Agente de Voz con LiveKit Cloud

Este proyecto implementa un agente de voz inteligente utilizando LiveKit Cloud, con capacidades de reconocimiento de voz, procesamiento de lenguaje natural y síntesis de voz.

Características

- **Reconocimiento de Voz (STT)**: Utiliza Deepgram Nova-3 para reconocimiento de voz en español
- **Procesamiento de Lenguaje**: Integrado con OpenAI GPT-4.1-mini para respuestas inteligentes
- **Síntesis de Voz (TTS)**: Utiliza Cartesia Sonic-2 para generar respuestas de voz naturales
- **Detección de Turnos**: Implementa detección multilingüe para conversaciones fluidas
- **Cancelación de Ruido**: Filtrado de ruido de fondo para mejor calidad de audio

Requisitos Previos

- **Python 3.12+**: Asegúrate de tener Python 3.12 o superior instalado
- **Cuenta de LiveKit Cloud**: Regístrate en LiveKit Cloud
- **Cuentas de API**: Necesitarás claves API para:
  - OpenAI (para GPT-4.1-mini)
  - Deepgram (para reconocimiento de voz)
  - Cartesia (para síntesis de voz)

Instalación

1. Instalar LiveKit CLI

Windows

```
winget install LiveKit.LiveKitCLI
```

macOS

```
brew install livekit-cli
```

Linux

```
curl -sSL https://get.livekit.io/cli | bash
```

2. Instalar uv (Gestor de Paquetes Python)

Windows (PowerShell)

```
# Instalar uv
powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"

# Agregar uv al PATH
$env:Path = "C:\Users\$env:USERNAME\.local\bin;$env:Path"
```

macOS/Linux

```
curl -LsSf https://astral.sh/uv/install.sh | sh
```

3. Configurar el Proyecto

```
# Crear nuevo proyecto con uv
uv init livekit-voice-agent --bare
cd livekit-voice-agent

# Instalar dependencias
uv add \
  "livekit-agents[silero,turn-detector]~=1.2" \
  "livekit-plugins-noise-cancellation~=0.2" \
  "livekit-plugins-deepgram" \
  "python-dotenv" \
  "tensorflow"
```

4. Configurar Variables de Entorno

Crea un archivo `.env.local` en el directorio del proyecto:

```
# LiveKit Cloud Configuration
LIVEKIT_URL=wss://your-project.livekit.cloud
LIVEKIT_API_KEY=your_api_key
LIVEKIT_API_SECRET=your_api_secret

# OpenAI Configuration
OPENAI_API_KEY=your_openai_api_key

# Deepgram Configuration
DEEPGRAM_API_KEY=your_deepgram_api_key

# Cartesia Configuration
CARTESIA_API_KEY=your_cartesia_api_key
```

5. Autenticar con LiveKit Cloud

```
# Autenticar con LiveKit Cloud
lk cloud auth

# Configurar variables de entorno del proyecto
lk app env -w
```

Uso

Modo Desarrollo

Para probar el agente localmente:

```
# Descargar archivos de modelos necesarios
uv run agent.py download-files

# Ejecutar en modo desarrollo
uv run agent.py dev
```

Despliegue en LiveKit Cloud

Para desplegar el agente en la nube:

```
# Crear y desplegar el agente
lk agent create
```

Este comando:

- Genera los archivos `Dockerfile`, `.dockerignore` y `livekit.toml`
- Registra el agente en tu proyecto de LiveKit Cloud
- Despliega el agente automáticamente

Pruebas

Una vez desplegado, puedes probar tu agente usando:

1. **Agents Playground**: Accede a LiveKit Agents Playground
2. **Aplicaciones de Cliente**: Integra con aplicaciones web, móviles o de escritorio
3. **Telefonía**: Configura para llamadas telefónicas

Estructura del Proyecto

```
livekit-voice-agent/
├── agent.py              # Código principal del agente
├── pyproject.toml        # Configuración del proyecto
├── uv.lock               # Archivo de bloqueo de dependencias
├── livekit.toml          # Configuración de LiveKit (generado)
├── Dockerfile            # Imagen Docker (generado)
├── .dockerignore         # Archivos ignorados en Docker (generado)
└── .env.local           # Variables de entorno (crear manualmente)
```

Configuración Avanzada

Personalizar el Agente

Puedes modificar el comportamiento del agente editando la clase `Assistant` en `agent.py`:

```
class Assistant(Agent):
    def __init__(self) -> None:
        super().__init__(
            instructions="""Tus instrucciones personalizadas aquí.
            Define cómo debe comportarse tu agente de voz."""
        )
```

Configurar Idiomas

Para cambiar el idioma del reconocimiento de voz:

```
stt=deepgram.STT(model="nova-3", language="es")  # Español
stt=deepgram.STT(model="nova-3", language="en")  # Inglés
```

Configurar Modelos

Puedes cambiar los modelos utilizados:

```
session = AgentSession(
    stt=deepgram.STT(model="nova-3", language="es"),
    llm="openai/gpt-4.1-mini",  # Cambiar modelo de LLM
    tts="cartesia/sonic-2:5c5ad5e7-1020-476b-8b91-fdcbe9cc313c",  # Cambiar TTS
    # ... otras configuraciones
)
```

Solución de Problemas

Errores Comunes

1. **Error de autenticación**: Verifica que las claves API sean correctas
2. **Modelos no encontrados**: Ejecuta `uv run agent.py download-files`
3. **Problemas de conectividad**: Verifica la URL de LiveKit Cloud

Logs y Debugging

```
# Ver logs detallados
uv run agent.py dev --verbose

# Modo debug
uv run agent.py dev --debug
```

Recursos Adicionales

- Documentación Oficial de LiveKit Agents
- LiveKit Cloud Dashboard
- Agents Playground
- API Reference

Contribuir

1. Fork el proyecto
2. Crea una rama para tu feature (`git checkout -b feature/AmazingFeature`)
3. Commit tus cambios (`git commit -m 'Add some AmazingFeature'`)
4. Push a la rama (`git push origin feature/AmazingFeature`)
5. Abre un Pull Request

Licencia

Este proyecto está bajo la Licencia MIT. Ver el archivo `LICENSE` para más detalles.

Nota: Asegúrate de mantener tus claves API seguras y nunca las compartas públicamente.