print("No se pudo conectar con Pepper. Revisa la IP y el puerto.")
        sys.exit(1)

    app.start()
    session = app.session
    main(session) si fuera un video y un audio

```

Figura 1: Código Principal

3. Segundo Punto: Desarrollando Chatbot Personalizado

3.1. Objetivo

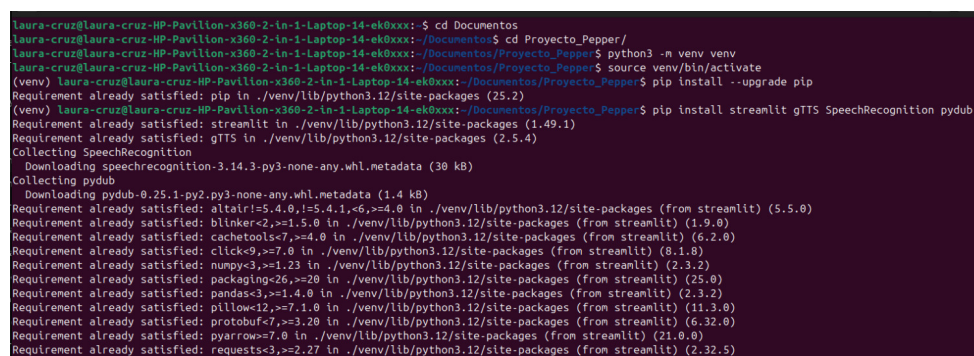
Desarrollar un chatbot que responda exclusivamente sobre los tres temas seleccionados.

3.2. Configuración Inicial

- Para aislar las dependencias del proyecto, se generó un entorno virtual en Python empleando `venv`:

```
python3 -m venv ven
```

- Instalación de librerías necesarias (`pip install requests streamlit gtts`).



```
laura-cruz@laura-cruz-HP-Pavilion-x360-2-in-1-Laptop-14-ek0xxx:~$ cd Documentos
laura-cruz@laura-cruz-HP-Pavilion-x360-2-in-1-Laptop-14-ek0xxx:~/Documentos$ cd Proyecto_Pepper/
laura-cruz@laura-cruz-HP-Pavilion-x360-2-in-1-Laptop-14-ek0xxx:~/Documentos/Proyecto_Pepper$ python3 -m venv venv
laura-cruz@laura-cruz-HP-Pavilion-x360-2-in-1-Laptop-14-ek0xxx:~/Documentos/Proyecto_Pepper$ source venv/bin/activate
(venv) laura-cruz@laura-cruz-HP-Pavilion-x360-2-in-1-Laptop-14-ek0xxx:~/Documentos/Proyecto_Pepper$ pip install --upgrade pip
Requirement already satisfied: pip in ./venv/lib/python3.12/site-packages (25.2)
(venv) laura-cruz@laura-cruz-HP-Pavilion-x360-2-in-1-Laptop-14-ek0xxx:~/Documentos/Proyecto_Pepper$ pip install streamlit gtts SpeechRecognition pydub
Requirement already satisfied: streamlit in ./venv/lib/python3.12/site-packages (1.49.1)
Requirement already satisfied: gtts in ./venv/lib/python3.12/site-packages (2.5.4)
Collecting SpeechRecognition
  Downloading speechrecognition-3.14.3-py3-none-any.whl.metadata (38 kB)
Collecting pydub
  Downloading pydub-0.25.1-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: altair!=5.4.0,!=5.4.1,<6,>=4.0 in ./venv/lib/python3.12/site-packages (from streamlit) (5.5.0)
Requirement already satisfied: blinker<2,>=1.5.0 in ./venv/lib/python3.12/site-packages (from streamlit) (1.9.0)
Requirement already satisfied: cachetools<7,>=4.0 in ./venv/lib/python3.12/site-packages (from streamlit) (6.2.0)
Requirement already satisfied: click<9,>=7.0 in ./venv/lib/python3.12/site-packages (from streamlit) (8.1.8)
Requirement already satisfied: numpy<3,>=1.23 in ./venv/lib/python3.12/site-packages (from streamlit) (2.3.2)
Requirement already satisfied: packaging<26,>=20 in ./venv/lib/python3.12/site-packages (from streamlit) (25.0)
Requirement already satisfied: pandas<3,>=1.4.0 in ./venv/lib/python3.12/site-packages (from streamlit) (2.3.2)
Requirement already satisfied: pillow<12,>=7.1.0 in ./venv/lib/python3.12/site-packages (from streamlit) (11.3.0)
Requirement already satisfied: protobuf<7,>=3.20 in ./venv/lib/python3.12/site-packages (from streamlit) (6.32.0)
Requirement already satisfied: pyarrow<=7.0 in ./venv/lib/python3.12/site-packages (from streamlit) (21.0.0)
Requirement already satisfied: requests<3,>=2.27 in ./venv/lib/python3.12/site-packages (from streamlit) (2.32.5)
```

Figura 2: Coreografía en terminal

- Creación de el archivo del chatbot Se utilizó el editor de texto `nano` para crear y editar el archivo Python principal (`chatbot.py`):

```
nano chatbot.py
```

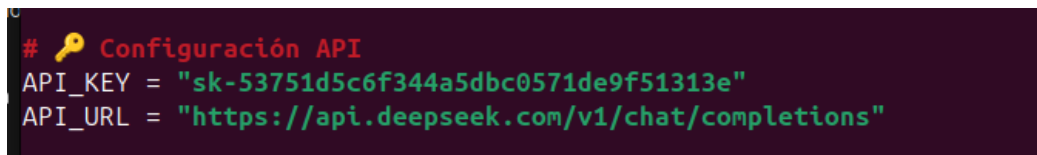
Dentro del archivo se escribió el código correspondiente al chatbot. Una vez completado, se guardó y se cerró el editor presionando:

- CTRL + O (guardar cambios)
- ENTER (confirmar nombre de archivo)

- CTRL + X (salir del editor)

3.3. Explicacion de codigo

- Configuración de la API
 - Aquí se guarda la clave privada que permite conectarse al modelo de lenguaje (DeepSeek).
 - API URL indica la dirección del servicio al que se le envían las preguntas.
 - Sin esta configuración, el chatbot no podría comunicarse con el modelo.

A screenshot of a terminal window with a dark purple background. The text is displayed in a monospaced font with syntax highlighting. The first line is a comment: "# 🔑 Configuración API". The second line is "API_KEY = \"sk-53751d5c6f344a5dbc0571de9f51313e\"", where the key is in green. The third line is "API_URL = \"https://api.deepseek.com/v1/chat/completions\"", where the URL is in green. The terminal has a small cursor icon at the end of the third line.

```
# 🔑 Configuración API
API_KEY = "sk-53751d5c6f344a5dbc0571de9f51313e"
API_URL = "https://api.deepseek.com/v1/chat/completions"
```

Figura 3: Coreografía en terminal

- Función enviar mensaje
 - Esta es la función principal del chatbot.
 - Recibe un mensaje del usuario y lo envía a la API.
 - El parámetro "system" define la personalidad del chatbot (en este caso, experto en innovación tecnológica).
 - El parámetro user es la pregunta del estudiante.
 - Devuelve la respuesta generada por la IA.
 - Si hay un error, devuelve un mensaje de advertencia.
- Función reproducir audio
 - Convierte un archivo de audio en código base64 para poder incrustarlo en el navegador.
 - Reproduce automáticamente la respuesta del chatbot sin mostrar un reproductor visible.

```

# 📡 Función para enviar mensajes
def enviar_mensaje(mensaje, modelo="deepseek-chat"):
    headers = {
        "Authorization": f"Bearer {API_KEY}",
        "Content-Type": "application/json"
    }
    data = {
        "model": modelo,
        "messages": [
            {"role": "system", "content": "Responde como un experto en temas de innovación tecnológica. " "Tu especialidad son tres áreas clave: " "1) Energía undinotri"},
            {"role": "user", "content": mensaje}
        ]
    }
    response = requests.post(API_URL, headers=headers, json=data)
    result = response.json()
    if "choices" in result:
        return result["choices"][0]["message"]["content"]
    else:
        return f"⚠️Error en la API: {result}"

```

Figura 4: Coreografía en terminal

```

# 🎵 Función para reproducir audio
def reproducir_audio(archivo):
    with open(archivo, "rb") as f:
        audio_bytes = f.read()
    audio_b64 = base64.b64encode(audio_bytes).decode()
    st.markdown(
        f"""
        <audio autoplay style="display:none;">
        <source src="data:audio/mp3;base64,{audio_b64}" type="audio/mp3">
        </audio>
        """
    )
    unsafe_allow_html=True

```

Figura 5: Coreografía en terminal

- Permite que la experiencia sea más natural: el usuario escribe y Pepper responde hablando al instante.

- Generación de audio con gTTS
 - Usa la librería gTTS (Google Text-to-Speech) para convertir el texto en voz.
 - Crea un archivo temporal en formato MP3.
 - Llama a la función reproducir audio para que el archivo se escuche automáticamente.
- Manejo del historial de conversación

```
# 🗣️ Generar audio invisible
tts = gTTS(respuesta, lang="es")
with tempfile.NamedTemporaryFile(delete=False, suffix=".mp3") as tmpfile:
    tts.save(tmpfile.name)
    reproducir_audio(tmpfile.name) # 🎵 Sonará automáticamente sin barra
```

Figura 6: Coreografía en terminal

- Streamlit borra los datos cada vez que se recarga la página.
- Con `st.session_state` se guarda el historial de mensajes.
- Así el chatbot mantiene la conversación completa (usuario y asistente).

```
# 🧠 Manejo de historial
if "messages" not in st.session_state:
    st.session_state.messages = []
```

Figura 7: Coreografía en terminal

4. Tercer Punto: Desarrollando Dashboard Integrado

4.1. Objetivo

Integrar en una interfaz los videos de Pepper y el chatbot utilizando Streamlit.

4.2. Creación del Dashboard

- Para la creación de la dashboard primero se creó una carpeta llamada de esta forma y dentro de ella se creó el archivo de python (`app.py`):

```
mkdir dashboard
nano app.py
```

A continuación se explicarán las funciones más importantes en el archivo (`app.py`):

- Integración de videos - Aquí se maneja la parte visual, donde el usuario selecciona y reproduce un video de Pepper.
 - `st.selectbox('Elige un video:', list(videos.keys()))`
 - Muestra un menú desplegable con los nombres de los videos disponibles.
 - El usuario elige entre Video 1, Video 2 o Video 3.
 - `st.button("Start Video")`.
 - Botón que inicia la reproducción del video seleccionado.
 - Solo se activa si el usuario hace clic.
 - `st.video(video path)`.
 - Reproduce el video dentro del Dashboard.
 - Puede soportar varios formatos
- Presentación de novedades - Es la parte informativa, donde se describen las tres innovaciones tecnológicas que como grupo se investigó
 - `st.markdown("Novedades Tecnológicas")`
 - Muestra el título general de la sección.
 - Se usa Markdown para personalizar el estilo (tamaño, emojis, color, etc.).
 - `st.subheader('Energía undimotriz inteligente')`
 - Crea subtítulos para separar cada tema tecnológico.
 - Se numeran y acompañan con emojis para hacerlo visualmente atractivo.
 - `st.info("Texto descriptivo")`

```

# =====
# 📺 Columna Izquierda: Videos
# =====
with col1:
    st.markdown("### 📺 Videos de Pepper")

    videos = {
        "Video 1": "Video_1.mp4",
        "Video 2": "Video_2.mp4",
        "Video 3": "Video_3.mp4"
    }

    seleccion = st.selectbox("Elige un video:", list(videos.keys()))

    if st.button("▶ Start Video"):
        video_path = os.path.join(ROOT, videos[seleccion])
        if os.path.exists(video_path):
            st.video(video_path)
        else:
            st.error(f"No se encontró {videos[seleccion]} en la carpeta principal.")

```

Figura 8: Coreografía en terminal

```

st.error(f"No se encontró {videos[seleccion]} en la carpeta principal.")
# =====
# 📺 Columna Centro: Novedades
# =====
with col2:
    st.markdown("### 📺 Novedades Tecnológicas")

    st.subheader("🔋 Energía undimotriz inteligente")
    st.info("Aprovecha el movimiento de las olas para generar energía limpia, integrándose con sistemas digitales que optimizan la producción y el control en tiempo real.")

    st.subheader("🤖 Robots blandos (Soft Robotics)")
    st.info("Robots inspirados en organismos vivos, hechos con materiales flexibles. Los sistemas digitales permiten control de movimiento preciso y aplicaciones innovadoras.")

    st.subheader("🦋 Terapias con exosomas modificados")
    st.info("Exosomas alterados genéticamente aplicados como terapias innovadoras, con monitoreo digital para evaluar y mejorar resultados en salud.")

```

Figura 9: Coreografía en terminal

- Crea cajas de información en color azul claro con las descripciones de cada innovación.
 - Sirve para resaltar y organizar el texto.
- Chatbot interactivo -En esta parte convierte el Dashboard en un chat interactivo en tiempo real, donde el usuario escribe su pregunta, recibe la respuesta escrita y escucha la voz de Pepper automáticamente. Como ya se describió anteriormente.

4.3. Ejecución del Dashboard

- Por ultimo lo que se hace es la ejecucion de la consola con `streamlit run app.py` y ver el funcionamiento de la interfaz.



Figura 10: Coreografía en terminal

5. Manejo de Git y GitHub en el Proyecto

5.1. Clonar el repositorio

Se clonó el repositorio remoto en GitHub hacia la carpeta local de trabajo:

```
cd Documentos/
git clone git@github.com:Lau-raCrz/Proyecto_corte1.git
```

Esto descargó el proyecto inicial en la carpeta Proyecto_corte1.

5.2. Configuración inicial en el proyecto

Dentro de la carpeta del proyecto local Proyecto_Pepper, se revisaron los archivos ocultos y se creó un archivo `.gitignore` para excluir archivos innecesarios en el control de versiones:

```
ls -a
nano .gitignore
```

Luego se activó el entorno virtual y se reinició el repositorio Git:

```
source venv/bin/activate
git init
```

5.3. Configurar repositorio remoto

Se eliminó el origin anterior y se agregó el repositorio correcto:

```
git remote remove origin
git remote add origin git@github.com:Lau-raCrz/
    Proyecto_corte1.git
git remote -v
```

5.4. Creación de rama personal

Cada estudiante debía trabajar en su propia rama. En este caso, se creó la rama `laura/video-pepper`:

```
git checkout -b laura/video-pepper
```

5.5. Subida de archivos personales

Se creó una carpeta `videos` y se copió un archivo de video dentro:

```
mkdir videos
cp ~/Documentos/Proyecto_Pepper/Video_3.mp4 videos/
    Video_Laura.mp4
```

Después se añadieron los archivos al área de *staging* y se hizo el primer commit:

```
git add videos/Video_Laura.mp4 dashboard/app.py .
    gitignore
git commit -m "Agrego video laura"
```

Finalmente, se subió al repositorio remoto:

```
git push -u origin laura/video-pepper
```

5.6. Sincronización con la rama principal

Se trajeron los cambios de la rama principal (`main`) desde GitHub:

```
git fetch origin main
git checkout main
git pull origin main
```

5.7. Fusión de ramas (Merge)

Para integrar el trabajo de la rama personal en la rama principal, se hizo un merge. Como las ramas tenían historiales distintos, fue necesario permitir fusión de historias no relacionadas:

```
git merge laura/video-pepper --allow-unrelated-histories
```

Esto unió los cambios de `laura/video-pepper` en `main`, incluyendo:

- `dashboard/app.py` (el código del dashboard).
- `.gitignore` (archivo de configuración).
- `videos/Video_Laura.mp4` (video agregado).

5.8. Subida final al repositorio

Después de la fusión, se subieron los cambios de la rama principal a GitHub:

```
git push origin main
```